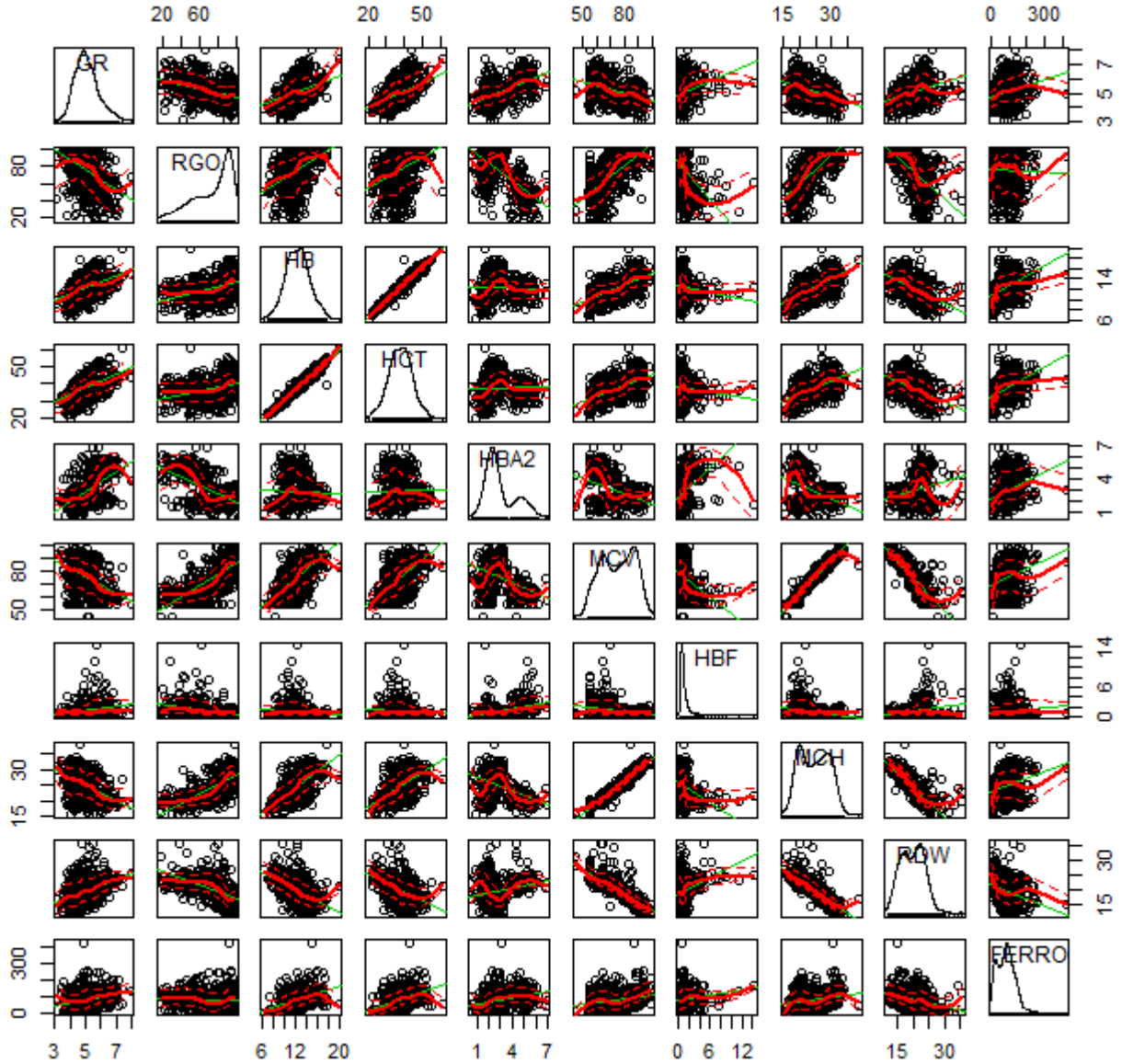
M. Besozzi

STATISTICA CON R
PER IL LABORATORIO
DI ANALISI CLINICHE



Introduzione

Perché usare **R**, se esistono già MedCalc, Minitab, SAS, SPSS, Stata, Systat e decine di altri programmi di statistica, oltre ad add-on per fare statistica con Excel (come ad esempio Analyse-it) e per fare statistica con Access (come ad esempio Total Access Statistics)?

Per almeno quattro buone ragioni:
→ **R** è gratuito, e questo fatto è significativo, visto il costo di un pacchetto di statistica;
→ **R** è disponibile per tutti e tre i sistemi operativi più diffusi, cioè per Windows, per MacOS X e per Linux;
→ una volta installata la versione base di **R** potete accedere a una raccolta di migliaia di pacchetti aggiuntivi, che offrono soluzioni di calcolo ed elaborazioni statistiche e grafiche per (praticamente) qualsiasi problema;
→ se non trovate quello che vi serve tra i pacchetti già disponibili (cosa poco probabile), **R** vi permette comunque di riutilizzare e adattare migliaia di funzioni già scritte e di scriverne di nuove per personalizzare un pacchetto già esistente ovvero per creare ex-novo il vostro pacchetto di analisi dei dati, orientato al vostro specifico problema.

R non è, come asseriscono gli stessi curatori del progetto, un semplice programma di statistica, ma è un ambiente per lo sviluppo dell'analisi statistica e grafica dei dati, per il quale il merito va alla bravura e all'impegno dell'**R** Development Core Team[1]. Software e documentazione di **R** sono open-source e sono rilasciati gratuitamente nei termini previsti dalla Free Software Foundation[2].

Il problema cruciale di **R** è che è basato su un linguaggio di programmazione, è quindi molto tecnico, e per questo è difficile superare le difficoltà iniziali, al punto che molti rinunciano ad utilizzarlo: e il senso di frustrazione che ne consegue è tale da allontanare in genere definitivamente il soggetto da successivi tentativi. Lo scopo di questo manuale, che riprende i contenuti del corso base di **R** che si trova sul mio sito[3], è proprio quello di aiutare a superare queste difficoltà iniziali, per favorire l'accesso al mondo di **R**. Coloro che, operando nel laboratorio di analisi cliniche, hanno la necessità di affrontare qualche problema di analisi dei propri dati, troveranno in **R** uno strumento straordinario, ma soprattutto scopriranno con il tempo di avere fatto un investimento strategico per il proprio sviluppo professionale.

[1] R Development Core Team (2009). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL http://www.R-project.org
[2] vedere http://www.gnu.org/ alla voce Licenze.
[3] http://www.bayes.it/html/statistica_con_r.html

1. Istruzioni rapide per l'uso

Per imparare ad usare **R**, oltre ovviamente a un interesse per la statistica, servono:
→ il programma
→ i dati
→ gli script
→ la documentazione.

1.1. Il programma

Come installare e utilizzare il **programma R** è descritto nel capitolo **2. R installazione e funzioni base**.

1.2. I dati

I **dati** relativi a specifici problemi del laboratorio clinico sono contenuti in file che potete scaricare aprendo i link che verranno forniti illustrando ciascun problema e la sua soluzione in **R**. Potete anche scaricare tutti i dati di esempio in una sola volta in questo modo:
→ create la cartella `C:\R\`;
→ salvate in questa cartella il file csv.zip;
→ scompattate il file csv.zip nella cartella `C:\R`.

Tutti gli script che trovate nei capitoli che seguono fanno riferimento ai nomi dei file che avrete scompattato e alla posizione `C:\R`. Potete ovviamente cambiare la posizione nella quale salvare i dati correggendo opportunamente gli script.

Considerate infine che in **R** il separatore decimale è il punto (**.**) e che questa convenzione per ragioni di omogeneità viene qui adottata oltre che nei file di dati anche in tutto il testo che segue.

1.3. Gli script

Gli **script**, cioè le sequenze di codice **R** che risolvono gli specifici problemi impiegando i dati forniti, li trovate nei capitoli **3. R funzioni statistiche**, **4. R funzioni grafiche** e **5. R problemi scelti**. Questi capitoli rappresentano il nucleo del manuale in quanto le sequenze di codice possono essere eseguite immediatamente copiandole dal testo nella `Console di R`. Possono anche essere salvate sotto forma di file di testo per realizzare una vera e propria libreria di script da utilizzare e adattare al bisogno in un secondo tempo.

1.4. La documentazione

La documentazione necessaria per utilizzare rapidamente **R** è contenuta in questo manuale. Ma si tratta ovviamente di una documentazione limitata e finalizzata all'apprendimento delle basi elementari di **R** per chi opera in un laboratorio di analisi cliniche. Sul sito ufficiale di **R**[4] trovate la sezione `Documentation`:

[4] http://www.r-project.org/

qui potete fare click su `Manuals` e scaricare, in formato pdf, tutti i manuali relativi alla versione di **R** che avete installato, ovvero da qui potete accedere alla sezione `Contributed documentation` nella quale trovate anche manuali in lingua italiana. Anche se in questo corso base non si fa diretto ricorso alla documentazione disponibile sul sito di **R**, questa risulterà ovviamente essenziale per affrontare le domande che inevitabilmente vi sorgeranno durante l'uso del programma, in particolare quando vorrete adattare gli script alle vostre esigenze.

In aggiunta vi possono aiutare nel processo di apprendimento di **R**:
→ il sito Quick-R for SAS/SPSS/Stata Users di Robert I. Kabacoff;
→ il sito R Graphical Manual di Osamu Ogasawara;
→ il sito R-Tutorial di Chi Yau;
→ il sito R-Forge, la piattaforma di sviluppo riservata alla comunità **R**;
→ il blog Revolutions, interamente dedicato alle notizie e informazioni per I membri della comunità **R**.

Della sterminata bibliografia di **R** cito solo alcuni manuali riguardanti argomenti generali, per i quali un ringraziamento va agli Autori che li hanno pubblicati consentendone il libero utilizzo a scopo didattico:
→ Introduzione ad R di Roberto Boggiani;
→ Esercitazioni di statistica biomedica di Matteo Dell'Omodarme;
→ Formulario di Statistica con R di Fabio Frascati;
→ Applied Statistics for Bioinformatics using R di Wim P. Krijnen;
→ Using R for Data Analysis and Graphics - Introduction, Code and Commentary di J. H. Maindonald;
→ Una guida all'utilizzo dell'ambiente statistico R di Angelo M. Mineo;
→ R: un ambiente opensource per l'analisi statistica dei dati di Vito Ricci;
→ Analisi delle serie storiche con R di Vito Ricci;
→ Rappresentazione analitica delle distribuzioni statistiche con R di Vito Ricci;
→ Principali tecniche di regressione con R di Vito Ricci;
→ simpleR - Using R for Introductory Statistics di John Verzani.

2. R installazione e funzioni base

La prima cosa da fare se volete cimentarvi con **R** è collegarvi al sito ufficiale di **R**[5], che trovate anche digitando semplicemente la lettera dell'alfabeto **R** sul motore di ricerca Google, per effettuare il download dell'ultima versione del programma. Vi verrà richiesto da quale dei siti del CRAN (Comprehensive **R** Archive Network), in Italia o in una delle nazioni che trovate elencate, volete effettuare il download. Sul sito dal quale avrete deciso di effettuare il download, qualsiasi esso sia, troverete i link alle versioni di **R** precompilate per i tre principali sistemi operativi: Linux, MacOS X e Windows.

Nella pagina di download selezionate la sola distribuzione/installazione base (dei pacchetti aggiuntivi parleremo tra poco), salvatela sul vostro disco in un posto sicuro (il file avrà un nome del tipo `R-3.0.0-win.exe`, dove invece di 3.0.0 troverete la sigla dell'ultima versione aggiornata che avete appena scaricato). Terminato il download fate doppio click sul file scaricato e seguite le istruzioni per installare **R** sul vostro PC.

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 la sigla x64.

2.1. La Console di R

Fate doppio click sull'icona con il collegamento a **R** per avviare il programma, e vi apparirà la `Console di` R (**Figura 2.1**) con la quale inizieremo a lavorare.

Il simbolo **>** è il `prompt`, e indica che **R** è in attesa che scriviamo che cosa vogliamo fare tramite la tastiera (come faremo tra poco) e nel suo linguaggio (del quale cercheremo di apprendere le basi).

Le prime due cose che facciamo con **R** sono molto semplici:
→ al `prompt` di **R** scrivete **demo()**, premete `invio`, e vi comparirà un elenco di dimostrazioni disponibili nel pacchetto base appena installato;
→ scrivete ora **demo(graphics)**, premete `invio`, seguite le istruzioni che compaiono nella `Console di` **R**, e potrete apprezzare una dimostrazione delle capacità grafiche della installazione base. Nei pacchetti aggiuntivi disponibili per **R** (vedere appresso), troverete una infinità di altre possibilità di elaborazione grafica;

Come avete potuto constatare **R** nella versione base è fornito di una interfaccia molto semplice. Si tratta infatti di una interfaccia a carattere o CUI (`Character User Interface`) che prevede l'immissione delle istruzioni una linea per volta per volta mediante una linea di comando o CLI (`Command Line Interface`) ci riporta agli interpreti di comandi di alcuni decenni fa. E potrebbe apparire obsoleta oggi che l'informatica ci ha abituati ad utilizzare esclusivamente interfacce grafiche o GUI (`Graphical User Interface`).

In effetti anche per **R** è disponibile una GUI che consente all'utente di interagire con R con una interfaccia "evoluta", simile a quella degli altri programmi di statistica in ambiente Windows[6]. Tuttavia qui verrà

[5] http://www.r-project.org/

[6] La GUI ufficiale di **R** è R `Commander`. Per installarla dovete collegarvi a Internet e scaricare la libreria `Rcmdr`. **R** si collegherà al CRAN prescelto per scaricare il pacchetto `Rcmdr` che volete installare, ma attenzione: `Rcmdr` presuppone che siano installati sul PC altri pacchetti, cui esso si appoggia. Per questo vedrete effettuare

utilizzata esclusivamente l'interfaccia a carattere della `Console di R`. Questo è utile a scopo didattico, in quanto consente di leggere in chiaro le istruzioni e quindi di apprendere sintassi e regole del linguaggio **R**. Ma è utile anche perché semplicemente copiando le istruzioni in file di testo sarà possibile salvarle per poi modificarle al bisogno e riutilizzarle, e costruirsi così la propria libreria di script in linguaggio **R** seguendo la sempre efficace metodologia dell'imparare-facendo.

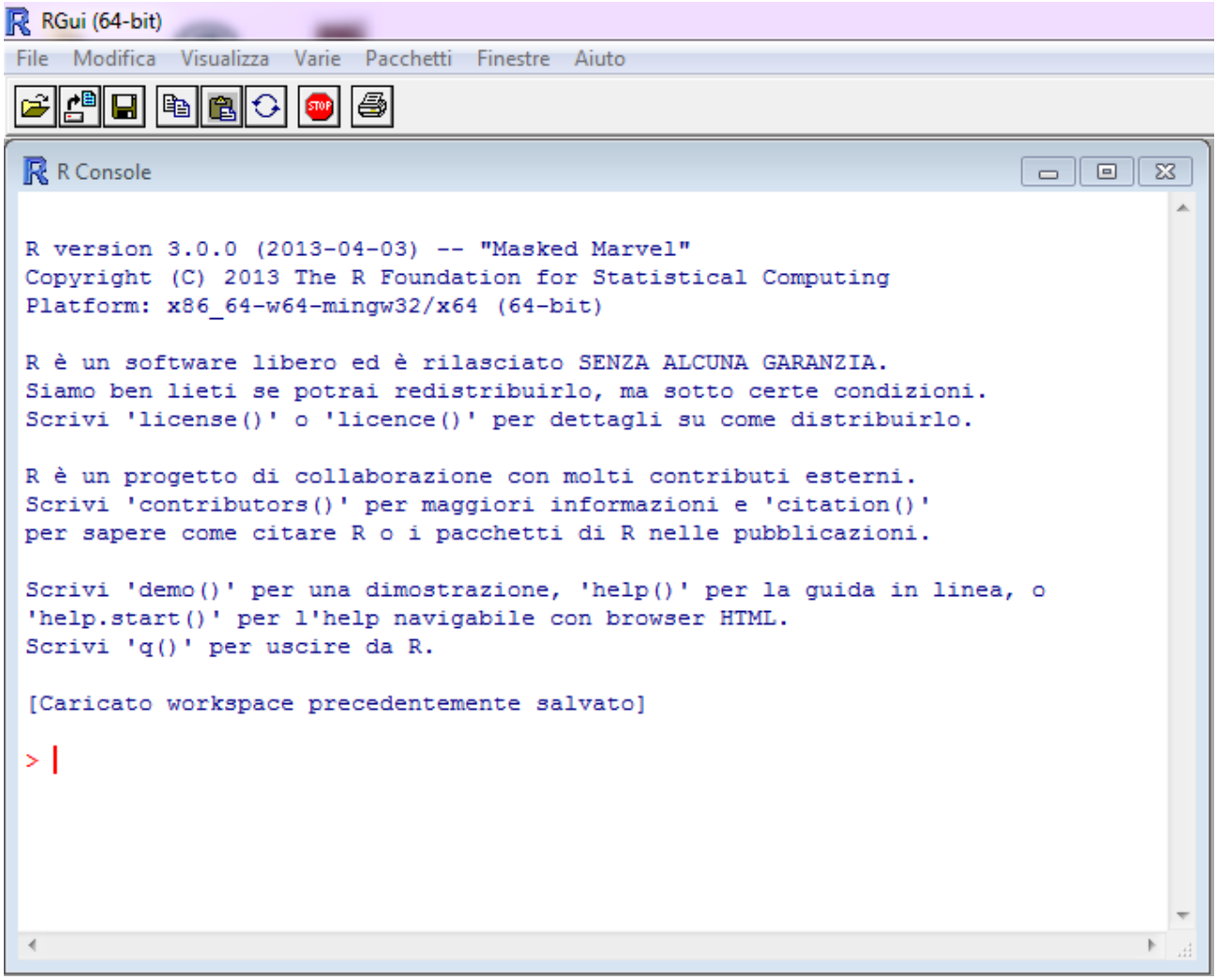


Figura 2.1 Come di presenta la Console di R versione 3.0.0 su una piattaforma Windows a 64 bit.

2.2. Il sistema di help di R

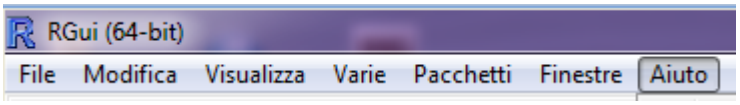
Ecco alcuni comandi che vi consentono di accedere rapidamente alle varie possibilità offerte dal sistema di `help di R`, e che vi suggerisco ovviamente di provare digitando i comandi nella `Console di R`:
→ scrivete **help.start()** e premete `invio` per accedere all'`help di R` navigabile con il browser, il più importante;

---

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 (`Rcmdr`) che avete selezionato. Una volta installato `R Commander` viene richiamato al bisogno sempre con il comando **library(Rcmdr)** digitato il quale vedrete comparire la sua interfaccia grafica dalla quale è possibile accedere alla documentazione di `R Commander` selezionando `Aiuto >> Introduzione a R Commander`. Potete anche accedere per altra documentazione al sito dell'autore John Fox (http://socserv.mcmaster.ca/jfox/Misc/Rcmdr/).

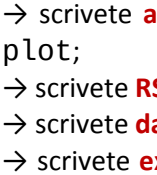
→ scrivete **help(plot)** e premete `invio` per avere un aiuto sulla funzione `plot`;
→ scrivete **?plot** e premete `invio` per avere un aiuto sulla funzione `plot`;
→ scrivete **apropos("plot")** e premete `invio` per la lista di tutte le funzioni che contengono la stringa `plot`;
→ scrivete **RSiteSearch("")** e premete `invio` per effettuare una ricerca aiuto sul sito web di **R**;
→ scrivete **data()** e premete `invio` per avere la lista dei set di dati di esempio;
→ scrivete **example(Theoph)** e premete `invio` poi fate click sulla finestra grafica che si apre sulla destra per avere un esempio della funzione `plot` applicata ai dati della cinetica della teofillina.

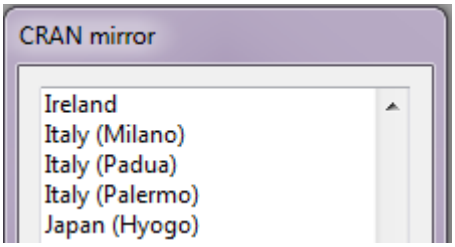
Potete accedere alle funzioni del sistema di `help di R` anche dal menù `Aiuto` della `Console di R`.

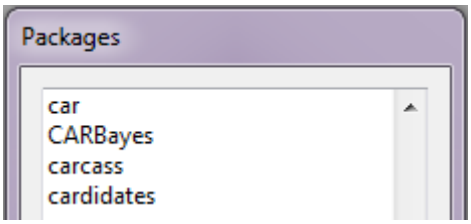


2.3. Pacchetti aggiuntivi rispetto all'installazione base

L'installazione del pacchetto base di **R** include una serie limitata di funzioni statistiche. Se vi collegate al CRAN nella sezione `Software` alla voce `Packages` l'elenco dei pacchetti di statistica aggiuntivi (oltre 4500 a maggio 2013, e in continuo aumento) che potete installare. Sono ciascuno orientato alla risoluzione di un problema specifico. Fate click sul nome del pacchetto che vi potrebbe forse interessare per accedere al suo `Reference manual` che contiene la documentazione necessaria per capire se fa al vostro bisogno. Potete quindi scaricare il pacchetto sul vostro PC dalla `Console di R` selezionando dal menù `Pacchetti` l'opzione `Installa pacchetti...` Comparirà prima una finestra per la selezione del CRAN da cui scaricare il pacchetto, e una volta selezionato il CRAN comparirà la finestra con l'elenco dei pacchetti. Per familiarizzare con questa funzione scaricate la libreria `car` che consente di effettuare alcune interessanti elaborazioni grafiche (la utilizzeremo più avanti). Attenzione: qualcuno dei pacchetti che si desidera installare potrebbe presupporre uno o più altri pacchetti, cui esso si appoggia. In questo caso 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. Data la necessità di disporre, per eseguire i vari esempi forniti nei prossimi capitoli, di alcuni pacchetti che inizialmente non avrete nella vostra installazione base di **R**, è importante seguire gli esempi che faremo avendo sempre il PC collegato con Internet.







2.4. Come strutturare i dati da importare in R

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, come è facile immaginare, ciascuno di noi è portato a continuare a gestirli con lo strumento cui è abituato. Questo nella maggior parte dei casi sarà un foglio elettronico (spreadsheet) in grado di gestire file in formato `.xls`, `.xlsx` e `.csv` (sulla importanza di quest'ultimo formato in **R** torneremo fra poco) e quindi tipicamente `Excel` o meglio ancora

`OpenOffice.org Calc` che ha il vantaggio di essere un programma open-source e gratuito[7].

Una struttura dei dati tipica è questa

id	Sesso	Colesterolo	Trigliceridi	Urea	Creatinina
MT	M	189	164	32	0.6
GF	F	215	188		1.2
MC	F	197	153	26	0.5
SB	M	236	280	22	
FE	F	203	158	48	1.3

nella quale le cose da notare sono assai semplici:
→ le colonne corrispondono alle variabili;
→ le righe corrispondono ai casi;
→ i nomi delle variabili sono riportati nella prima riga, e sono facoltativi (ma ovviamente raccomandati per mantenere ordine e chiarezza nel proprio lavoro);
→ l'identificativo univoco di ciascun caso è riportato nella prima colonna (nella variabile che io per comodità denominerò sempre `id`), ed è facoltativo (può essere utile in casi specifici). Se l'identificativo non viene utilizzato **R** numera automaticamente i casi in ordine screscente;
→ le variabili possono essere sia numeriche, sia qualitative (per esempio, qui, la variabile `Sesso`);
→ è possibile definire una o più variabili in base alla/e quale/i raggruppare i dati (nell'esempio, 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 sono raggruppati sia codificata in modo rigoroso (sesso maschile sempre `M`, e non `M` o `m` a caso, eccetera);
→ è ammessa la mancanza di dati (per esempio qui manca il valore della `Creatinina` del caso SB, e il campo è quindi vuoto), che **R** al momento di importare i dati riconoscerà e classificherà automaticamente riportando la sigla NA (`Not Available`).

Quella riportata è una struttura dati tipica ma non è la sola possibile in **R**. Altre strutture dati che i pacchetti di **R** sono in grado di utilizzare, e in taluni casi che sono specificamente richieste da alcuni pacchetti perché i dati possano essere elaborati, saranno illustrate con gli esempi trattati di volta in volta.

2.5. I dati dimostrativi e gli script

L'impostazione di questo manuale è quella dell'imparare-facendo. Per questo ho predisposto una serie di esempi che includono sia i dati da elaborare sia il codice **R** che li elabora.

I file `nomedelfile.csv` contengono i dati da elaborare. Si tratta di dati in formato `.csv` (`comma separated value`), il formato dati raccomandato per **R**. I file `csv` possono essere generati con Excel e OpenOffice.org Calc semplicemente selezionando il formato `csv` al momento di salvare i dati. Da notare che negli esempi forniti in questo manuale il separatore nei file `csv` è sempre il punto e virgola (`;`). **R** riconosce il punto e virgola come separatore di campo in quanto come vedremo negli script viene specificato il separatore utilizzato nel file di dati `csv` con il parametro **sep=";"**. Cambiando il valore tale parametro è possibile importare dati delimitati per esempio con la virgola (**sep=","**), con uno spazio vuoto (**sep=" "**) o con qualsiasi altro separatore (ovviamente attenzione alla compatibilità tra file realizzati su PC diversi, che potrebbero essere diversamente configurati ed utilizzare differenti separatori di campo).

[7] Potete effettuare il download di OpenOffice dal sito http://it.openoffice.org/

Il codice **R** che elabora i dati viene riportato direttamente nel testo di questo documento, con le seguenti convenzioni:
→ le righe di codice in carattere normale e precedute dal simbolo # rappresentano dei semplici promemoria, dei commenti, e non sono eseguite;
→ le righe di codice **in grassetto e colore** rappresentano il codice **R** che viene eseguito.

Ecco un esempio per iniziare a familiarizzare con dati e script:
→ create la cartella `C:\R\`;
→ salvate in questa cartella il file Boxplot.csv;
→ copiate il codice che segue e incollatelo nella **Console di R.**

```
# INIZIO ESEMPIO
# la seguente riga di codice importa i dati, notare / invece di \ su windows
mydata <- read.table("c:/R/Boxplot.csv", header=TRUE, sep=";")
# la seguente riga traccia i boxplot delle IgA per ciascuna diagnosi
boxplot(IgA~Diagnosi, data=mydata, main="IgA nelle malattie croniche del fegato", xlab="Diagnosi clinica", ylab="IgA in mg/dL", notch=FALSE, col="yellow")
# FINE ESEMPIO
```

Vedete comparire un grafico (**Figura 2.2**) del tipo "box & whiskers plot" (diagramma a scatola e baffi) che illustra la concentrazione delle IgA (in g/L) in un gruppo di soggetti sani (`Controlli`) e la confronta graficamente con quella rilevata in soggetti con cirrosi alcolica (AC), epatite cronica attiva (CAH), epatite cronica persistente (CPH), epatite alcolica non cirrotica (NCAH).

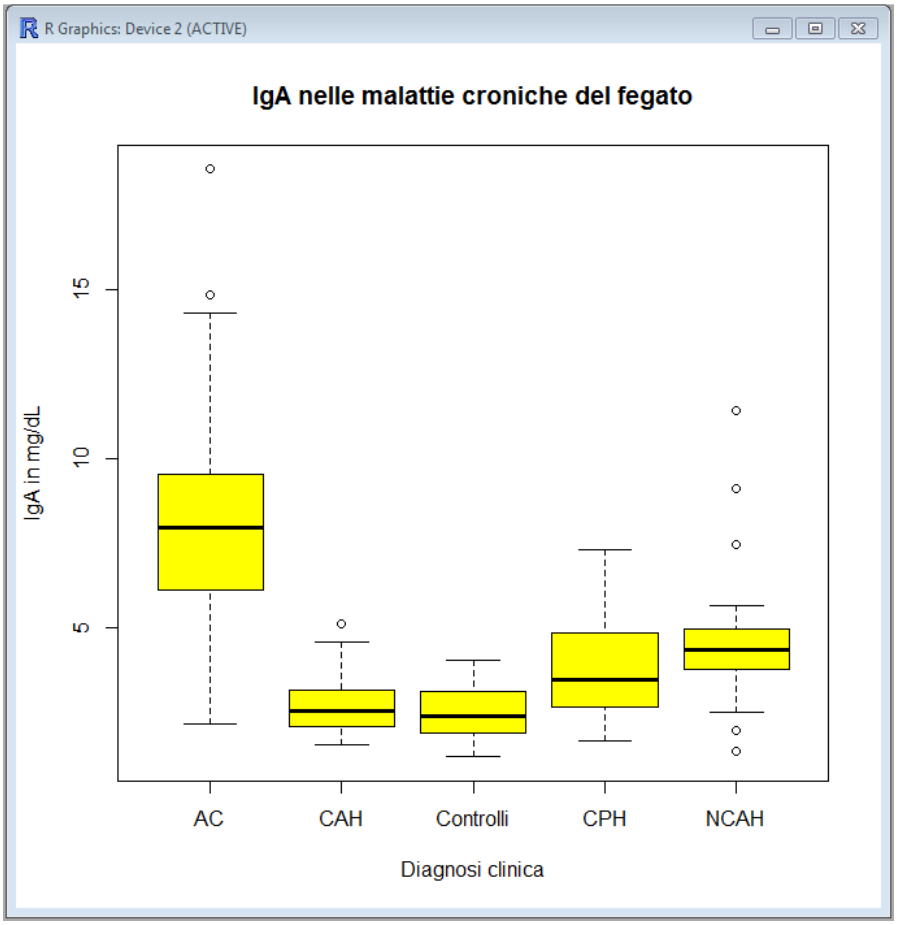


Figura 2.2 Box & whiskers plot delle IgA in alcune malattie croniche del fegato e in un gruppo di controllo.

Da notare che se togliete le righe di commento (che iniziano con #) vedete che tutta la complessa sequenza di istruzioni che porta alla realizzazione del grafico è realizzata con due sole righe di codice. Ecco evidenziato un fatto importante: **R** è un linguaggio conciso ed essenziale.

Cerchiamo ora di capire meglio cosa è accaduto. La prima riga di codice è:
mydata <- read.table("c:/R/Boxplot.csv", header=TRUE, sep=";")

Ci dice che i dati devono essere letti (**read.table()**) dal file **c:/R/Boxplot.csv**, aggiunge che il file ha una intestazione (**header=TRUE**) nella quale si trovano i nomi delle variabili, che il separatore tra campi è un punto e virgola (**sep=";"**) e che il contenuto del file di dati deve essere assegnato (**<-** è l'operatore di assegnazione) ad un oggetto denominato **mydata** (potrebbe essere denominato in qualsiasi altro modo: provate a farlo, ricordandovi di correggere il nome dell'oggetto anche nella seconda riga di codice).

Aprite il file `Boxplot.csv` con Excel o OpenOffice Calc: come vedete nella prima colonna sono riportati i valori della variabile `Diagnosi` e nella seconda colonna sono riportati (per concisione nella figura sono state eliminate le numerose righe di dati intermedi) i valori della variabile `IgA` (**Figura 2.3**).

	A	B
1	Diagnosi	IgA
2	Controlli	1.22
3	.....	
4	Controlli	2.37
5	NCAH	7.44
6	.....	
7	NCAH	3.75
8	CPH	2.45
9	.....	
10	CPH	3.47
11	CAH	2.35
12	.....	
13	CAH	2.93
14	AC	3.51
15	.....	
16	AC	6.22

Figura 2.3 Come appare un file in formato csv aperto con Excel o con OpenOffice Calc.

Se invece aprite il file `Boxplot.csv` con un editor di testo come il `Blocco note di Windows` (**Figura 2.4**) vedete i dati nel formato in cui sono stati salvati sul disco:

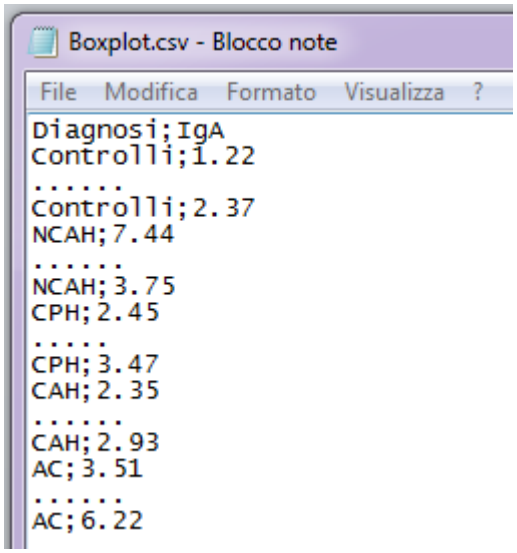


Figura 2.4 Come appare un file in formto csv aperto con un editor di testo come il Blocco note di Windows.

Nella prima riga sono stati salvati i nomi delle due variabili contenute nel file, rispettivamente la diagnosi (`Diagnosi`) e la concentrazione delle IgA in mg/dL (`IgA`), che sono in due campi (colonne) diversi separati da un punto e virgola (`;`). Nelle righe successive sono stati salvati i valori delle due variabili per ciascuno dei casi osservati, sempre separati dal punto e virgola (separatore di campo).

La seconda riga di codice è:
boxplot(IgA~Diagnosi, data=mydata, main="IgA nelle malattie croniche del fegato", xlab="Diagnosi clinica", ylab="IgA in mg/dL", notch=FALSE, col="yellow")

Ci dice di tracciare un diagramma a scatola e baffi (**boxplot()**), traendo i dati dall'oggetto `mydata` (**data=mydata**), aggregando i dati per diagnosi (**IgA~Diagnosi**), aggregando cioè i valori della variabile `IgA` per ciascun valore della variabile `Diagnosi`. Aggiunge il titolo del diagramma (**main="IgA nelle malattie croniche del fegato"**), l'etichetta dell'asse delle x (**xlab="Diagnosi clinica"**), l'etichetta dell'asse delle y (**ylab="IgA in mg/dL"**), specifica che vogliamo la scatola di colore giallo (**col="yellow"**), e che le scatole non devono avere l'incisura (**notch=FALSE**, i dettagli di questo tipo di rappresentazione grafica saranno discussi successivamente in uno specifico esempio).

Torniamo ora al codice **R** dell'esempio con l'obiettivo di salvarlo per poterlo riutilizzare in un secondo tempo, modificandolo per adattarlo a nuove esigenze. Per questo dovete semplicemente selezionare e copiare il codice **R** per incollarlo in un editor di file di testo. In Windows è possibile utilizzare il `Blocco note di Windows` che trovate in `Programmi >> Accessori`. Dopo avere copiato il codice salvate il file e denominatelo `Boxplot.txt` (**Figura 2.5**).

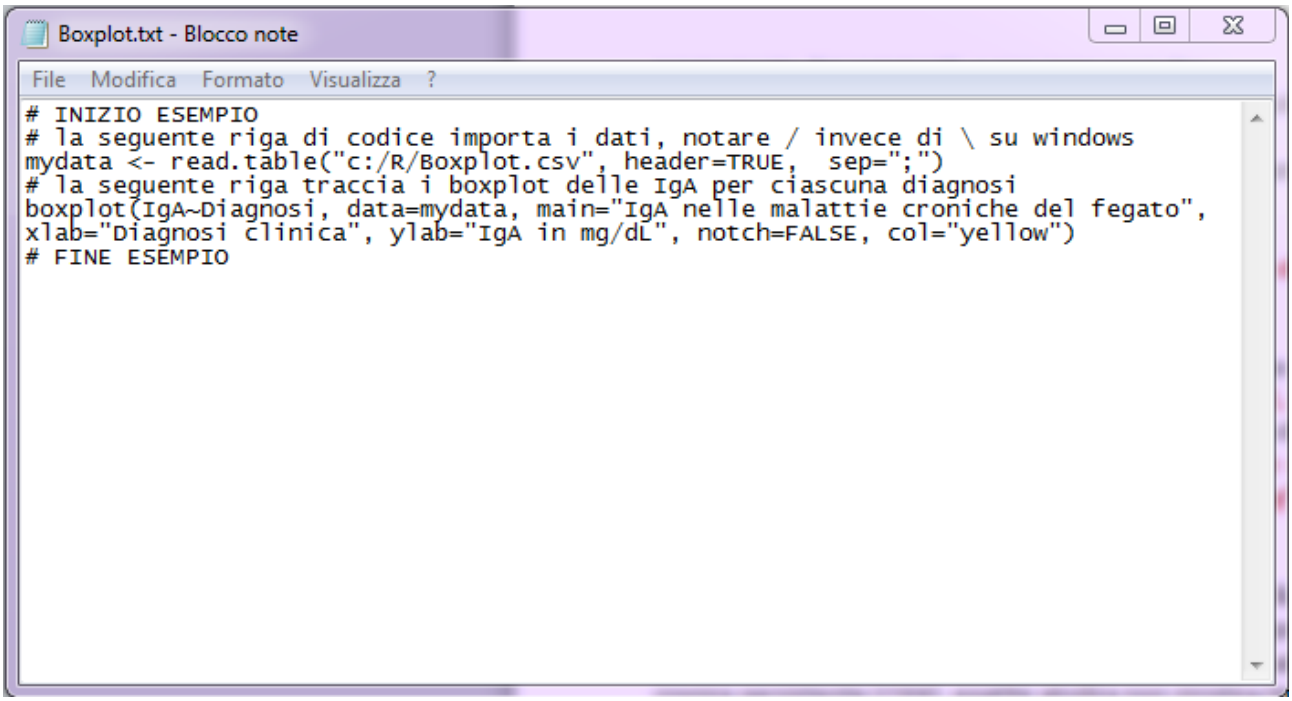


Figura 2.5 Il codice **R** per generare i box & whiskers plot, copiato nel Blocco note di Windows.

A questo punto avrete due file `Boxplot`. Il primo è il file `Boxplot.csv` che vi ho fornito e che contiene i dati da elaborare, il secondo è il file `Boxplot.txt` che avete salvato e che contiene il codice R per tracciare i `box & whiskers plot`.

In alternativa al `Blocco note di Windows` per salvare i codice **R** potete utilizzare l'`Editor di R`,

che potete aprire dalla Console di R selezionando File >> Nuovo script. Incollate il codice copiato nell'Editor di R e salvatelo in un file. Ricordate che dalla console di **R** i file sono salvati di default con l'estensione .R e con il Blocco note di Windows i file sono salvati invece con l'estensione .txt: ma di fatto si tratta di file di testo assolutamente identici. Per salvare il codice **R** potete scegliere di utilizzare il Blocco note di Windows oppure l'Editor di R a vostro piacimento.

Un file di dati (.csv) e un file (.txt o .R) con il codice **R** per elaborarlo sono tutto quanto serve per lavorare con **R**.

2.6. Come importare in R i dati di un file .csv

È necessario ricordare sempre che:
→ il formato .csv è il formato raccomandato per importare i dati in **R**;
→ Excel e OpenOffice.org Calc salvano in formato .xls o .xlsx (come formato di default);
→ è possibile salvare i dati in formato .csv anche con Excel e OpenOffice.org Calc salvando il file con l'opzione Salva con nome... e scegliendo come Tipo di File il formato .csv.

Ecco un esercizio per migliorare la capacità di importare in **R** i dati di un file .csv:
→ create la cartella C:\R\;
→ salvate in questa cartella i file InputCSVconid.csv e InputCSVsenzaid.csv.

Entrambi i file .csv contengono gli stessi dati relativi alla composizione in calcio, fosfato, ossalato e magnesio di 10 calcoli delle vie urinarie. Aprite il file InputCSVcoinid.csv utilizzando il Blocco note di Windows, per vedere come sono organizzati i dati in un tipico file .csv (comma separated values) nel quale la prima variabile (id) contiene l'identificativo univoco del caso mentre le successive quattro contengono la quantità di Calcio, Fosfato, Ossalato Magnesio:

```
id;Calcio;Fosfato;Ossalato;Magnesio
C1;99;81;69;61
C2;78;65;53;43
C3;81;66;38;54
C4;45;23;19;16
C5;44;18;24;19
C6;102;83;72;66
C7;83;68;49;45
C8;74;71;41;57
C9;38;19;22;14
C10;48;14;21;12
```

Copiate e incollate nella Console di **R** questa riga di codice:
mydata <- read.table("c:/R/InputCSVconid.csv", header=TRUE, sep=";", row.names="id")
con la quale è possibile importare in **R** i dati specificando che gli identificativi univoci di ciascun caso sono contenuti nella colonna id (**row.names="id"**). Il codice riportato sopra può essere interamente riutilizzato per le vostre specifiche esigenze, ricordando che dovete semplicemente:
→ sostituire il nome del file **"c:/R/InputCSVconid.csv"** con quello del vostro file;
→ controllare il separatore di campo usato ed eventualmente correggere opportunamente il punto e virgola in **sep=";"**.

Adesso nella Console di R scrivete
mydata
e premete invio e vedrete il contenuto dell'oggetto **mydata**, rappresentato appunto dai dati che avete

appena importato:

```
    Calcio Fosfato Ossalato Magnesio
C1      99      81       69       61
C2      78      65       53       43
C3      81      66       38       54
C4      45      23       19       16
C5      44      18       24       19
C6     102      83       72       66
C7      83      68       49       45
C8      74      71       41       57
C9      38      19       22       14
C10     48      14       21       12
```

Se controllate ora il secondo esempio, trovate che nel file `InputCSVsenzaid.csv` i dati sono i medesimi, tranne che per il fatto che manca il campo/variabile id:

```
Calcio;Fosfato;Ossalato;Magnesio
99;81;69;61
78;65;53;43
81;66;38;54
45;23;19;16
44;18;24;19
102;83;72;66
83;68;49;45
74;71;41;57
38;19;22;14
48;14;21;12
```

Copiate e incollate nella `Console di R` questa riga di codice:
mydata <- read.table("c:/R/InputCSVsenzaid.csv", header=TRUE, sep=";")
nella quale il solo dato evidente e significativo (a parte il diverso nome del file) è che ora è scomparso il parametro **row.names="id"**.

Adesso nella `Console di R` scrivete
mydata
e premete `invio` e vedrete il contenuto dell'oggetto **mydata**, rappresentato appunto dai dati che avete appena importato:

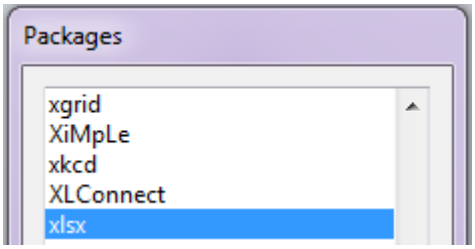
```
   Calcio Fosfato Ossalato Magnesio
1      99      81       69       61
2      78      65       53       43
3      81      66       38       54
4      45      23       19       16
5      44      18       24       19
6     102      83       72       66
7      83      68       49       45
8      74      71       41       57
9      38      19       22       14
10     48      14       21       12
```

Come potete notare nel caso in cui manchi l'identificativo della riga/caso, **R** provvede ad assegnarlo automaticamente, sotto forma di un numero progressivo (1 per la prima riga, 2 per la seconda riga, eccetera).

2.7. Come importare in R i dati di un file .xls o .xlsx

Se il formato `.csv` è il formato raccomandato per importare i dati in **R** la ragione è semplice: si tratta di un formato standard (fissato una volta per tutte e universalmente riconosciuto). Mentre i formati `.xls` e `.xlsx` non lo sono, e la struttura dei file potrebbe cambiare nelle nuove versioni di Excel e di OpenOffice Calc senza alcun preavviso, causando errori imprevedibili nell'importazione dai dati in **R**.

Nonostante questo in **R** trovate alcune librerie che consentono di importare i dati direttamente da file `.xls` e `.xlsx`. Qui illustro la libreria `xlsx` che potete scaricate dal CRAN selezionando nel menù `Pacchetti` di `R` l'opzione `Installa pacchetti`... e quindi selezionando `xlsx` dall'elenco dei pacchetti (`Packages`) disponibili.



Ora:
→ create la cartella `C:\R\`;
→ salvate in questa cartella i file InputXLS.xls e InputXLSX.xlsx.

Quindi copiate e incollate nella `Console di R` questo codice:

```
#
require(xlsx)
mydata <- read.xlsx("C:/R/InputXLS.xls", sheetName="Conidriga")
#
```

dove gli unici argomenti richiesti dalla funzione **read.xlsx** sono il nome del file con il percorso completo (**C:/R/InputXLS.xls**) e il nome del foglio (**"Conidriga"**) all'interno del file. Se ora digitate **mydata** e premete `invio` potete scorrere nella `Console di R` i dati. Per avere la conferma del fatto che sono stati importati correttamente confrontateli con l'originale aprendo con Excel o OpenOffice Calc il foglio `Conidriga` del file `InputXLS.xls`.

Ripetete il tutto con il seguente codice:

```
#
require(xlsx)
mydata <- read.xlsx("C:/R/InputXLSX.xlsx", sheetName="Senzaidriga")
#
```

che mostra come la libreria possa essere applicata anche al più recente formato di file `.xlsx`. Anche in questo caso se digitate **mydata** e premete `invio` potete scorrere nella `Console di R` i dati. Per avere la conferma del fatto che sono stati importati correttamente confrontateli con l'originale aprendo con Excel o OpenOffice Calc il foglio `Senzaidriga` del file `InputXLSX.xlsx`.

Per le molte altre cose che è possibile fare con la libreria `xlsx` si rimanda alla documentazione della libreria xlsx che trovate sul CRAN anche digitando semplicemente "`package xlsx`" nella casella di ricerca di Google.

2.8. Gestione dei file con dati mancanti

→ create la cartella `C:\R\`;
→ salvate in questa cartella il file InputNA.csv.

Innanzitutto aprite il file `InputNA.csv` utilizzando il `Blocco note di Windows`. Contiene la concentrazione delle `IgA` (in g/L) in un gruppo di soggetti sani (`Controlli`) e in soggetti con cirrosi

alcolica (AC), epatite cronica attiva (CAH), epatite cronica persistente (CPH), epatite alcolica non cirrotica (NCAH) organizzati in cinque colonne. Come vedete (si tratta degli stessi del paragrafo 2.5 ma organizzati in modo diverso) le cinque classi di pazienti contengono ciascuna un numero differente di casi:

```
Normali;NCAH;CPH;CAH;AC
1.22;7.44;2.45;2.35;3.51
2.81;4.58;1.63;3.21;4.23
4.02;3.71;3.44;3.88;7.66
2.23;4.94;2.47;1.56;9.54
2.35;3.49;1.95;1.78;11.35
1.64;3.88;4.56;2.49;6.43
2.08;4.71;7.31;3.11;5.28
1.96;4.32;5.78;4.56;2.14
1.54;4.9;3.4;5.11;4.76
1.63;11.43;5.12;2.36;7.91
3.25;4.63;6.88;2.98;9.33
2.9;4.11;3.21;2.53;18.57
3.44;5.03;3.64;1.77;8.81
2.55;9.12;2.8;1.51;14.31
1.18;1.32;3.47;2.93;10.83
1.78;4.33;;;8.48
2.56;5.66;;;9.56
1.36;4.08;;;9.01
1.83;2.48;;;12.44
2.4;1.95;;;7.61
2.61;3.75;;;7.03
3.02;;;;8.8
3.18;;;;6.12
2.97;;;;5.22
1.99;;;;7.99
2.57;;;;6.11
2.13;;;;5.88
3.56;;;;12.3
3.76;;;;14.83
2.28;;;;9.07
1.24;;;;6.83
1.88;;;;6.22
2.76;;;;
1.88;;;;
3.12;;;;
3.54;;;;
3.88;;;;
3.02;;;;
1.18;;;;
2.66;;;;
2.28;;;;
1.33;;;;
1.99;;;;
3.15;;;;
3.18;;;;
4.03;;;;
1.16;;;;
1.96;;;;
3.44;;;;
2.37;;;;
```

Copiate e incollate nella Console di R questo codice:

```
#
mydata <- read.table("c:/R/InputNA.csv", header=TRUE, sep=";")
mydata
x <- mydata[c("Normali")]
colMeans(x)
x <- mydata[c("NCAH")]
colMeans(x)
x <- mydata[c("NCAH")]
colMeans(x, na.rm=TRUE)
#
```

Ora nella utilizzate i tasti Pag-su e Pag-giù per scorrere nella finestra della Console di R il codice eseguito:

→ innanzitutto sono importati i dati

```
mydata <- read.table("c:/R/InputNA.csv", header=TRUE,  sep=";")
```

→ viene visualizzato l'oggetto **mydata** che contiene i dati importati e si vede che **R** ha sostituito automaticamente i dati mancanti con la sigla NA (che sta per Not Available)

```
mydata
```

→ la media nella colonna Normali, nella quale non vi sono dati mancanti, viene calcolata

```
x <- mydata[c("Normali")]
colMeans(x)
Normali
  2.457
```

→ la media della colonna NCAH a causa dei dati mancanti non può essere calcolata, e viene restituito NA

```
x <- mydata[c("NCAH")]
colMeans(x)
NCAH
  NA
```

→ con il parametro **na.rm=TRUE** che rimuove i dati mancanti la media della colonna NCAH viene invece calcolata

```
x <- mydata[c("NCAH")]
colMeans(x, na.rm=TRUE)
  NCAH
4.755238
```

Vedremo nel paragrafo dedicato alle statistiche elementari come con la funzione **na.omit()** sia possibile eliminare definitivamente da una tabella i casi con dati mancanti.

2.9. Inserimento dei dati dalla Console di R

Se normalmente i dati sono importati dall'esterno, in alcuni casi potrebbe essere utile gestirli direttamente dalla Console di R. Per questo ho predisposto tre esempi, che illustrano la sintassi da utilizzare per inserire direttamente da tastiera vettori (array) numerici e non, e combinarli in tabelle (dataset) assegnando i nomi alle variabili. Sperimentateli per ora a scopo didattico (ma prima o poi vi verranno utili). Dopo avere eseguito ogni esempio utilizzate casi i tasti Pag-su e Pag-giù per scorrere nella finestra della Console di R quanto è accaduto.

Il primo esempio genera due vettori, li combina in una matrice, assegna i nomi alle variabili (colonne) e assegna un descrittore ai casi (righe).

```
# INIZIO PRIMO ESEMPIO ********************************
# PRIMO VETTORE
# genera gli interi da 1 a 10
x <- 1:10
# SECONDO VETTORE
# genera dieci valori di deviata normale standardizzata z
y <- rnorm(10)
# COMBINA I DUE VETTORI IN UNA MATRICE
# combina x e y nell'oggetto mymatrix
mymatrix <- data.frame(x,y)
# mostra l'oggetto mymatrix
mymatrix
# assegna i nomi alle variabili/colonne
names(mymatrix) <- c("Progressivo", "Deviata normale standardizzata z")
# mostra mymatrix
mymatrix
# assegna un descrittore ai casi/righe
row.names(mymatrix) <- c("Uno","Due","Tre","Quattro","Cinque","Sei","Sette","Otto","Nove","Dieci")
# mostra mymatrix
mymatrix
# FINE PRIMO ESEMPIO ********************************
```

Il secondo esempio genera un matrice 2x2 (due righe per due colonne) e assegna i nomi alle righe e i nomi alle colonne.

```
# INIZIO SECONDO ESEMPIO ********************************
# GENERA UNA MATRICE 2x2
# array con i valori (numerici) contenuti nelle celle
cells <- c(1,26,24,68)
# array con i nomi delle righe
rnames <- c("R1", "R2")
# array con i nomi delle colonne
cnames <- c("C1", "C2")
# costruisce la matrice con i valori numerici e i nomi definiti sopra
mymatrix <- matrix(cells, nrow=2, ncol=2, byrow=TRUE, dimnames=list(rnames, cnames))
# mostra mymatrix
mymatrix
# FINE SECONDO ESEMPIO ********************************
```

Il terzo esempio genera una tabella (dataset) che contiene valori numerici, alfanumerici e logici, e assegna i nomi alle variabili (colonne).

```
# INIZIO TERZO ESEMPIO ********************************
# GENERA UN DATAFRAME (sinonimo di DATASET in SPSS e in SAS)
# può contenere variabili sia numeriche, sia alfanumeriche, sia logiche
# array con valori numerici
d <- c(9901,9902,9903,9904)
# array con valori alfanumerici
e <- c("rosso", "bianco", "blu", NA)
# array con valori logici
f <- c(TRUE,TRUE,TRUE,FALSE)
# genera il dataframe/dataset
mydataset <- data.frame(d,e,f)
```

```
# assegna i nomi alle variabili/colonne
names(mydataset) <- c("Identificativo","Colore","Dato valido")
# mostra mydataset
mydataset
# FINE TERZO ESEMPIO ********************************
```

2.10. Salvare una sessione di R in un file

→ create la cartella `C:\R\`;
→ salvate in questa cartella il file OutputDati.csv;

Quindi copiate e incollate nella `Console di R` questo codice:

```
#
mydata <- read.table("c:/R/OutputDati.csv", header=TRUE,  sep=";", row.names="id")
sink("c:/R/OutputFile.txt", append=FALSE, split=FALSE)
mydata
sink()
#
```

Il codice esegue alcune cose molto semplici:
→ sono importati i dati

```
mydata <- read.table("c:/R/OutputDati.csv", header=TRUE,  sep=";", row.names="id")
```

→ l'output viene ridiretto dalla `Console di R` a un file, di cui sono specificati nome e percorso (`c:/R/OutputFile.txt`)

```
sink("c:/R/OutputFile.txt", append=FALSE, split=FALSE)
```

→ viene visualizzato l'oggetto **mydata** che contiene i dati importati scrivendo i dati su file invece di inviarli alla Console di R, come dimostrato dal fatto che nella cartella `C:\R\` vi comparirà il file `OutputFile.txt` nel quale trovate appunto i dati dell'oggetto **mydata**

```
mydata
```

→ viene ripristinato l'output alla `Console di R`

```
sink()
```

2.11. Salvare i grafici di R in un file

→ create la cartella `C:\R\`;
→ salvate il file OutputGrafici.csv.

Eseguite il codice seguente:

```
# INIZIO ESEMPIO ********************************
# la prima riga contiene i nomi delle variabili, il separatore è parametrizzabile
# la prima colonna contiene la Diagnosi
# la seconda colonna contiene la concentrazione delle IgA in mg/dL
# notare / invece di \ su windows
mydata <- read.table("c:/R/OutputGrafici.csv", header=TRUE, sep=";")
# genera un file bmp (Windows bitmap)
bmp("c:/R/ Filebmp.bmp")
boxplot(IgA~Diagnosi, data=mydata, main="IgA nelle malattie croniche del fegato",
  xlab="Diagnosi clinica", ylab="IgA in mg/dL", notch=FALSE, col="yellow")
```

```
dev.off()
# genera un file jpeg (Joint Photographic Experts Group)
jpeg("c:/R/ Filejpeg.jpg")
boxplot(IgA~Diagnosi, data=mydata, main="IgA nelle malattie croniche del fegato",
  xlab="Diagnosi clinica", ylab="IgA in mg/dL", notch=FALSE, col="yellow")
dev.off()
# genera un file pdf (Portable Document Format)
pdf("c:/R/ Filepdf.pdf")
boxplot(IgA~Diagnosi, data=mydata, main="IgA nelle malattie croniche del fegato",
  xlab="Diagnosi clinica", ylab="IgA in mg/dL", notch=FALSE, col="yellow")
dev.off()
# genera un file png (Portable Network Graphics)
png("c:/R/ Filepng.png")
boxplot(IgA~Diagnosi, data=mydata, main="IgA nelle malattie croniche del fegato",
  xlab="Diagnosi clinica", ylab="IgA in mg/dL", notch=FALSE, col="yellow")
dev.off()
# genera un file ps (postscript)
postscript("c:/R/ Fileps.ps")
boxplot(IgA~Diagnosi, data=mydata, main="IgA nelle malattie croniche del fegato",
  xlab="Diagnosi clinica", ylab="IgA in mg/dL", notch=FALSE, col="yellow")
dev.off()
# genera un file wmf (Windows metafile)
win.metafile("c:/R/ Filewmf.wmf")
boxplot(IgA~Diagnosi, data=mydata, main="IgA nelle malattie croniche del fegato",
  xlab="Diagnosi clinica", ylab="IgA in mg/dL", notch=FALSE, col="yellow")
dev.off()
# FINE ESEMPIO ********************************
```

Controllate infine che nella cartella `C:\R\` compaiano i sei file generati dallo script. I grafici hanno qualità diversa, e questo può essere utile in relazione al diverso impiego che di essi si può fare.

2.12. L'interfaccia grafica (GUI) di R

Nonostante in questo manuale si faccia riferimento esclusivamente alla `Console di R`, non va dimenticato che **R** dispone anche di una interfaccia grafica (GUI, Graphic User Interface) che consente di utilizzare il linguaggio senza conoscerlo, semplicemente selezionando i comandi da una serie di menù. E che consente quindi all'utente di interagire con **R** con una interfaccia "evoluta", simile a quella degli altri programmi di statistica in ambiente Windows.

La GUI ufficiale di **R** è `R Commander`. Per installarla è necessario essere collegati a Internet, accedere ad uno dei CRAN e scaricare la libreria `Rcmdr`. Dato che questa libreria a sua volta si appoggia ad altre librerie, insieme a questa viene 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 della libreria `Rcmdr`.

Una volta installato `R Commander` viene richiamato al bisogno sempre con il comando

```
library(Rcmdr)
```

digitato il quale compare la sua interfaccia grafica (**Figura 2.6**). Per tutto quanto può essere utile si rimanda alla pagina di John Fox, l'autore del programma[8].

[8] http://socserv.mcmaster.ca/jfox/misc/rcmdr/

Figura 2.6 La finestra di R Commander, l'interfaccia grafica (GUI) di **R**.

Si rammenta una cosa molto importante riportata nella documentazione. Per funzionare propriamente sotto Windows, R `Commander` necessita della Single Document Interface (SDI). Per questo è necessario (ma solamente la prima volta) configurare l'avvio di **R** come segue:
→ fare click con il tasto destro del mouse sull'icona di **R** che avete sul desktop;
→ scegliere `Proprietà`;
→ selezionare la scheda `Collegamento`;
→ nel campo `Destinazione` dopo **Rgui.exe** aggiungere **--sdi** (aggiungere prima uno spazio quindi **--sdi**, in modo che la `Destinazione` risulti indicata come **Rgui.exe --sdi**).

2.13. Utilizzare i dati e gli script con MacOS X

Per i possessori di Mac utilizzare i dati e gli script qui riportati è fortunatamente molto semplice, basta ricordare due cose:
→ quando scaricate i file di dati, MacOS X li salva automaticamente nella cartella `Download`;
→ se nel codice che importa i dati in questo documento trovate scritto (ad esempio) **mydata <- read.table("c:/R/Statcorr.csv", header=TRUE, sep=";")**, per il Mac dovete correggete il codice in **mydata <- read.table("Statcorr.csv", header=TRUE, sep=";")** eliminando **c:/R/** ovvero, in altre parole, lasciate all'interno delle virgolette solamente il nome del file.

Fatta questa modifica, è possibile eseguire il codice esattamente come riportato negli esempi.

3. R funzioni statistiche

Le funzioni statistiche di **R** sono trattate assumendo che abbiate familiarizzato adeguatamente con tutti gli argomenti generali contenuti nel capitolo 2. Le indicazioni qui riportate fanno riferimento alla versione di **R** per Windows. Per MacOS X vedere sempre nel capitolo 2 come utilizzare i dati e gli script con MacOS X.

Potete imparare a utilizzare le funzioni statistiche di R eseguendo il codice che trovate nelle pagine seguenti con i dati di esempio forniti come file `.csv` generati con Excel e OpenOffice.org Calc che devono essere scaricati e installati sul PC nella cartella `C:\R\`. Se li installate in una cartella diversa, dovete modificare opportunamente il codice **read.table("c:/R/** specificando il nuovo percorso nel quale avete installato i file. Per eseguire il codice **R** dovete semplicemente selezionarlo nella pagina web includendo il cancelletto # che chiude ogni blocco di codice, copiarlo, e incollarlo nella `Console di R`.

3.1. Test chi-quadrato (χ^2)

Nel caso delle scale nominali e delle scale ordinali esiste un solo modo per esprimere le osservazioni in modo quantitativo (numerico): contare gli eventi. Per verificare se un evento si verifica in due o più gruppi/categorie con la stessa frequenza, o con la frequenza prevista da un modello teorico, sono impiegati il test chi-quadrato o una delle sue varianti, il test di Fisher o il test di McNemar.

3.1.1. Test chi-quadrato quando sono note le frequenze teoriche (1 riga per n colonne)

La frequenza attesa di nuovi nati di sesso maschile e di sesso femminile è pari a 0.5 per entrambi i sessi. Tra i cariotipi eseguiti su liquido amniotico per diagnosi prenatale nell'arco di due mesi se ne sono osservati 487 di tipo maschile (XY) e 503 di tipo femmine (XX). Il numero di casi osservati è in linea con la frequenza attesa?

Copiate e incollate nella `Console di R` ed eseguite questo codice:

```
#
casi.osservati <- c(487,503) # sono immessi i casi osservati
freq.attese <- c(0.5,0.5) # sono immesse le frequenze attese
# il chi-quadrato viene calcolato e salvato nell'oggetto Chiquad
Chiquad <- chisq.test(casi.osservati, p = freq.attese)
Chiquad$observed # mostra le frequenze osservate
Chiquad$expected # mostra le frequenze attese
Chiquad #  mostra i risultati del test chi-quadrato
#
```

Nella `Console di R` sono mostrate le frequenze osservate (**Chiquad$observed**)

```
[1] 487 503
```

quindi sono mostrate le frequenze attese (**Chiquad$expected**)

```
[1] 495 495
```

e infine viene mostrato il risultato del test chi-quadrato (**Chiquad**):

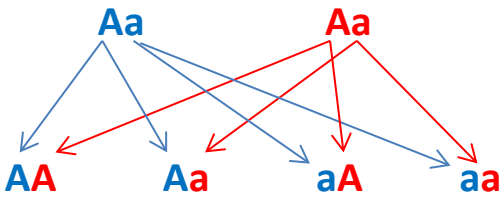
```
        Chi-squared test for given probabilities

data:  casi.osservati
X-squared = 0.2586, df = 1, p-value = 0.6111
```

In statistica ci si assume il rischio di considerare una differenza "significativa" quando la probabilità di osservarla per caso è bassa, in genere meno del 5% (`p < 0.05`). Il valore di `p`, che indica la probabilità di

osservare per caso una differenza quale quella effettivamente osservata (487 maschi e 503 femmine contro una frequenza attesa di 495 e 495 rispettivamente), è 0.6111: pertanto dobbiamo ritenere che la differenza osservata sia presumibilmente legata al caso, ovvero statisticamente "non significativa".

Per l'eredità di un carattere autosomico recessivo presente in entrambi i genitori (padre **Aa** e madre **Aa**) le quattro possibili combinazioni sono



e hanno frequenza pari a 0.25 (**AA**), 0.50 (**Aa / Aa**), 0.25 (**aa**). Da genitori entrambi **Aa** sono nati 85 figli **AA**, 173 **Aa** e 94 **aa**. Il numero di casi osservati è in linea con la frequenza attesa?

Copiate e incollate nella `Console di R` ed eseguite questo codice:

```
#
casi.osservati <- c(85,173,94) # sono immessi i casi osservati
freq.attese <- c(0.25,0.50,0.25) # sono immesse le frequenze attese
# il chi-quadrato viene calcolato e salvato nell'oggetto Chiquad
Chiquad <- chisq.test(casi.osservati, p = freq.attese)
Chiquad$observed # mostra le frequenze osservate
Chiquad$expected # mostra le frequenze attese
Chiquad # mostra i risultati del test chi-quadrato
#
```

Dopo avere mostrato le tre frequenze osservate (**Chiquad$observed**)

```
[1]  85 173  94
```

e le tre frequenze attese (**Chiquad$expected**)

```
[1]  88 176  88
```

nella `Console di R` viene mostrato il risultato del test chi-quadrato (**Chiquad**) precedentemente calcolato:

```
        Chi-squared test for given probabilities

data:  casi.osservati
X-squared = 0.5625, df = 2, p-value = 0.7548
```

Il valore di p, che indica la probabilità di osservare per caso una differenza quale quella effettivamente osservata (85 omozigoti **AA** osservati contro 88 attesi, 173 eterozigoti **Aa** osservati contro 176 attesi, 94 omozigoti **aa** osservati contro 88 attesi), in questo caso è 0.7548. Anche questa volta la probabilità di osservare per caso una differenza quale quella effettivamente osservata è molto elevata, al punto che dobbiamo ritenere la differenza osservata presumibilmente legata al caso, ovvero statisticamente "non significativa".

Il codice **R** che segue riprende gli stessi identici dati con un codice semplificato, calcolando però questa volta il valore di p sia nel modo tradizionale, con la distribuzione teorica di chi-quadrato, sia con il metodo Monte Carlo. Ho preparato una breve illustrazione del concetto che sta alla base del metodo Monte Carlo[9] in quanto lo troverete poi applicato in varie altre situazioni. Copiate e incollate nella `Console di R` questo codice ed eseguitelo:

[9] http://www.bayes.it/pdf/MetodoMonteCarlo.pdf

```
#
casi.osservati <- c(85,173,94) # sono immessi i casi osservati
freq.attese <- c(0.25,0.50,0.25) # sono immesse le frequenze attese
# chi-quadrato, p calcolato dalla distribuzione teorica di chi-quadrato
chisq.test(casi.osservati, p = freq.attese)
# chi-quadrato, p calcolato mediante una simulazione Monte Carlo con 10000 replicati
chisq.test(casi.osservati, p = freq.attese, simulate.p.value = TRUE, B = 10000)
#
```

Dopo avere immesso i dati (prime due righe) viene calcolato e mostrato nella Console di R il risultato del test chi-quadrato (**chisq.test**) con il valore di p calcolato in base alla distribuzione di probabilità teorica chi-quadrato:

```
        Chi-squared test for given probabilities

data:  casi.osservati
X-squared = 0.5625, df = 2, p-value = 0.7548
```

Nella successiva e ultima riga di codice viene calcolato e mostrato nella Console di R il risultato del test chi-quadrato (**chisq.test**) con il valore di p calcolato mediante una simulazione Monte Carlo:

```
        Chi-squared test for given probabilities with simulated p-value
(based on 10000 replicates)

data:  casi.osservati
X-squared = 0.5625, df = NA, p-value = 0.7698
```

Come potete constatare le conclusioni con i due approcci sono quasi identiche.

3.1.2. Test per una tabella di 2 righe x 2 colonne: test chi-quadrato, test di Fisher, test di McNemar

Il principio che vale per le tabelle nelle quali le osservazioni sono organizzate in 2 righe e 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:
→ si utilizza il test di McNemar quando le osservazioni non sono indipendenti (dati appaiati).

Il **test chi-quadrato** viene applicato ai dati relativi alla presenza o assenza di emolisi, rispetto ad un valore soglia prefissato, utilizzando due sistemi di prelievo (A e B), in situazioni di estrema difficoltà del prelievo venoso. Impiegando il sistema A si sono osservati 68 casi di emolisi su 109 prelievi (pari al 62.4%). Impiegando il sistema B si sono osservati 93 casi di emolisi su 125 prelievi (pari al 74.4%).

Ci si chiede se il numero di casi di emolisi osservati con l'uno e l'altro sistema di prelievo sia significativamente diverso. Le osservazioni riguardano in totale 234 differenti prelievi e altrettanti campioni di sangue, sono numerose e sono indipendenti.

I dati sono stati raccolti in questa tabella:

Prelievo	Emolisi_no	Emolisi_si
Sistema A	41	68
Sistema B	32	93

Scaricate e salvate nella cartella C:\R\ il file Chiquad_2x2.csv. Quindi copiate e incollate nella Console di R ed eseguite questo codice:

```
# con la prima riga sono importati i dati
mydata <- read.table("c:/R/Chiquad_2x2.csv", header=TRUE, sep=";", row.names="Prelievo")
# i dati sono mostrati nella Console di R
mydata
# test chi quadrato con la correzione di Yates
chisq.test(mydata, correct=TRUE)
# test chi quadrato senza la correzione di Yates
chisq.test(mydata, correct=FALSE)
#
```

Il test chi-quadrato con 1 grado di libertà è esatto solo asintoticamente per dimensioni molto grandi dei campioni pertanto nell'esempio riportato sopra viene applicata la correzione di Yates per la continuità (**correct = TRUE**) ottenendo `p = 0.06615` (differenza non significativa assumendo la soglia di significatività `p = 0.05`):

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

data:  mydata
X-squared = 3.3761, df = 1, p-value = 0.06615
```

Da notare che senza la correzione di Yates per la continuità (**correct = FALSE**) si potrebbe pensare ad una differenza significativa (p = 0.04783):

```
        Pearson's Chi-squared test

data:  mydata
X-squared = 3.9159, df = 1, p-value = 0.04783
```

In genere si consiglia di utilizzare il **test di Fisher** quando:
→ il totale delle osservazioni è inferiore a 20 *oppure*
→ in una delle celle abbiamo un valore osservato inferiore a 10 *oppure*
→ in una delle celle abbiamo un valore atteso inferiore a 5.

Nel caso della valutazione di un test diagnostico per una malattia rara è stato possibile reclutare solamente 7 malati. Sono conseguentemente stati reclutati altrettanti soggetti sani di controllo. IL test risultava positivo in 3 malati e negativo in 4 malati. Era invece negativo in 6 e positivo solamente in 1 dei soggetti sani. Il numero di osservazioni era quindi molto ridotto di osservazioni e ricorrevano addirittura tutte e tre le condizioni per l'utilizzo del test di Fisher indicate sopra. I dati sono stati raccolti in questa tabella:

Esito	Sano	Malato
Test positivo	1	3
Test negativo	6	4

Scaricate e salvate nella cartella `C:\R\` il file Fisher_2x2.csv che contiene i dati. Quindi copiate e incollate nella `Console di R` ed eseguite questo codice:

```
# con la prima riga sono importati i dati
mydata <- read.table("c:/R/Fisher_2x2.csv", header=TRUE, sep=";", row.names="Esito")
# i dati sono mostrati nella Console di R
mydata
# esegue il test di Fisher
fisher.test(mydata)
#
```

Ecco il risultato del test di Fisher come compare nella `Console di R`:

```
        Fisher's Exact Test for Count Data
```

```
data:  mydata
p-value = 0.5594
alternative hypothesis: true odds ratio is not equal to 1
95 percent confidence interval:
 0.003646918 4.442542966
sample estimates:
odds ratio
 0.2480182
```

Il valore di p = 0.5594 conferma l'esistenza di una differenza non significativa. In questo caso viene testata anche l'ipotesi che l'odd ratio sia diverso da 1. Se i limiti di confidenza dell'odd ratio includono il valore 1, la differenza non è significativa. Qui abbiamo un odd ratio di 0.2480182 i cui limiti di confidenza al 95% vanno da 0.003646918 a 4.442542966 e che quindi includono il valore 1. La conclusione è evidente: se questa è l'incertezza delle nostre conclusioni, incertezza che include il valore 1, dobbiamo dedurne ancora una volta che la differenza tra gli esiti osservata non è significativa. Va da sé che trarre conclusioni da un numero così ridotto di osservazioni è comunque una pratica possibilmente da evitare.

Il **test di McNemar** viene applicato quando le osservazioni non sono indipendenti. Un caso tipico è quello di 200 pazienti quali prima di un trattamento è stato effettuato un test diagnostico, che poteva risultare positivo o negativo. Tutti i 200 pazienti sono stati successivamente sottoposti ad un trattamento terapeutico al termine del quale il test è stato ripetuto. Dei 120 pazienti cui il test è risultato positivo prima del trattamento, dopo il trattamento 90 sono risultati positivi al test e 30 sono risultati negativi al test. Degli 80 pazienti cui il test è risultato negativo prima del trattamento, dopo il trattamento 20 sono risultati positivi al test e 60 sono risultati negativi al test. I dati sono stati raccolti in questa tabella:

Esito	Dopo positivo	Dopo negativo
Prima positivo	90	30
Prima negativo	20	60

Scaricate e salvate nella cartella `C:\R\` il file McNemar_2x2.csv che contiene i dati. Quindi copiate e incollate nella `Console di R` ed eseguite questo codice:

```
# con la prima riga sono importati i dati
mydata <- read.table("c:/R/McNemar_2x2.csv", header=TRUE, sep=";", row.names="Esito")
# viene generato un oggetto matrice contenente i dati per esigenze delle funzioni successive
matrix <- data.matrix(mydata)
# i dati sono mostrati nella Console di  R
matrix
# esegue il test di McNemar
mcnemar.test(matrix, correct=TRUE)
# l'analisi dei dati viene arricchita con un grafico a barre
barplot(t(matrix), beside=TRUE, legend=TRUE, ylim=c(0,200), col=c("darkblue","red"), ylab="Frequenze osservate nel campione", xlab="Esito prima del trattamento", main="Esito del trattamento in 200 soggetti")
#
```

Ecco il risultato del test di McNemar come compare nella `Console di R`:

```
        McNemar's Chi-squared test with continuity correction

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

Il valore di p = 0.2031 conferma l'esistenza di una differenza non significativa. La rappresentazione dei dati

sotto forma di un grafico a barre ci aiuta nella sintesi e nella interpretazione dei risultati del test di McNemar (**Figura 3.1**).

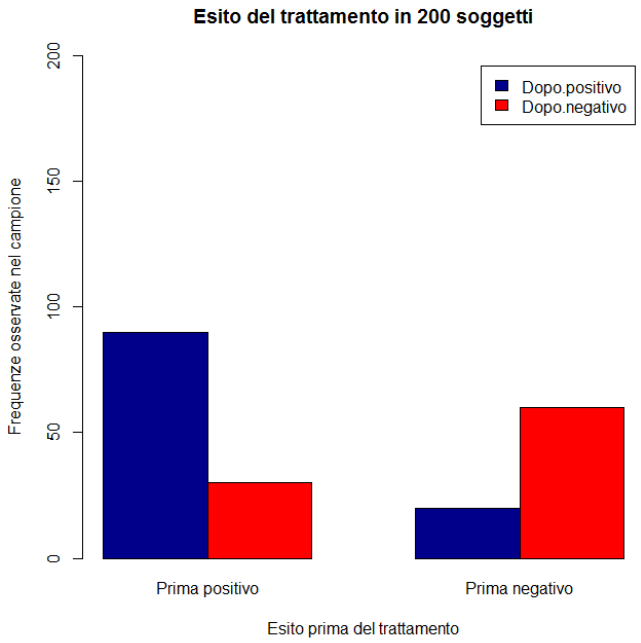


Figura 3.1 Grafico a barre dei dati sottoposti al test di McNemar in quanto si trattava di dati appaiati (esito di un test, positvo o negativo, negli stessi 200 soggetti prima e dopo uno specifico trattamento).

3.1.3. Test chi-quadrato per tabelle di contingenza (r righe per c colonne)

Il test chi-quadrato per una tabella di contingenza di **r** righe per **c** colonne è la forma più generalizzata del test.

Cinque terreni di coltura selettivi per lo *Streptococcus pyogenes* sono stati provati al fine di valutare la loro capacità di fornire un isolamento selettivo delle colonie dopo la semina di un tampone rinofaringeo. L'esito di ciascuna prova è stato registrato, e i risultati sono stati raccolti in questa tabella:

Esito	Terreno_A	Terreno_B	Terreno_C	Terreno_D	Terreno_E
Isolamento no	39	44	20	41	42
Isolamento si	177	166	200	183	168

Scaricate e salvate nella cartella `C:\R\` il file Chiquad_rxc.csv che contiene i dati. Quindi copiate e incollate nella `Console di R` ed eseguite questo codice:

```
# con la prima riga sono importati i dati
mydata <- read.table("c:/R/Chi_rxc.csv", header=TRUE, sep=";", row.names="Esito")
# calcola il chi-quadrato
chisq.test(mydata)
# ricalcola p mediante una simulazione Monte Carlo con un milione di replicati
chisq.test(mydata, simulate.p.value = TRUE, B = 1000000)
#
```

Dopo avere importato i dati (prima riga di codice), con la seconda riga di codice viene calcolato il test chi-quadrato con il valore di p determinato a partire dalla distribuzione teorica di chi-quadrato:

```
        Pearson's Chi-squared test

data:  mydata
X-squared = 13.6785, df = 4, p-value = 0.008395
```

Con la terza riga di codice viene nuovamente calcolato il test chi-quadrato: questa volta però il valore di p viene calcolato con il metodo Monte Carlo utilizzando un milione di replicati:

```
        Pearson's Chi-squared test with simulated p-value (based on 1e+06
replicates)

data:  mydata
X-squared = 13.6785, df = NA, p-value = 0.008293
```

Il valore di p corrispondente alla statistica chi-quadrato (χ^2) rappresenta la probabilità di osservare per caso una differenza tra frequenze osservate e frequenze attese della grandezza di quella effettivamente osservato: se tale probabilità è sufficientemente piccola, si conclude per una differenza significativa di incidenza nei diversi gruppi dell'esito della prova (isolamento si / isolamento no). In questo caso la probabilità di osservare per caso una differenza tra frequenze osservate e frequenze attese della grandezza di quella effettivamente osservata è dell'ordine dell'8 per mille (0.008293). Possiamo concludere che la differenza non sia presumibilmente dovuta al caso, ovvero che sia statisticamente significativa.

Ma a quale terreno va imputata la differenza osservata?

In casi i questo genere può essere utile integrare il risultato numerico con una rappresentazione grafica dei dati utilizzati per calcolare il chi-quadrato. Con il codice **R** che segue proviamo a farlo sotto forma di un grafico a barre:

```
# con la prima riga sono importati i dati
mydata <- read.table("c:/R/Chiquad_rxc.csv", header=TRUE, sep=";", row.names="Esito")
# viene generato un oggetto matrice contenente i dati per esigenze delle funzioni successive
matrix <- data.matrix(mydata)
# l'analisi dei dati viene arricchita con un grafico a barre utile per una sintesi dei risultati
barplot(t(matrix), beside=TRUE, legend=TRUE, ylim=c(0,300), col=c("red","orange", "yellow", "green",
"skyblue"), ylab="Frequenze osservate", xlab="Esito della coltura in termini di isolamento delle colonie",
main="Valutazione dell'isolamento di Streptococcus pyogenes")
#
```

In effetti il grafico a barre ci aiuta a individuare nel terreno C quello che consente di avere il migliore isolamento delle colonie (**Figura 3.2**).

La conferma numerica del dato è ulteriormente suffragata viene da una tabella nella quale, utilizzando le funzioni più elementari di Excel e OpenOffice Calc, a partire dai dati originali sopra riportati e contenuti nel file `Chiquad_rxc.csv`, sono stati calcolati i valori percentuali di successo (isolamento si) e di insuccesso (isolamento no) per ciascun terreno:

Esito	Terreno_A	Terreno_B	Terreno_C	Terreno_D	Terreno_E
Isolamento no	18.1	21.0	9.1	18.3	20.0
Isolamento si	81.9	79.0	90.9	81.7	80.0

La conclusione è che il terreno di coltura C fornisce una percentuale di isolamento delle colonie del 91% circa, che risulta quindi essere la migliore rispetto a quella di tutti gli altri terreni, e che con lo stesso terreno di coltura C si osserva una percentuale di insuccessi del 9% circa, che risulta inferiore a quella di tutti gli altri terreni.

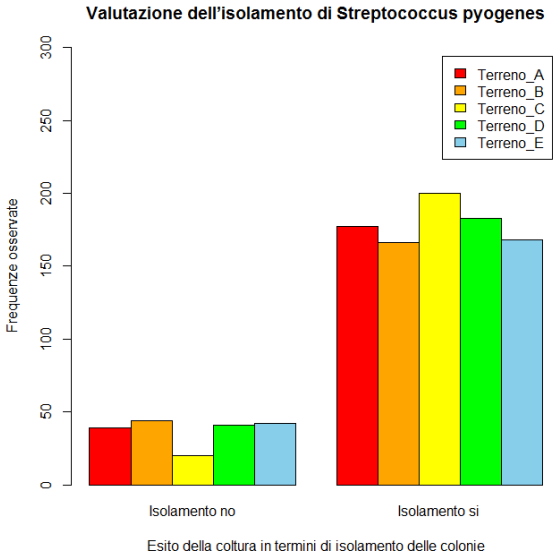


Figura 3.2 Grafico a barre dei dati sottoposti al test chi-quadrato (capacita da parte di 5 diversi terreni di coltura di fornire colonie isolate di Streptococcus pyogenes). Il terreno C è quello che fornisce i migliori risultati.

3.2. Statistiche elementari

Scaricate e salvate nella cartella `C:\R\` il file Trigliceridi.csv. Contiene la concentrazione nel siero dei trigliceridi (espressi in mg/dL) in 1000 soggetti, scelti a caso tra i pazienti che hanno effettuano ambulatoriamente questa insieme ad altre analisi di laboratorio. I risultati sono contenuti nella variabile `Trigliceridi` a sua volta contenuta nel file `Trigliceridi.csv`.

Ora copiate e incollate nella `Console di R` ed eseguite questo codice:

```
# con la prima riga sono importati i dati
mydata <- read.table("c:/R/Trigliceridi.csv", header=TRUE, sep=";")
attach(mydata) # consente di effettuare i calcoli utilizzando direttamente il nome della variabile
mean(Trigliceridi) # calcola la media
sd(Trigliceridi) # calcola la deviazione standard
#
```

Le conclusioni tratte con **R** sono:

```
> mean(Trigliceridi) # calcola la media
[1] 136.509
> sd(Trigliceridi) # calcola la deviazione standard
[1] 92.76759
```

Dopo avere analizzato statisticamente i dati, riassumiamo le conclusioni che abbiamo attenuto a un nostro amico, comunicandogli questa informazione: "*Abbiamo verificato che in 1000 soggetti, scelti a caso tra i pazienti che effettuano ambulatoriamente analisi di laboratorio, la concentrazione media dei trigliceridi era 136.509 mg/dL, con una deviazione standard (ds) di 92.76759 mg/dL*".
Quale conclusione il nostro amico può dedurre dai dati che gli abbiamo fornito?

Il nostro amico sa che in base alle proprietà della distribuzione gaussiana tra la media – 1.96 d.s. e la media + 1.96 d.s. si deve trovare il 95% dei dati campionari. Prende il valore della media (136.509 mg/dL), prende il valore della deviazione standard (92.76759 mg/dL) e lo moltiplica per 1.96 ottenendo 181.8244764. Da questo il nostro amico deduce che il 95% dei valori misurati (dopo arrotondamento dei risultati a una cifra decimale) cadeva nell'intervallo

136.5±181.8 mg/dL

e pertanto conclude che una certa quota dei nostri pazienti aveva valori negativi della concentrazione nel siero dei trigliceridi (visto che 136.5 – 181.4 = - 45.3). Tuttavia, prima di pubblicare questa rivoluzionaria conclusione, la prudenza impone un riesame dei risultati effettivamente ottenuti: e il riesame dei dati (aprite il file `Trigliceridi.csv` con Excel o OpenOffice Calc) dimostra che, ovviamente, nessuno dei valori misurati era inferiore a zero. Un evidente paradosso biologico non porta ad una rivoluzionaria conclusione, ma più semplicemente consente di individuare un grave errore nel dare forma ai dati: calcolando media e deviazione standard è stata (implicitamente) data forma gaussiana a dati che non sono distribuiti in modo gaussiano.

La media e la deviazione standard possono essere utilizzate solamente nel caso in cui i dati seguono una distribuzione gaussiana. Poiché media e deviazione standard sono i "parametri" che descrivono una distribuzione gaussiana, i 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 specifici riguardanti la distribuzione, che sono detti "metodi non parametrici". Nell'ambito delle statistiche elementari viene dato ampio spazio ai test che consentono di verificare se i dati sono distribuiti in modo gaussiano. Se i dati non sono distribuiti in modo gaussiano è necessario utilizzare test statistici non parametrici. Altrimenti si può arrivare a conclusioni grottesche come quelle riportate sopra. Nel caso delle statistiche elementari, l'alternativa non parametrica a media e deviazione standard è rappresentata dalla mediana, dalla distanza interquartile e dai quantili (o frattili) non parametrici. Tuttavia test non parametrici sono disponibili, oltre che per le statistiche elementari, anche per il confronto tra metodi e per la regressione lineare, come vedremo trattando questi argomenti.

3.2.1. Verifica della gaussianità dei dati.

Scaricate e salvate nella cartella `C:\R\` il file Statelem.csv. Nella prima riga sono riportati i nomi delle variabili, nelle successive i dati, relativi a quasi settemila casi per quali erano disponibili sesso, età, e i valori di colesterolo, colesterolo HDL, colesterolo LDL e trigliceridi (nel siero, in mg/dL). Come si vede alcuni dati possono mancare (nel secondo caso mancano i risultati di colesterolo LDL e trigliceridi, nel terzo caso manca il risultato del colesterolo LDL, e altri valori mancano nei casi successivi).

Sesso	Eta	Colesterolo	HDL	LDL	Trigliceridi
M	33	56	44	9	19
M	62	60	5		
F	90	70	30		99
M	75	80	53		
F	32	82	51		23
M	71	84	25		
F	86	89			
F	64	91	35		88

Copiate e incollate nella `Console di R` il codice che segue per analizzare la variabile `Colesterolo`. Fatelo una riga alla volta soffermandovi sui singoli passaggi per familiarizzare con il linguaggio. Da notare che viene utilizzata la libreria **nortest** che, se non lo avete ancora fatto, dovete scaricare dal CRAN (in caso contrario si verificherà un errore nell'esecuzione del codice laddove è previsto l'utilizzo della libreria).

```
# con la prima riga sono importati i dati
mydata <- read.table("c:/R/Statelem.csv", header=TRUE, sep=";")
# nome delle variabili contenute in mydata
names(mydata)
# struttura dell'oggetto mydata
```

str(mydata)
lista dei primi 10 casi di mydata
head(mydata, n=10)
lista degli ultimi 5 casi di mydata
tail(mydata, n=5)
mostra i casi con dati NA (Not Available)
mydata[!complete.cases(mydata),]
crea un nuovo oggetto denominato newdata omettendo i casi con dati NA
newdata <- na.omit(mydata)
crea un oggetto denominato newdataset che include solamente le colonne da 2 a 6 con le variabili quantitative
newdataset <- newdata[c(2,3,4,5,6)]
struttura di newdataset
str(newdataset)
crea un vettore che contiene i dati della variabile "Colesterolo"
avector <- newdataset[, "Colesterolo"]
apre la libreria nortest che contiene vari test di gaussianità
library(nortest)
test di Anderson-Darling
ad.test(avector)
test di Cramer-von Mises
cvm.test(avector)
test di Lilliefors (Kolmogorov-Smirnov)
lillie.test(avector)
test chi-quadrato di Pearson
pearson.test(avector)
test di Shapiro-Francia
sf.test(avector)
traccia un istogramma dei dati
hist(avector, main="Istogramma dei dati osservati", xlab="Colesterolo totale in mg/dL", ylab = "Frequenza")
traccia la distribuzione di densità dei dati
windows() # apre una nuova finestra
plot(density(avector), main="Distribuzione di densità dei dati osservati", xlab="Colesterolo totale in mg/dL", ylab = "Frequenza")
traccia la distribuzione cumulativa empirica dei dati
windows() # apre una nuova finestra
plot(ecdf(avector), main="Distribuzione cumulativa empirica dei dati", xlab="Colesterolo totale in mg/dL", ylab = "Frequenza cumulativa")
traccia il grafico che mostra l'adeguatezza dei dati a una distribuzione gaussiana
windows() # apre una nuova finestra
zetavector<-(avector-mean(avector))/sd(avector) # calcola la deviata normale standardizzata
qqnorm((zetavector), main="Quantili campionari vs. quantili teorici", xlab="Quantili teorici", ylab = "Quantili campionari") # grafico dei quantili campionari vs. quantili teorici
abline (0,1) # retta a 45 gradi di riferimento
#
A questo punto, dato che con le tre istruzioni **windows()** avete aperto tre nuove finestre, avrete un totale di quattro finestre, con altrettanti grafici, sovrapposte. Spostate o iconizzate la finestra dell'ultimo grafico per vedere la finestra con il grafico precedente, e così via. Infine utilizzate i tasti `Pag-su` e `Pag-giù` per scorrere nella finestra della `Console di R` il codice eseguito e leggerne i risultati.

I test di gaussianità concordano tutti sul fatto che i valori del `Colesterolo` non sono distribuiti in modo gaussiano. La probabilità di osservare per caso uno scostamento dalla distribuzione gaussiana dell'entità di

quello osservato per il colesterolo è molto bassa (p variabile con i vari test, ma sempre almeno inferiore a 0.001), quindi lo scostamento della distribuzione del colesterolo dalla distribuzione gaussiana è da ritenersi statisticamente significativo. Ecco il risultato dei singoli test:

```
> ad.test(avector)

        Anderson-Darling normality test

data:  avector
A = 2.0806, p-value = 2.733e-05

> cvm.test(avector)

        Cramer-von Mises normality test

data:  avector
W = 0.321, p-value = 0.0001917

> lillie.test(avector)

        Lilliefors (Kolmogorov-Smirnov) normality test

data:  avector
D = 0.0287, p-value = 9.183e-05

> pearson.test(avector)

        Pearson chi-square normality test

data:  avector
P = 83.9086, p-value = 0.0001891

> sf.test(avector)

        Shapiro-Francia normality test

data:  avector
W = 0.9931, p-value = 8.518e-09
```

Ecco ora un secondo esempio che ripete le stesse cose con un codice più compatto e più semplice sulla variabile Trigliceridi che presenta una asimmetria molto marcata con una lunghissima coda sulla destra, ben documentata nei grafi che sono prodotti:

```
# con la prima riga sono importati i dati
mydata <- read.table("c:/R/Verigauss.csv", header=TRUE, sep=";")
# elimina i casi con dati mancanti
newdata <- na.omit(mydata)
# carica la libreria necessaria
library(nortest)
# calcola i vari test statistici
ad.test(newdata$Trigliceridi)
cvm.test(newdata$Trigliceridi)
lillie.test(newdata$Trigliceridi)
pearson.test(newdata$Trigliceridi)
```

```
sf.test(newdata$Trigliceridi)
# traccia un semplice istogramma
hist(newdata$Trigliceridi, main="Istogramma dei dati osservati", xlab="Trigliceridi in mg/dL", ylab = "Frequenza")
# traccia il kernel density plot
windows() # apre una nuova finestra
plot(density(newdata$Trigliceridi), main="Distribuzione di densità dei dati osservati", xlab="Trigliceridi in mg/dL", ylab = "Frequenza")
# funzione di distribuzione cumulativa empirica (osservata) dei dati
windows() # apre una nuova finestra
plot(ecdf(newdata$Trigliceridi), main="Distribuzione cumulativa empirica dei dati", xlab="Trigliceridi in mg/dL", ylab = "Frequenza cumulativa")
# confronta i quantili campionari (punti) con i quantili teorici (retta)
windows() # apre una nuova finestra
zetavector<-(newdata$Trigliceridi-mean(newdata$Trigliceridi))/sd(newdata$Trigliceridi)
qqnorm((zetavector), main="Quantili campionari vs. quantili teorici", xlab="Quantili teorici", ylab = "Quantili campionari")
abline (0,1)
#
```

I risultati dell'analisi grafica sono oltremodo interessanti (**Figura 3.3**)

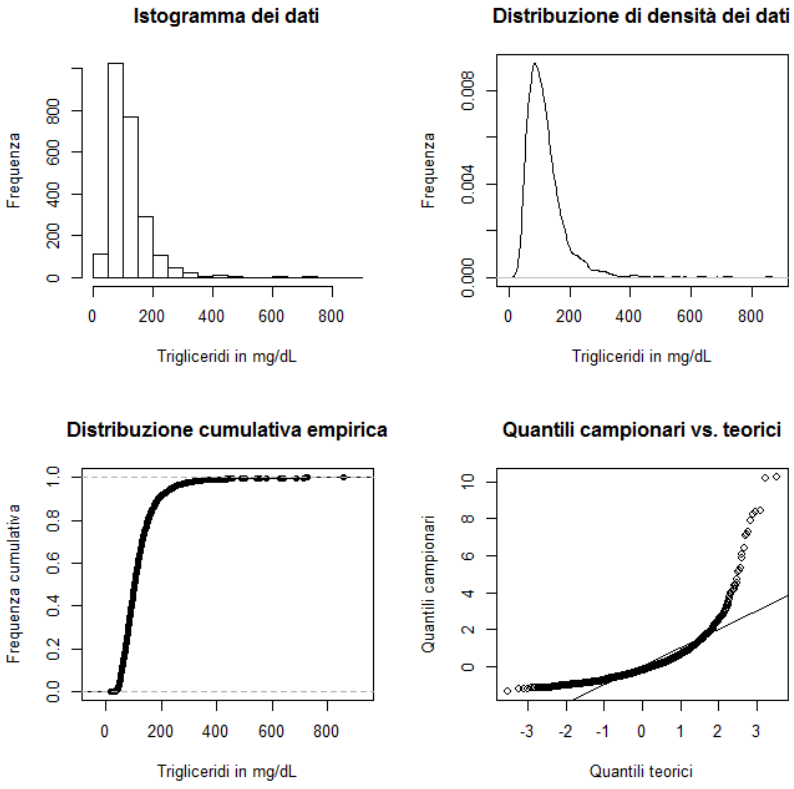


Figura 3.3 Analisi grafica della distribuzione della concentrazione dei trigliceridi nel siero. Istogramma, kernel density plot e quantili campionari confrontati con i quantili di una distribuzione gaussiana teorica, danno evidenza della forte asimmetria della distribuzione che pertanto non è gaussiana.

Anche in questo caso, dato che con le tre istruzioni **windows()** avete aperto tre nuove finestre, avrete un totale di quattro finestre, con altrettanti grafici, sovrapposte. Spostate o iconizzate la finestra dell'ultimo

grafico per vedere la finestra con il grafico precedente, e così via.

Sia l'istogramma, sia la distribuzione di densità dei dati, sia la retta dei quantili teorici sulla quale dovrebbero essere allineati i quantili campionari qualora la distribuzione fosse gaussiana, forniscono una chiara evidenza del fatto che i valori dei `Trigliceridi` non sono distribuiti in modo gaussiano. E rappresentano una conferma dei vari test statistici eseguiti, i cui risultati erano i seguenti:

```
> ad.test(newdata$Trigliceridi)

        Anderson-Darling normality test

data:  newdata$Trigliceridi
A = Inf, p-value = NA

> cvm.test(newdata$Trigliceridi)

        Cramer-von Mises normality test

data:  newdata$Trigliceridi
W = 19.6163, p-value = 7.37e-10

Warning message:
In cvm.test(newdata$Trigliceridi) :
  p-value is smaller than 7.37e-10, cannot be computed more accurately
> lillie.test(newdata$Trigliceridi)

        Lilliefors (Kolmogorov-Smirnov) normality test

data:  newdata$Trigliceridi
D = 0.1423, p-value < 2.2e-16

> pearson.test(newdata$Trigliceridi)

        Pearson chi-square normality test

data:  newdata$Trigliceridi
P = 1001.822, p-value < 2.2e-16

> sf.test(newdata$Trigliceridi)

        Shapiro-Francia normality test

data:  newdata$Trigliceridi
W = 0.73, p-value < 2.2e-16
```

Lo scostamento della distribuzione dei trigliceridi dalla distribuzione gaussiana è talmente consistente che il test di Anderson-Darling non è computabile, e il test di Cramer-von Mises riporta il valore di p semplicemente come "inferiore al valore minimo computabile". La probabilità di osservare per caso uno scostamento dalla distribuzione gaussiana dell'entità di quello osservato per i trigliceridi è molto bassa, con p variabile con i vari test, ma sempre almeno inferiore a `0.000000001`. La distribuzione dei trigliceridi dista anni luce da una distribuzione gaussiana.

Per la documentazione dell'apparato teorico matematico-statistico che sta dietro a ciascuno dei test utilizzati si rimanda alla documentazione sul web.

3.2.2. Statistiche esplorative (misure di posizione, misure di dispersione, quantili)

Scaricate e salvate nella cartella C:\R\ il file Statelem.csv. Il file contiene gli stessi dati utilizzati per la precedente verifica della gaussianità.

Da notare che nel codice che segue sono impiegate le librerie **Hmisch**, **pastsecs** e **psych** che dovete scaricare dal CRAN prima di eseguirlo (in caso contrario si verificherà un errore nell'esecuzione del codice laddove è previsto l'utilizzo delle librerie). Copiate e incollate nella `Console di R` il codice ed eseguitelo una riga alla volta per familiarizzare con il linguaggio soffermandovi sui singoli passaggi:

```
# con la prima riga sono importati i dati
mydata <- read.table("c:/R/Statelem.csv", header=TRUE, sep=";")
# nome delle variabili contenute in mydata
names(mydata)
# struttura dell'oggetto mydata
str(mydata)
# lista dei primi 10 casi di mydata
head(mydata, n=10)
# lista degli ultimi 5 casi di mydata
tail(mydata, n=5)
# mostra i casi con dati NA (Not Available)
mydata[!complete.cases(mydata),]
# crea un nuovo oggetto denominato newdata omettendo i casi con dati NA
newdata <- na.omit(mydata)
# crea un oggetto denominato newdataset che include solamente le colonne da 2 a 6 con le variabili quantitative
newdataset <- newdata[c(2,3,4,5,6)]
# calcola la media del colesterolo LDL su mydata
attach(mydata)
mean(LDL) # da notare che R restituisce NA anche per la media
mean(LDL, na.rm=TRUE) # rimuovendo i dati NA questa volta R restituisce il valore della media
# è ora facile calcolare le altre principali statistiche del colesterolo LDL su mydata
var(LDL, na.rm=TRUE) # calcola la varianza
sd(LDL, na.rm=TRUE) # calcola la deviazione standard
min(LDL, na.rm=TRUE) # calcola il valore minimo
max(LDL, na.rm=TRUE) # calcola il valore massimo
range(LDL, na.rm=TRUE) # calcola il range
quantile(LDL, probs = seq (0, 1, 0.25), na.rm=TRUE) # calcola i quartili, sostituendo in seq() il valore 0.25 con 0.10 calcola i decili, eccetera...
# con sapply si calcolano le statiche di tutte le variabili (questa volta su newdataset, nel quale sono inclusi solamente i dati completi, e dal quale è stata esclusa la variabile qualitativa Sesso)
sapply(newdataset, mean) # calcola la media
sapply(newdataset, sd) # calcola la deviazione standard
sapply(newdataset, var) # calcola la varianza
sapply(newdataset, min) # calcola il valore minimo
sapply(newdataset, max) # calcola il valore massimo
sapply(newdataset, range) # calcola il range
sapply(newdataset, median) # calcola la mediana
sapply(newdataset, quantile) # calcola i quartili
# se volete avere rapidamente le statistiche sintetiche di mydata
summary(mydata)
# queste sono le statistiche che potete ottenere utilizzando la libreria Hmisc
library(Hmisc)
```

describe(mydata)
queste sono le statistiche che potete ottenere utilizzando la libreria pastecs
library(pastecs)
stat.desc(mydata)
queste sono le statistiche che potete ottenere utilizzando la libreria psych
library(psych)
describe(mydata)
#

Come si vede nella prima parte del codice è possibile calcolare le statistiche una per una per ciascuna variabile, come ad esempio con le funzioni **min()**, **max()**, **range()**.

Guardiamo ora in particolare la riga di codice utilizzata per calcolare i quartili del colesterolo LDL e cosa accade eseguendola:

```
> quantile(LDL, probs = seq (0, 1, 0.25), na.rm=TRUE) # calcola i quartili, sostituendo in seq() il valore 0.25 con 0.10 calcola i decili, eccetera...
    0%    25%    50%    75%   100%
  9.00 100.25 123.50 147.00 292.00
```

Nella riga superiore sono state tabulate le etichette che indicano il valore minimo (0%), il valore massimo (100%), il primo quartile (25%), il secondo quartile (50% ovvero la mediana), e il terzo quartile (75%), nella riga inferiore sono stati tabulati i valori di colesterolo LDL (in mg/dL) corrispondenti.

Sostituendo in **seq()** il valore 0.25 con 0.1 potete calcolare i decili:

```
> quantile(LDL, probs = seq (0, 1, 0.1), na.rm=TRUE)
   0%   10%   20%   30%   40%   50%   60%   70%   80%   90%  100%
  9.0  83.0  96.0 106.0 115.0 123.5 133.0 142.0 154.0 168.0 292.0
```

Sostituendo in **seq()** il valore 0.1 con 0.01 potete calcolare i percentili:

```
> quantile(LDL, probs = seq (0, 1, 0.01), na.rm=TRUE)
    0%     1%     2%     3%     4%     5%     6%     7%     8%     9%    10%
  9.00  54.00  63.46  68.00  71.00  73.00  75.00  76.11  79.00  81.00  83.00
   11%    12%    13%    14%    15%    16%    17%    18%    19%    20%    21%
 84.00  86.00  87.00  88.00  90.00  91.00  93.00  94.00  95.00  96.00  97.00
   22%    23%    24%    25%    26%    27%    28%    29%    30%    31%    32%
 98.00  98.00 100.00 100.25 101.00 102.00 103.44 105.00 106.00 107.00 107.00
   33%    34%    35%    36%    37%    38%    39%    40%    41%    42%    43%
108.00 109.00 110.00 111.00 112.00 113.00 114.00 115.00 115.00 116.00 117.00
   44%    45%    46%    47%    48%    49%    50%    51%    52%    53%    54%
118.00 119.00 120.00 121.00 122.00 123.00 123.50 124.23 125.00 127.00 127.00
   55%    56%    57%    58%    59%    60%    61%    62%    63%    64%    65%
128.00 129.00 130.00 131.00 132.00 133.00 134.00 135.00 135.99 136.00 137.00
   66%    67%    68%    69%    70%    71%    72%    73%    74%    75%    76%
138.00 140.00 140.00 142.00 142.00 143.00 144.00 145.00 146.02 147.00 148.00
   77%    78%    79%    80%    81%    82%    83%    84%    85%    86%    87%
150.00 151.00 152.67 154.00 155.00 156.00 157.00 159.00 161.00 162.00 163.00
   88%    89%    90%    91%    92%    93%    94%    95%    96%    97%    98%
165.00 167.00 168.00 171.00 173.00 175.00 179.00 182.00 186.00 192.00 200.00
   99%   100%
213.27 292.00
```

Cambiando semplicemente il nome della variabile, è possibile effettuare il calcolo dei quartili sui trigliceridi:

```
> quantile(Trigliceridi, probs = seq (0, 1, 0.25), na.rm=TRUE)
  0%  25%  50%  75% 100%
  19   75  102  139 1248
```

o ancora sul colesterolo HDL:

```
> quantile(HDL, probs = seq (0, 1, 0.25), na.rm=TRUE)
  0%  25%  50%  75% 100%
   5   50   60   73  146
```

Questo è anche un buon esempio di come il codice fornito possa essere facilmente riutilizzato per adattarlo ai propri dati.

Ma come si vede è anche possibile calcolare una singola statistica su tutte le variabili contemporaneamente con la funzione **sapply()**.

Infine le librerie **Hmisch**, **pastsecs** e **psych** forniscono automaticamente un riepilogo complessivo delle principali statistiche di tutte le variabili. Da notare che nel riepilogo fornito dalla libreria **psych** la colonna `mad` riporta il valore della "median absolute deviation", cioè la mediana delle deviazioni assolute dalla mediana, che è l'equivalente non parametrico della deviazione standard.

```
             var    n   mean    sd median trimmed   mad min  max range  skew kurtosis   se
Sesso*         1 6787   1.43  0.49    1.0    1.41  0.00   1    2     1  0.30    -1.91 0.01
Eta            2 6787  59.01 17.66   62.0   59.89 19.27   3  104   101 -0.41    -0.45 0.21
Colesterolo    3 6787 208.74 42.33  207.0  207.85 41.51  56  435   379  0.27     0.44 0.51
HDL            4 5918  62.15 17.41   60.0   61.19 17.79   5  146   141  0.55     0.38 0.23
LDL            5 2874 125.17 34.04  123.5  124.22 34.84   9  292   283  0.33     0.29 0.64
Trigliceridi   6 5625 117.88 73.88  102.0  106.91 45.96  19 1248  1229  4.11    33.09 0.99
```

In una distribuzione perfettamente gaussiana la media coincide con la mediana, e la deviazione standard coincide con la mad. Tanto più i dati si discostano da una distribuzione gaussiana, tanto più la media differisce dalla mediana e/o tanto più la deviazione standard differisce dalla mad: questo avviene nel caso dei trigliceridi, che abbiamo visto in precedenza essere distribuiti in modo grossolanamente non gaussiano. Il fatto che si tratti di una distribuzione non gaussiana è confermato anche dal valore del coefficiente di asimmetria (`skew`) e del coefficiente di curtosi (`kurtosis`).

3.3. Confronto tra medie

Il confronto tra medie può essere effettuato sia nel caso di campioni indipendenti sia nel caso di dati appaiati. Accanto alla versione tradizionale parametrica, rappresentata dal test t di Student, esistono gli equivalenti non parametrici, che devono essere utilizzati se i dati non sono distribuiti in modo gaussiano. Quindi anche nel caso del confronto tra medie deve essere effettuata una analisi preliminare dei dati per decidere quale sia il test appropriato.

3.3.1. Confronto tra due campioni indipendenti (test t di Student e test non parametrici)

Scaricate e salvate nella cartella C:\R\ il file Statind.csv.

Si tratta dei dati relativi alla determinazione della concentrazione della riboflavina in due tessuti, il fegato e il muscolo. Si tratta ovviamente di campioni indipendenti. Il contenuto del file aperto con un editor di testo come il `Blocco note di Windows` vi apparirà così, con i nomi delle variabili nella prima riga, e i dati di ciascun caso nelle righe successive:

```
Tessuto;Riboflavina
Fegato;0.95
Fegato;2.18
Fegato;1.12
Fegato;1.86
Muscolo;0.22
```

```
Muscolo;0.18
Muscolo;0.46
Muscolo;0.86
Muscolo;0.64
Muscolo;0.28
Muscolo;0.33
Muscolo;0.35
Muscolo;0.42
```

Come separatore di campo viene utilizzato il punto e virgola (`;`). Nella prima riga sono riportati i nomi delle variabili, nelle successive i dati. La variabile `Tessuto` della prima colonna è la variabile classificativa che specifica il tipo di tessuto nel quale è stata determinata la concentrazione di riboflavina, che a sua volta è riportata poi nella variabile numerica `Riboflavina` della seconda colonna.

Copiate e incollate nella `Console di` R ed eseguite questo codice:

```
# con la prima riga sono importati i dati
mydata <- read.table("c:/R/Statind.csv", header=TRUE, sep=";")
# con il test F per il rapporto tra varianze si verifica se le varianze delle misure effettuate nel tessuto e nel muscolo sono omogenee
attach(mydata)
var.test(Riboflavina~Tessuto) # le varianze non sono omogenee!
# test t di Student per due campioni indipendenti con varianze non omogenee
t.test(Riboflavina~Tessuto, var.equal = FALSE)
# in alternativa si può applicare il test U di Mann-Whitney per campioni indipendenti (test non parametrico)
wilcox.test(Riboflavina~Tessuto)
# in alternativa si può applicare il test di Kruskal-Wallis (test non parametrico)
kruskal.test(Riboflavina~Tessuto)
#
```

Dopo avere importato i dati viene effettuata l'analisi della varianza (**var.test(Riboflavina~Tessuto)**) per verificare se i due campioni hanno varianze omogenee:

```
        F test to compare two variances

data:  Riboflavina by Tessuto
F = 7.3934, num df = 3, denom df = 8, p-value = 0.02156
alternative hypothesis: true ratio of variances is not equal to 1
95 percent confidence interval:
   1.36511 107.49899
sample estimates:
ratio of variances
          7.393386
```

In effetti il test F per il rapporto tra varianze con un `p = 0.02156` indica una varianza significativamente differente tra i risultati ottenuti nel tessuto e quelli ottenuti nel muscolo. Per questa ragione nel successivo test t di Student compare l'argomento **var.equal = FALSE** (se le varianze fossero state omogenee si sarebbe utilizzato l'argomento **var.equal = TRUE**):

```
        Welch Two Sample t-test

data:  Riboflavina by Tessuto
t = 3.6757, df = 3.367, p-value = 0.02867
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 0.2059442 2.0179447
sample estimates:
 mean in group Fegato mean in group Muscolo
```

```
          1.5275000             0.4155556
```

La probabilità di osservare per caso una differenze tra le medie come quella effettivamente osservata è pari al 2,8% circa. Pertanto si conclude che le medie sono significativamente diverse: si osserva nel fegato una concentrazione di riboflavina maggiore di quella osservata nel muscolo.

I successivi due test non parametrici, il test U di Mann-Whitney e il test di Kruskal-Wallis, non sono sensibili alle differenze tra le varianze nei due campioni, e non necessitano quindi la correzione che si rende invece necessaria per il test t di Student.

```
        Wilcoxon rank sum test

data:  Riboflavina by Tessuto
W = 36, p-value = 0.002797
alternative hypothesis: true location shift is not equal to 0

        Kruskal-Wallis rank sum test

data:  Riboflavina by Tessuto
Kruskal-Wallis chi-squared = 7.7143, df = 1, p-value = 0.005479
```

In ogni caso sia il test t di Student sia i due test non parametrici consentono di concludere che esiste una differenza significativa nella concentrazione di riboflavina nei due tessuti.

3.3.2. Confronto tra dati appaiati (test t di Student e test non parametrici)

Scaricate e salvate nella cartella C:\R\ il file Statapp.csv.

Il contenuto del file aperto con un editor di testo come il Blocco note di Windows vi apparirà così, con i nomi delle variabili nella prima riga, e i dati di ciascun caso nelle righe successive (per semplicità sono state eliminate le righe di dati intermedi):

```
Subito;Dopo24ore
17;16
.................
652;631
```

Nella prima riga sono riportati i nomi delle variabili, nelle successive i dati. La variabile `Subito` della prima colonna riporta i valori di aspartato-aminotransaminasi (AST, in U/L) misurati su un campione di siero immediatamento dopo il prelievo, mentre la variabile `Dopo24Ore` della seconda colonna riporta i valori determinati sullo stesso campione di siero dopo 24 ore di conservazione dei campioni, tappati per evitare fenomeni di evaporazione, e conservati alla temperatura di + 4 °C.

Copiate e incollate nella `Console di R` ed eseguite questo codice:

```
# con la prima riga sono importati i dati
mydata <- read.table("c:/R/Statapp.csv", header=TRUE, sep=";")
# test t di Student per dati appaiati
attach(mydata)
t.test(Subito, Dopo24ore, paired=TRUE)
# in alternativa si può applicare il test di Wilcoxon (Wilcoxon Signed Rank Test) per dati appaiati (un test non parametrico)
wilcox.test(Subito, Dopo24ore, paired=TRUE, exact=FALSE)
#
```

Sia il test ti di Student per dati appaiati con un valore di `p = 0.4718` sia il test di Wilcoxon con un valore

di p = 0.6255 consentono di concludere che la concentrazione dell'AST misurata nel siero conservato per 24 ore alla temperatura di +4 °C non differisce significativamente da quella misurata sul siero immediatamente dopo il prelievo:

```
        Paired t-test

data:  Subito and Dopo24ore
t = 0.7328, df = 21, p-value = 0.4718
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -3.926385  8.199112
sample estimates:
mean of the differences
               2.136364

        Wilcoxon signed rank test with continuity correction

data:  Subito and Dopo24ore
V = 111, p-value = 0.6255
alternative hypothesis: true location shift is not equal to 0
```

Anche in questo caso la scelta tra test parametrico (test t di Student) e non parametrico (test di Wilcoxon) può essere fatta mediante analisi della gaussianità dei dati. Mediante Excel o OpenOffice Calc è facile aggiungere ai dati originari una colonna con una nuova variabile, la Differenza (presa con il segno) tra la concentrazione trovata immediatamente dopo il prelievo e quella osservata dopo 24 ore.

Subito	Dopo24ore	Differenza
17	16	1
18	17	1
19	24	-5
20	21	-1
22	24	-2
24	25	-1
24	27	-3
30	25	5
37	42	-5
42	40	2
45	48	-3
52	57	-5
62	60	2
67	71	-4
95	86	9
101	106	-5
174	180	-6
327	300	27
433	440	-7
476	430	46
495	515	-20
652	631	21

Se effettuate un test di gaussianità sulla variabile Differenza troverete che essa non è distribuita in

modo gaussiano. Pertanto in questo caso è necessario che le conclusioni siano tratte sulla base del risultato ottenuto con il test di Wilcoxon (test non parametrico).

3.3.3. Confronto con una media teorica (test t di Student)

In laboratorio si è preparata mediante pesata e diluizione una soluzione con una concentrazione di 8.0 g/dL di albumina umana. Definiamo questo valore come la "media teorica" della concentrazione dell'albumina in quanto, nelle opportune condizioni, la misura della massa del soluto con una bilancia e la misura del volume della soluzione finale con vetreria tarata di classe A consentono di avere una concentrazione finale nota con una accuratezza che è di alcuni ordini di grandezza superiore a quella di un comune metodo analitico.

Analizzando il siero con il metodo analitico del quale si intende verificare l'accuratezza, si ottengono i seguenti valori: 8.2, 8.3, 7.9, 8.1 e 8.0 g/dL. Quello che segue è il codice **R** necessario per sapere se i risultati ottenuti si discostano significativamente dal valore assegnato.

```
# aprite la Console di R e inserite direttamente i valori ottenuti in un vettore
myvalues <- c(8.2, 8.3, 7.9, 8.1, 8.0)
# se lo desiderate potete visualizzare a scopo di verifica i dati inseriti
myvalues # mostra i dati contenuti nell'oggetto
# potete ora confrontare i cinque valori ottenuti con il valore assegnato di 8.0 g/dL mediante il test t di Student per una media teorica
t.test(myvalues, mu = 8)
#
```

I risultati sono:

```
        One Sample t-test

data:  myvalues
t = 1.4142, df = 4, p-value = 0.2302
alternative hypothesis: true mean is not equal to 8
95 percent confidence interval:
 7.903676 8.296324
sample estimates:
mean of x
      8.1
```

La media misurata di 8.1 non differisce significativamente dalla media teorica essendo `p = 0.2302`. La probabilità di osservare per caso una differenza di 0.1 tra il valore teorico e il valore misurato è del 23% circa, troppo elevata per escludere il caso dalle possibili cause della differenza. Il dato viene confermato dal fatto che i limiti di confidenza al 95% della media (uguale a 8.1) delle cinque misure effettuate sono rispettivamente 7.903676 (il limite inferiore) e 8.296324 (il limite superiore) e pertanto includono nell'incertezza delle conclusioni il valore 8 della media teorica.

3.4. Regressione lineare

La regressione lineare viene tradizionalmente associata al coefficiente di correlazione lineare r. Nonostante quest'ultimo abbia in sé un significato limitato, la possibilità che si ha con **R** di sviluppare una matrice dei coefficienti di correlazione tra più variabili e di generare matrici di scatter plot, che altro non sono che diagrammi cartesiani multipli che illustrano graficamente le relazioni tra dette variabili, rappresenta uno strumento utile per l'analisi esplorativa dei dati.

3.4.1. Correlazione (coefficiente di correlazione lineare r)

Scaricate e salvate nella cartella C:\R\ il file Statcorr.csv. I dati contenuti hanno una struttura molto semplice e il file aperto con Excel o con OpenOffice.org Calc appare così:

GR	RGO	HB	HCT	HBA2	MCV	HBF	MCH	RDW	FERRO
4.90	97	13.3	40.6	1.8	82.8	0.6	27.1	17.3	106
4.66	81	10.8	34.3	2.6	73.6	1.6	23.2	21.5	148
5.43	57	11.5	36.1	4.8	66.5	2.5	21.1	21.0	104
5.41	63	10.8	39.7	2.5	73.4	1.8	20.0	19.9	74
4.94	60	10.4	32.3	1.4	65.0	0.7	21.1	23.7	17
……	……	……	……	……	……	……	……	……	……

Le variabili contenute nel file sono gli eritrociti (GR, in 10^12/L), la resistenza globulare osmotica (RGO, in %), l'emoglobina (HB, in g/dL), l'ematocrito (in %), l'emoglobina A2 (in %), il volume globulare medio (MCV, in fL), l'emoglobina F (in %), l'emoglobina corpuscolare media (MCH, in pg), l'ampiezza della distribuzione dei globuli rossi (Red Distribution Width, in %) misurati in 643 soggetti che includevano soggetti sani e soggetti con beta-talassemia, con alfa-talassemia, con anemia sideropenica. Si tratta degli stessi dati utilizzati per illustrate gli scatter plot. Da notare che sono utilizzate la libreria **Hmisc** e la libreria **car** che, se non lo avete ancora fatto, dovete scaricare dal CRAN prima di eseguire l'esempio (in caso contrario si verificherà un errore nell'esecuzione del codice laddove è previsto l'utilizzo delle librerie).

Copiate e incollate nella `Console di R` ed eseguite questo codice:

```
# con la prima riga sono importati i dati
mydata <- read.table("c:/R/Statcorr.csv", header=TRUE, sep=";")
# matrice dei coefficienti di correlazione, method può essere pearson (il classico r), spearman, kendall
cor(mydata, use="complete.obs", method="pearson")
# calcola i coefficienti di correlazione con i livelli di significatività
library(Hmisc)
x <- as.matrix(mydata) # trasforma mydata in una matrice denominata x
rcorr(x, type="pearson") # type può essere pearson (il classico r) o spearman
#
```

Innanzitutto sono calcolati i coefficienti di correlazione `r` tra tutte le possibili combinazioni di variabili, la diagonale divide la matrice in due parti simmetriche. I valori del coefficiente di correlazione `r` sulla diagonale sono ovviamente tutti uguali esattamente a 1.00, in quanto rappresentano la correlazione di ciascuna variabile con sé stessa:

```
          GR   RGO    HB   HCT  HBA2   MCV   HBF   MCH   RDW FERRO
GR      1.00 -0.42  0.40  0.48  0.54 -0.41  0.26 -0.42  0.37  0.27
RGO    -0.42  1.00  0.43  0.37 -0.63  0.74 -0.43  0.74 -0.61 -0.02
HB      0.40  0.43  1.00  0.97 -0.03  0.64 -0.13  0.65 -0.54  0.45
HCT     0.48  0.37  0.97  1.00  0.02  0.59 -0.10  0.56 -0.48  0.46
HBA2    0.54 -0.63 -0.03  0.02  1.00 -0.42  0.44 -0.43  0.26  0.30
MCV    -0.41  0.74  0.64  0.59 -0.42  1.00 -0.32  0.97 -0.85  0.26
HBF     0.26 -0.43 -0.13 -0.10  0.44 -0.32  1.00 -0.32  0.30  0.13
MCH    -0.42  0.74  0.65  0.56 -0.43  0.97 -0.32  1.00 -0.85  0.25
RDW     0.37 -0.61 -0.54 -0.48  0.26 -0.85  0.30 -0.85  1.00 -0.31
FERRO   0.27 -0.02  0.45  0.46  0.30  0.26  0.13  0.25 -0.31  1.00

n= 643
```

Quindi viene riportato il valore di probabilità `p` di osservare per caso il valore di `r` calcolato:

```
P
        GR      RGO     HB      HCT     HBA2    MCV     HBF     MCH     RDW     FERRO
GR              0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000
RGO     0.0000          0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.5911
HB      0.0000  0.0000          0.0000  0.4331  0.0000  0.0015  0.0000  0.0000  0.0000
HCT     0.0000  0.0000  0.0000          0.5489  0.0000  0.0140  0.0000  0.0000  0.0000
HBA2    0.0000  0.0000  0.4331  0.5489          0.0000  0.0000  0.0000  0.0000  0.0000
MCV     0.0000  0.0000  0.0000  0.0000  0.0000          0.0000  0.0000  0.0000  0.0000
HBF     0.0000  0.0000  0.0015  0.0140  0.0000  0.0000          0.0000  0.0000  0.0006
MCH     0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000          0.0000  0.0000
RDW     0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000          0.0000
FERRO   0.0000  0.5911  0.0000  0.0000  0.0000  0.0000  0.0006  0.0000  0.0000
```

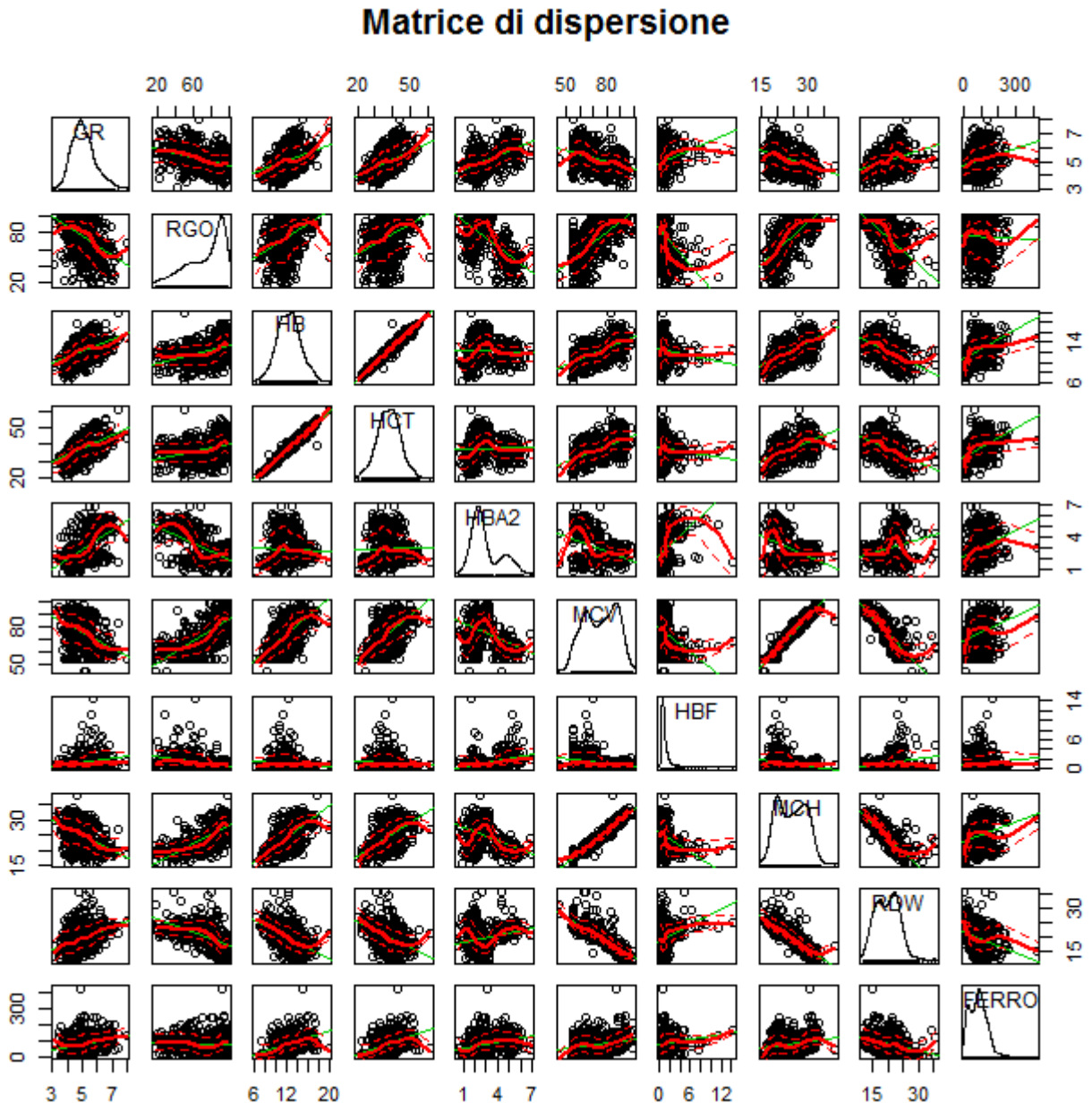


Figura 3.4 Sintesi grafica dell'analisi delle variabili mediante una matrice degli scatter plot.

Con la libreria **car** mediante una sola riga di codice potete rappresentare graficamente le relazioni tra le variabili:

#
library(car)
scatterplotMatrix(~GR+RGO+HB+HCT+HBA2+MCV+HBF+MCH+RDW+FERRO, reg.line=lm, smooth=TRUE, span=0.5, diagonal = "density", main="Matrice di dispersione", data=mydata)
#
La matrice dei diagrammi di dispersione (scatter plot) conferma le forti correlazioni esistenti tra HB e HCT e tra MCH e MCV (**Figura 3.4**).

Questa parte relativa al coefficiente di correlazione può essere utilmente integrata con la parte nella quale r è trattato in forma grafica sotto forma di correlogrammi.

3.4.2. Regressione lineare (regressione lineare semplice e regressione lineare multipla)

Scaricate e salvate nella cartella C:\R\ il file Statreglin.csv. Il contenuto del file aperto con un editor di testo come il `Blocco note di Windows` vi apparirà così, con i nomi delle variabili nella prima riga, e i dati di ciascun caso nelle righe successive:

```
Sesso;Eta;Colesterolo;HDL;LDL;Trigliceridi
M;33;56;44;9;19
F;81;101;63;26;62
.................
F;74;385;52;256;327
F;64;397;70;292;96
```

Nella prima riga sono riportati i nomi delle variabili, nelle successive i dati, relativi a oltre duemila casi per quali erano disponibili sesso, età, e i valori di colesterolo totale, colesterolo HDL, colesterolo LDL e trigliceridi (concentrazione nel siero, in mg/dL). Si tratta degli stessi dati utilizzati per le statistiche elementari, dai quali sono stati questa volta esclusi tutti i casi con dati mancanti. Da notare che sono utilizzate la libreria **car**, la libreria **relaimpo** e la libreria **gvlma** che, se non lo avete ancora fatto, dovete scaricare dal CRAN prima di eseguire l'esempio (in caso contrario si verificherà un errore nell'esecuzione del codice laddove è previsto l'utilizzo delle librerie).

Copiate e incollate nella `Console di R` ed eseguite un blocco di codice alla volta al fine di familiarizzare con questo tipo di analisi dei dati. Che è verbosa e ridondante. Ma solamente a scopo didattico, al fine di fornire una ampia panoramica del codice disponibile per questo tipo di analisi, e tesaurizzarlo per usi futuri:

con la prima riga sono importati i dati
mydata <- read.table("c:/R/Statreglin.csv", header=TRUE, sep=";")
viene calcolata la regressione lineare mediante la funzione lm(y ~ x1 + x2 + x3, data=); y è la variabile dipendente in ordinate. Se x1 è l'unica variabile indipendente, viene calcolata la regressione lineare semplice, se x1 + x2 + ... sono più variabili indipendenti viene calcolata la regressione lineare multipla
fit <- lm(LDL ~ Colesterolo + HDL + Trigliceridi, data=mydata)
calcola intercetta e coefficienti delle x
coefficients(fit)
#
Dopo avere importato di dati, la regressione multipla viene calcolata e salvata (seconda riga) in un oggetto denominato **fit**, cui più semplicemente fanno riferimento le successive funzioni, quindi sono mostrati (**coefficients(fit)**) i coefficienti dell'equazione della regressione lineare multipla:

```
(Intercept)  Colesterolo          HDL Trigliceridi
  -0.7457405    0.8907005   -0.7904253   -0.1158645
```

che quindi è

$$LDL = -0.7457405 + 0.8907005 \cdot Colesterolo - 0.7904253 \cdot HDL - 0.1158645 \cdot Trigliceridi$$

calcola gli intervalli di confidenza dell'intercetta e dei coefficienti delle x
confint(fit, level=0.95)
#
Questi sono gli intervalli di confidenza al 95% calcolati:

```
                   2.5 %     97.5 %
(Intercept)  -2.0868561  0.5953751
Colesterolo   0.8844198  0.8969812
HDL          -0.8069285 -0.7739221
Trigliceridi -0.1197046 -0.1120245
```

Il dato più interessante è che l'intercetta non è significativamente diversa da 0 (zero). Pertanto l'equazione riportata sopra può essere semplificata e riscritta come

$$LDL = 0.8907005 \cdot Colesterolo - 0.7904253 \cdot HDL - 0.1158645 \cdot Trigliceridi$$

mostra un riepilogo dei risultati
summary(fit)
ricalcola i valori mediante l'equazione della retta di regressione
fitted(fit)
calcola le differenze residue tra valore osservato e valore calcolato
residuals(fit)
analisi della varianza per le differenze spiegate dalle x
anova(fit)
matrice di covarianza dell'intercetta e dei coefficienti delle x
vcov(fit)
#
Questi riepiloghi forniscono una analisi dettagliata e puntuale dei dati, che è necessario avere sempre ben presente, ma che nel nostro caso risulta ridondante per cui non la commenteremo.

confronta la regressione a tre variabili con quella a due variabili mediante analisi della varianza
fit1 <- lm(LDL ~ Colesterolo + HDL + Trigliceridi, data=mydata)
fit2 <- lm(LDL ~ Colesterolo + HDL, data=mydata)
anova(fit1, fit2)
#
L'analisi della varianza conferma che la regressione calcolata con tre e quella calcolate con due sole variabili indipendenti differiscono significativamente, pertanto è opportuno utilizzare la prima, più completa, per esprimere i risultati:

```
Analysis of Variance Table

Model 1: LDL ~ Colesterolo + HDL + Trigliceridi
Model 2: LDL ~ Colesterolo + HDL
  Res.Df    RSS Df Sum of Sq      F    Pr(>F)
1   2404  79918
2   2405 196298 -1   -116380 3500.8 < 2.2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

calcola l'importanza relativa di ciascuna variabile indipendente con quattro diversi metodi
library(relaimpo)
myplot <- calc.relimp(fit,type=c("lmg","last","first","pratt"), rela=TRUE)
mostra il grafico

plot(myplot, main="Importanza relativa delle variabili indipendenti")
ripete il calcolo aggiungendo gli intervalli di confidenza dei valori mediante bootstrap
boot <- boot.relimp(fit, b = 1000, type = c("lmg", "last", "first", "pratt"), rank=TRUE, diff=TRUE, rela=TRUE)
booteval.relimp(boot) # mostra i risultati
windows() # apre una nuova finestra
mostra il grafico
plot(booteval.relimp(boot,sort=TRUE), main="Importanza relativa delle variabili indipendenti")
#
Con le prime tre righe di codice viene generato il grafico della importanza relativa delle variabili indipendenti nel determinare la retta di regressione, mettendo a confronto le conclusioni ottenute con quattro metodi di calcolo, che peraltro forniscono risultati molto simili. Con le successive quattro righe di codice viene generato lo stesso identico grafico, questa volta con i limiti di confidenza calcolati mediante bootstrap (**Figura 3.5**).

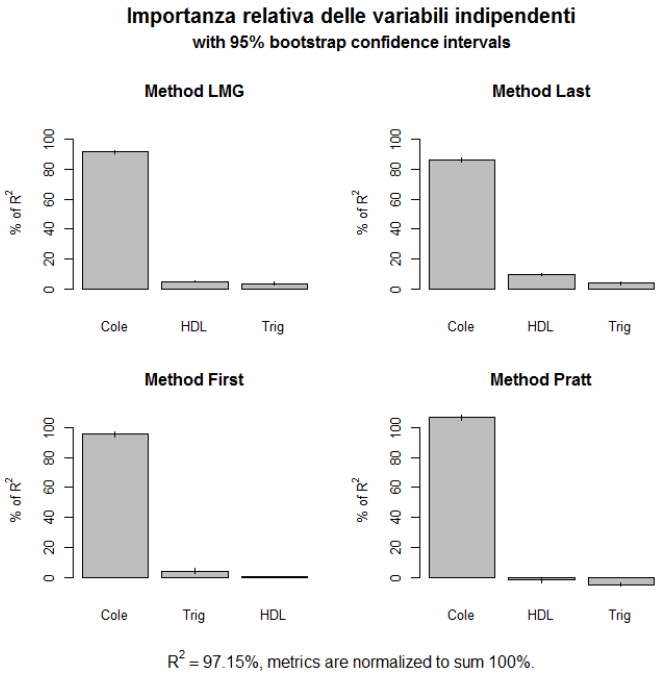


Figura 3.5 Importanza relativa della concentrazione di colesterolo totale (Cole), colesterolo HDL (HDL) e trigliceridi (Trig), variabili indipendenti, nel determinare la concentrazione del colesterolo LDL (variabile dipendente) utilizzando un modello di regressione lineare multipla.

grafico (leverage plot) dell'influenza dei dati sulle conclusioni
windows() # apre una nuova finestra
leveragePlots(fit, ask=FALSE)
#
Mostra il grafico (che qui non viene riportato) dell'influenza delle tre variabili indipendenti (colesterolo, HDL e Trigliceridi) sulle conclusioni (variabile dipendente LDL). Una discussione tecnica sulla costruzione e il significato dei leverage plot la trovate sul sito del software JMP[10].

identificazione degli outliers, inizia caricando la libreria car
library(car)
identificazione degli outliers, valore p di Bonferonni per le osservazioni estreme (outliers)
outlierTest(fit)
#
Il test di Bonferroni viene impiegato per identificare i dati aberranti, i dati che cioè si discostano in modo "eccessivo" dai rimanenti. Il giudizio finale rimane ovviamente a carico di chi ha raccolto i dati, che dovrà analizzarli per capire le ragioni che hanno determinato la differenza statisticamente "poco plausibile"

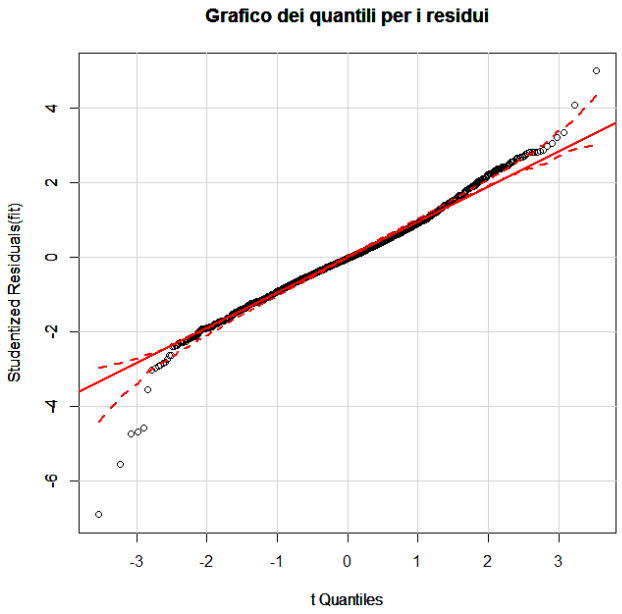
[10] http://www.jmp.com/support/help/Leverage_Plot_Details.shtml

osservata. I dati numero 2398, 2259, 855, 606, 1551 e 753 si discostano significativamente dagli altri:

```
      rstudent unadjusted p-value Bonferonni p
2389 -6.908758         6.2382e-12   1.5022e-08
2259 -5.566038         2.8954e-08   6.9722e-05
855   4.996746         6.2502e-07   1.5050e-03
606  -4.745328         2.2038e-06   5.3068e-03
1551 -4.696544         2.7951e-06   6.7305e-03
753  -4.584611         4.7814e-06   1.1514e-02
```

mostra il grafico dei quantili per i residui studentizzati
windows() # apre una nuova finestra
qqPlot(fit, main="Grafico dei quantili per i residui")
#
Il grafico dei quantili per i residui studentizzati (**Figura 3.6**) conferma anch'esso la presenza di dati che si discostano molto dalla distribuzione attesa.

Figura 3.6 Il confronto tra i quantili campionari e quanto atteso nel caso di una distribuzione gaussiana (linea continua) dimostra che la distribuzione dei dati campionari non è gaussiana.



identificazione degli outliers, grafico della distanza D di Cook, identifica i valori con D > 4/(n-k-1)
windows() # apre una nuova finestra
cutoff <- 4/((nrow(mydata)-length(fit$coefficients)-2))
plot(fit, which=4, cook.levels=cutoff)
#
La distanza D di Cook (**Figura 3.7**) misura l'effetto conseguente alla eliminazione di una specifica osservazione. Viene riportato il numero del dato per quelli che determinano l'effetto maggiore, al fine di consentirne la rapida identificazione: sono confermati tra l'altro i dati numero 606, 753 e 2389 che abbiamo già visto identificati sopra con il test di Bonferroni.

identificazione degli outliers, un altro grafico dell'influenza dei dati sulle conclusioni
windows() # apre una nuova finestra
cutoff <- 4/((nrow(mydata)-length(fit$coefficients)-2))
plot(fit, which=4, cook.levels=cutoff)
influencePlot(fit, main="Grafico dell'influenza dei dati", sub="Dimensione dei cerchi proporzionale alla distanza D di Cook")
#

In questo ulteriore grafico è la dimensione dei cerchi ad essere proporzionale alla distanza di Cook (**Figura 3.8**).

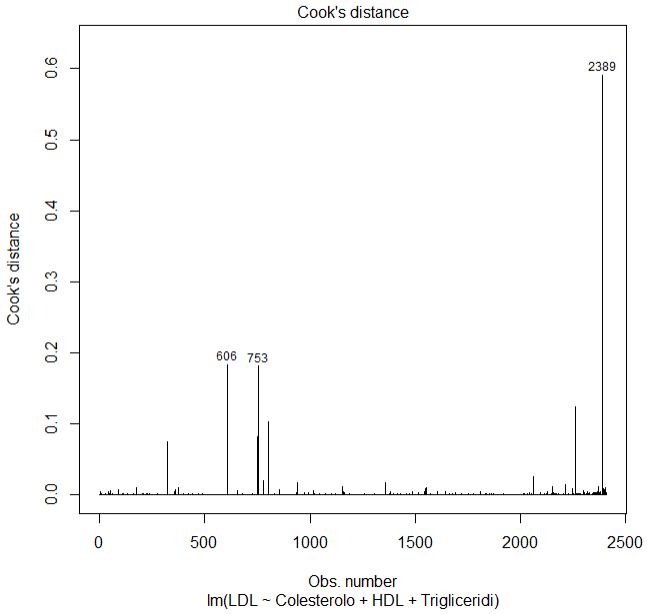


Figura 3.7 Distanza D di Cook. Per i dati la cui eliminazione determina l'effetto maggiore viene riportato il numero dei dato al fine di consentirne la rapida identificazione.

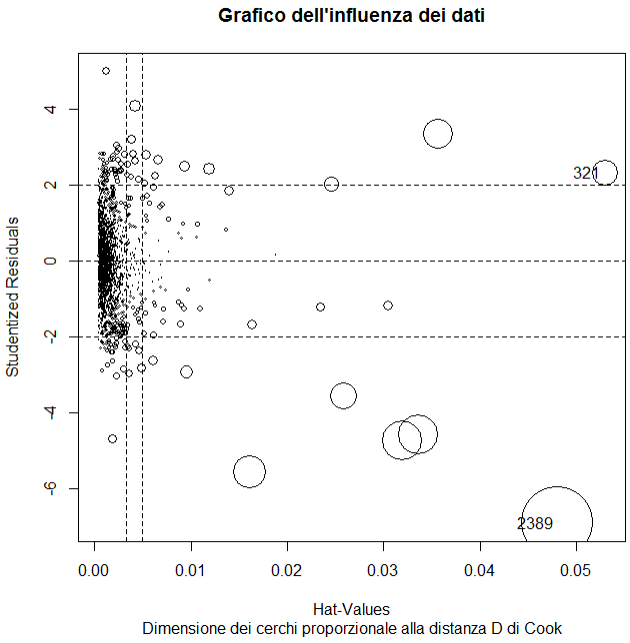


Figura 3.8 In questo grafico dell'influenza dei dati sulle conclusioni la dimensione dei cerchi è proporzionale alla distanza D di Cook.

```
# test per la linearità, grafico di Ceres
windows() # apre una nuova finestra
ceresPlots(fit, ask=FALSE)
#
```

Il grafico di Ceres (**Figura 3.9**), sempre sviluppato da Cook, conferma l'esistenza di una relazione lineare tra colesterolo totale e colesterolo LDL, mentre il colesterolo HDL e i trigliceridi contribuiscono al colesterolo LDL in modo non lineare.

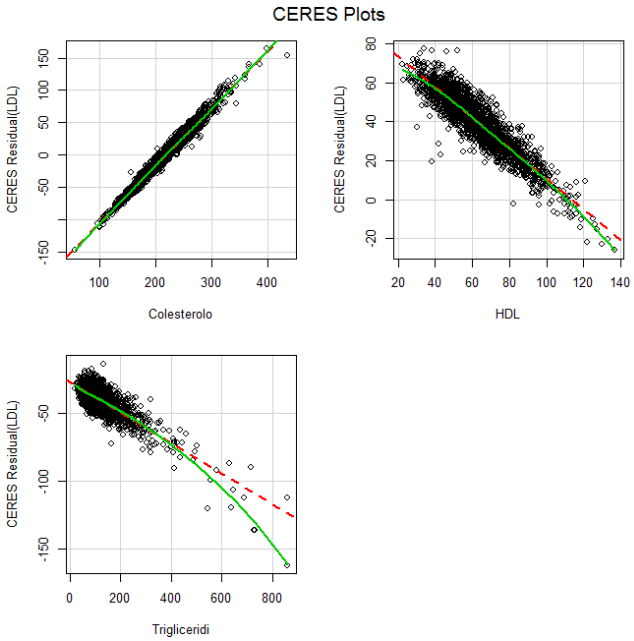


Figura 3.9 Il grafico di Ceres documenta l'esistenza di un contributo al colesterolo LDL (varibile dipendente) lineare da parte del colesterolo totale e non lineare da parte di colesterolo LDL e trigliceridi.

A questo punto, dato che con le molteplici istruzioni **windows()** avete aperto via via nuove finestre che si sono andate sovrapponendo, ciascuna con il proprio grafico, spostate o iconizzate la finestra dell'ultimo grafico per vedere la finestra con il grafico precedente, e così via.

Ed ecco finalmente l'ultimo blocco di codice:

```
# test per la linearità, un test globale per l'assunto di linearità
library(gvlma)
gvmodel <- gvlma(fit)
summary(gvmodel)
#
```

Questo test globale e generalista, pur con tutti i limiti derivanti dal comprimere le conclusioni in pochi indici numerici, conferma che l'assunto di linearità non è soddisfatto:

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

Call:
 gvlma(x = fit)

                     Value  p-value                   Decision
Global Stat        557.2861 0.000000 Assumptions NOT satisfied!
Skewness             0.5274 0.467682    Assumptions acceptable.
Kurtosis           412.5266 0.000000 Assumptions NOT satisfied!
Link Function        7.4492 0.006347 Assumptions NOT satisfied!
Heteroscedasticity 136.7829 0.000000 Assumptions NOT satisfied!
```

La conclusione è ora abbastanza chiara. I dati raccolti presentano due tipi di problemi. Il primo è che alcuni di essi andrebbero rivalutati per capire il significato del loro eccessivo scostamento dai dati rimanenti. Il secondo è che la relazione tra le variabili non è del tutto lineare. Conseguentemente la regressione lineare (in questo caso multipla), ancorché sia statisticamente significativa, deve essere intesa come una approssimazione di "grana media" della relazione che intercorre tra le quattro componenti in esame.

3.5. Analisi multivariata

Analisi multivariata è l'espressione con cui si fa rifermento alle numerose tecniche statistiche che consentono lo studio di sistemi complessi a più variabili[11]. Qui vediamo come utilizzare **R** per l'analisi dei gruppi o cluster analysis.

Scaricate e salvate nella cartella C:\R\ il file Clusterhclust.csv.

id	Calcio	Fosfato	Ossalato	Magnesio
C1	99	81	69	61
C2	78	65	53	43
C3	81	66	38	54
C4	45	23	19	16
C5	44	18	24	19
C6	102	83	72	66
C7	83	68	49	45
C8	74	71	41	57
C9	38	19	22	14
C10	48	14	21	12

Si tratta dei dati relativi alla composizione in calcio, fosfato, ossalato e magnesio di 10 calcoli delle vie urinarie. Il contenuto del file aperto con Excel o con OpenOffice.org Calc vi apparirà come vedete sopra, con i nomi delle variabili nella prima riga, i dati di ciascun caso nelle righe successive, e l'identificativo di ciascun caso nella prima colonna (C1 = calcolo 1, eccetera):

Copiate e incollate nella `Console di R` ed eseguite questo codice:

```
# con la prima riga sono importati i dati
mydata <- read.table("c:/R/Clusterhclust.csv", header=TRUE, sep=";", row.names="id")
# visualizza i dati
mydata
# effettua il clustering gerarchico con il metodo di Ward
d <- dist(mydata, method = "euclidean") # matrice delle distanze euclidee
fit <- hclust(d, method="ward")
plot(fit, main="Cluster analysis: dendrogramma", xlab="Differenti calcoli delle vie urinarie analizzati", ylab="Distanza nella composizione") # traccia il dendrogramma
groups <- cutree(fit, k=3) # divide in 3 cluster principali, valore k da cambiare al bisogno
# evidenzia i 3 cluster, attenzione a digitare invio nella Console di R
rect.hclust(fit, k=3, border="red")
#
```

Al termine vi comparirà una finestra con il dendrogramma (**Figura 3.10**).

In ascisse sono riportati i singoli calcoli, in ordinate la distanza alla quale questi vanno via via confluendo per "somiglianza" in cluster sempre più estesi. Minore la è distanza alla quale avviene la confluenza, maggiore è la somiglianza nella composizione dei calcoli e dei successivi cluster. Non esiste un valore soglia della distanza alla quale fermare il processo di raggruppamento per somiglianza dei calcoli, anche se qui sembra ragionevole affermare che (i) si vanno formando tre gruppi/cluster di calcoli e (ii) la composizione chimica dei calcoli C1 e C6 è più simile a quella dei calcoli C2,C7, C3 e C(che a quella dei calcoli C10, C9, C4

[11] http://www.treccani.it/enciclopedia/analisi-multivariata/

e C5.

Potete facilmente riutilizzare il codice per le vostre specifiche esigenze con queste semplici modifiche:
→ sostituire il nome del file **"c:/R/Clusterhclust.csv"** con quello del vostro file;
→ controllare il separatore di campo usato ed eventualmente correggere opportunamente il punto e virgola in **sep=";"**;
→ togliere per intero **,row.names="id"** se il votro file non contiene una variabile **id** con gli identificativi univoci dei vostri casi;
→ adattare il testo delle legende **main=""**, **xlab=""** e **ylab=""**;
→ cambiare nel comando **groups <- cutree(fit, k=3)** il valore di **k**, che indica quanti sono i gruppi principali identificati dalla cluster analysis (sta a voi valutarlo in base alle distanze verticali dei dendrogrammi);
→ adattare il codice **rect.hclust(fit, k=3, border="red")** al valore di **k** prescelto ed eventualmente cambiare il colore del bordo (**border="red"**) dei riquadri che evidenziano i gruppi principali identificati dalla cluster analysis.

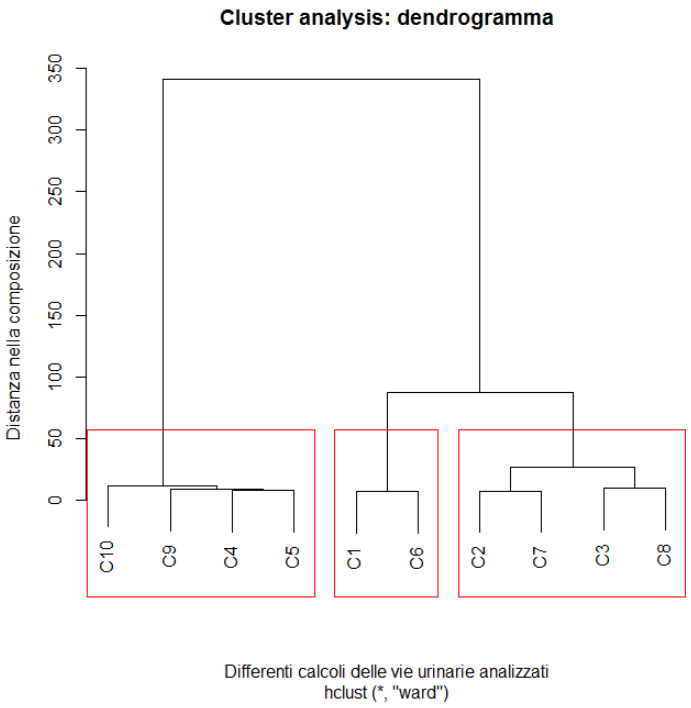


Figura 3.10 Dendrogramma che illustra la confluenza dei calcoli della vie urinarie, in termini di composizione chimica, in tre gruppi (cluster) principali.

Ora scaricate e salvate nella cartella C:\R\ il file Clusterpvclust.csv. Il contenuto del file vi apparirà così, con i nomi delle variabili nella prima riga, i dati di ciascun caso nelle righe successive, e l'identificativo di ciascun caso nella prima colonna (C1 = calcolo 1, eccetera):

id	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10
Calcio	99	78	81	45	44	102	83	74	38	48
Fosfato	81	65	66	23	18	83	68	71	19	14
Ossalato	69	53	38	19	24	72	49	41	22	21
Magnesio	61	43	54	16	19	66	45	57	14	12

Si tratta sempre degli stessi dati relativi alla composizione di 10 calcoli delle vie urinarie visti in precedenza, ma attenzione: righe e colonne sono state scambiate tra loro. La trasposizione tra righe e colonne si rende necessaria per utilizzare la libreria **pvclust** (che, se non lo avete ancora fatto, dovete scaricare dal CRAN

prima di eseguire l'esempio). Il metodo di clustering gerarchico applicato prevede in più, rispetto al caso precedente, il calcolo mediante bootstrap dei valori di probabilità p che caratterizzano i cluster formati in due modi differenti: come "Bootstrap Probability values" (indicati con BP) e come "Approximately Unbiased probability values" (indicati con AU). Per i dettagli vedere la documentazione della libreria.

Copiate e incollate nella Console di R ed eseguite questo codice:

```
# con la prima riga sono importati i dati
mydata <- read.table("c:/R/Clusterpvclust.csv", header=TRUE, sep=";", row.names="id")
# visualizza mydata
mydata
# carica la libreria necessaria
library(pvclust)
# clustering gerarchico con il metodo di Ward e i valori di p calcolati mediante bootstrap
fit <- pvclust(mydata, method.hclust="ward", method.dist="euclidean")
# traccia il dendrogramma con i valori di p
plot(fit, main="Cluster analysis: dendrogramma", xlab="Differenti calcoli delle vie urinarie analizzati",
ylab="Distanza nella composizione", print.pv=TRUE, print.num=TRUE)
# evidenzia i cluster fortemente supportati dai dati
pvrect(fit, alpha=0.95, pv="au", type="geq", max.only=TRUE)
# legenda: BP = bootstrap probability, AU = approximately unbiased, p-values = probability values
#
```

Anche in questo caso al termine vi comparirà una finestra con il dendrogramma (**Figura 3.11**).

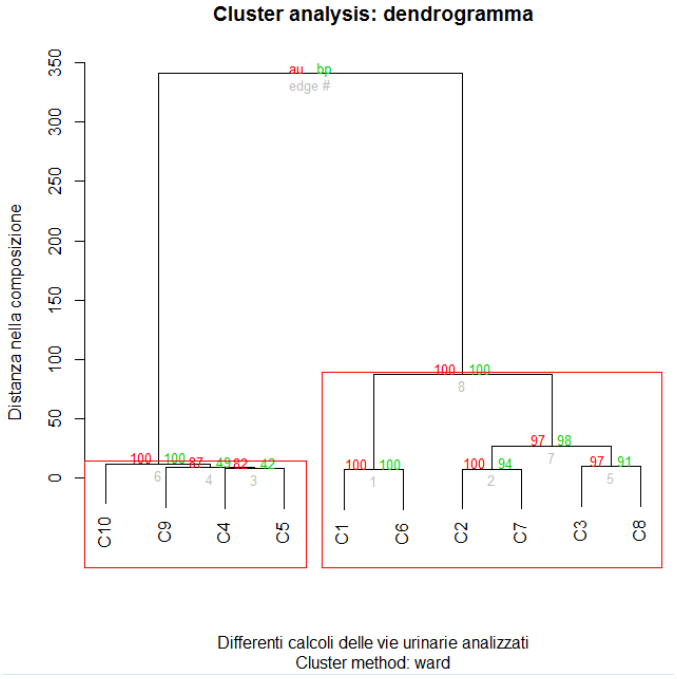


Figura 3.11 Dendrogramma che illustra la confluenza dei calcoli della vie urinarie, in termini di composizione chimica, in due gruppi (cluster) principali.

Il fatto che i cluster siano due (**Figura 3.10**) o tre (**Figura 3.11**) e possano dipendere dal metodo adottato non è particolarmente preoccupante. La cluster analysis è un metodo per l'analisi esplorativa dei dati, e le conclusioni non devono essere viste come qualcosa di irreversibile (e quindi nel caso specifico contradditorio). Il metodo di clusterizzazione adottato deve essere integrato da una modellizzazione che includa (nel caso specifico) il processo di formazione dei calcoli delle vie urinarie, le patologie che ne sono alla base, i fattori che ne scatenano la formazione, al fine di collegare la somiglianza statistica dei calcoli agli elementi che determinano la loro formazione e la loro confluenza in gruppi significativi.

3.6 Statistica bayesiana

L'applicazione del teorema di Bayes alla diagnostica di laboratorio è estesamente trattata nel mio sito tanto da non richiedere ulteriori considerazioni. Mi limito ad un esempio che consente di illustrare come utilizzare R per la valutazione di un test diagnostico, ricordando che trovate nella parte relativa alle curve ROC un indispensabile complemento.

L'esempio è tratto da Scott IA, Greenberg PB, Poole PJ. Cautionary tales in the clinical interpretation of studies of diagnostic tests. Internal Medicine Journal 38 (2008) 120–129. Un nuovo test diagnostico è stato provato su 1586 pazienti. Di 744 pazienti che avevano la malattia, 670 sono risultati positivi al test. Di 842 pazienti che non avevano la malattia, 640 sono risultati negativi al test.

Possiamo riportate i dati forniti in una tabella e completarla con Excel o OpenOffice Calc:

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

Se ne deduce pertanto che erano 670 i veri positivi (VP), 202 i falsi positivi (FP), 74 i falsi negativi (FN) e infine 640 i veri negativi (VN):

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

Il codice R è estremamente conciso se si utilizza la libreria **epiR**, che ovviamente deve essere preventivamente scaricata dal CRAN:

```
# carica la libreria necessaria
library(epiR)
# i dati sono immessi direttamente dalla Console di R
data <- as.table(matrix(c(670,202,74,640), nrow = 2, byrow = TRUE))
# sono calcolate e mostrate tutte le statistiche
epi.tests(data, conf.level = 0.95, verbose = TRUE)
#
```

Il fatto molto interessante è che le grandezze calcolate sono riportate ciascuna con il rispettivo intervallo di confidenza al 95%.

```
$aprev
        est     lower     upper
1 0.5498108 0.5249373 0.5744996

$tprev
        est     lower     upper
1 0.4691047 0.4443055 0.4940184

$se
```

```
        est     lower     upper
1 0.9005376 0.8767462 0.9210923

$sp
       est     lower     upper
1 0.760095 0.7297765 0.7885803

$diag.acc
        est     lower     upper
1 0.8259773 0.8064049 0.8443346

$diag.or
       est    lower    upper
1 28.68611 21.51819 38.24174

$nnd
       est  lower    upper
1 1.513701 1.4091 1.648743

$youden
        est     lower     upper
1 0.6606326 0.6065226 0.7096726

$ppv
        est     lower     upper
1 0.7683486 0.7388926 0.7959784

$npv
        est     lower     upper
1 0.8963585 0.8716393 0.9177402

$plr
       est    lower    upper
1 3.753726 3.320688 4.243235

$nlr
        est     lower     upper
1 0.1308552 0.1050643 0.1629771
```

Ecco qui le grandezze calcolate con **R**, il loro significato, le formule con cui sono calcolate, e il risultato numerico ottenuto (per semplicità l'intervallo di confidenza viene omesso):

$aprev
prevalenza apparente, soggetti con il test positivo = (VP+FP)/(VP+FP+FN+VN) = 0.5498108
$tprev
prevalenza reale, soggetti con la malattia = (VP+FN)/(VP+FP+FN+VN) = 0.4691047
$se
sensibilità, positività nei malati = VP/(VP+FN) = 0.9005376
$sp
specificità, negatività nei sani = VN/(VN+FP) = 0.760095
$diag.acc
accuratezza diagnostica = (VP+VN)/(VP+FP+FN+VN) = 0.8259773
$diag.or
odd ratio = rapporto LR+/LR- = 28.68611
$nnd
numero necessario per la diagnosi = 1/(sensibilità - (1 - specificità) = 1.513701

$youden
indice di Youden = sensibilità + specificità - 1 = 0.6606326
$ppv
valore predittivo di un test positivo = VP/(VP+FP) = 0.7683486
$npv
valore predittivo di un test negativo = VN/(VN+FN) = 0.8963585
$plr
rapporto di verosimiglianza LR+ per un test positivo = (VP/(VP+FN))/(FP/(FP+VN)) = 3.753726
$nlr
rapporto di verosimiglianza LR- per un test negativo = (FN/(VP+FN))/(VN/(FP+VN)) = 0.1308552

4. R funzioni grafiche

Le funzioni grafiche di **R** sono trattate assumendo che abbiate familiarizzato adeguatamente con tutti gli argomenti generali contenuti nel capitolo 2. Le indicazioni qui riportate fanno riferimento alla versione di **R** per Windows. Per MacOS X vedere sempre nel capitolo 2 come utilizzare i dati e gli script con MacOS X.

Potete imparare a utilizzare le funzioni grafiche di **R** eseguendo il codice che trovate nelle pagine seguenti con i dati di esempio forniti come file `.csv` generati con Excel e OpenOffice.org Calc che devono essere scaricati e installati sul PC nella cartella `C:\R\`. Se li installate in una cartella diversa, dovete modificare opportunamente il codice **read.table("c:/R/** specificando il nuovo percorso nel quale avete installato i file. Per eseguire il codice **R** dovete semplicemente selezionarlo nella pagina web includendo il cancelletto # che chiude ogni blocco di codice, copiarlo, e incollarlo nella `Console di R`.

4.1. Istogrammi

Scaricate e salvate nella cartella C:\R\ il file Densplot.csv. Il file contiene per 1000 soggetti il sesso (M o F), l'età (in anni) e la concentrazione del colesterolo nel siero (in mg/dL). Aperto con Excel o con OpenOffice.org Calc appare così:

Sesso	Eta	Colest
M	13	172
F	14	132
M	14	176
F	15	156
F	16	190
……	……	……

Aperto con un editor di testo come il `Blocco note di Windows` vi apparirà così, con i nomi delle variabili nella prima riga, i dati dei singoli casi nelle righe successive, e come separatore di campo il punto e virgola (;):

```
Sesso;Eta;Colest
M;13;172
F;14;132
M;14;176
F;15;156
F;16;190
...;...;...
```

In realtà è questa la modalità (formato ASCII) con la quale i dati sono stati salvati nel file. Excel e OpenOffice Calc caricano questi dati una riga alla volta, riconoscono e utilizzano il separatore `;` per separare i campi riportati in ciascuna riga, e formattano le variabili nelle colonne, righe e celle tipiche del tabellone elettronico. Aggiungendo poi la possibilità di manipolarle tipiche del tabellone elettronico.

Copiate e incollate nella `Console di R` questo codice ed eseguitelo un blocco alla volta per familiarizzare con il linguaggio soffermandovi sui singoli passaggi:

```
# con la prima riga sono importati i dati
mydata <- read.table("c:/R/Densplot.csv", header=TRUE, sep=";")
# visualizza i dati nella Console di R
mydata
# traccia un istogramma semplice dei dati
hist(mydata$Colest, main = "Istogramma semplice", xlab = "Colesterolo totale in mg/dL", ylab = "Frequenza")
#
```

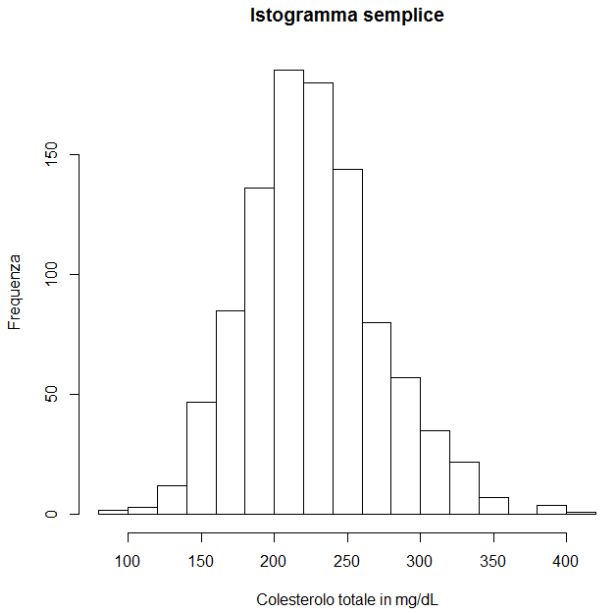


Figura 4.1 Istogramma semplice della distribuzione della concentrazione del colesterolo nel siero.

Se l'istogramma semplice non vi piace (**Figura 4.1**), potete pensare di realizzare un istogramma colorato (**Figura 4.2**) nel quale gestire anche il numero delle classi in cui suddividere i dati con questo codice:

```
#
windows() # apre una nuova finestra
# traccia un istogramma colorato, con 20 classi
hist(mydata$Colest, breaks=20, col="red", main = "Istogramma colorato con 20 classi", xlab = "Colesterolo totale in mg/dL", ylab = "Frequenza")
#
```

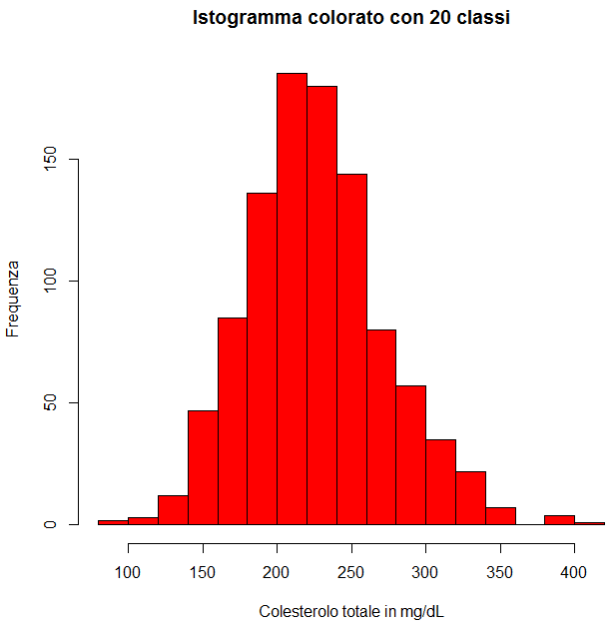


Figura 4.2 Istogramma della distribuzione della concentrazione del colesterolo nel siero. Dati suddivisi in 20 classi.

Con l'argomento **col="red"** l'istogramma viene colorato in rosso, mentre con l'argomento **breaks=20** viene definito il numero delle classi desiderate.

Provate a cambiarlo per vedere l'effetto risultante, per esempio provate con 100 classi (**Figura 4.3**):

```
#
windows() # apre una nuova finestra
hist(mydata$Colest, breaks=100, col="red", main = "Istogramma colorato con 100 classi", xlab = "Colesterolo totale in mg/dL", ylab = "Frequenza")
#
```

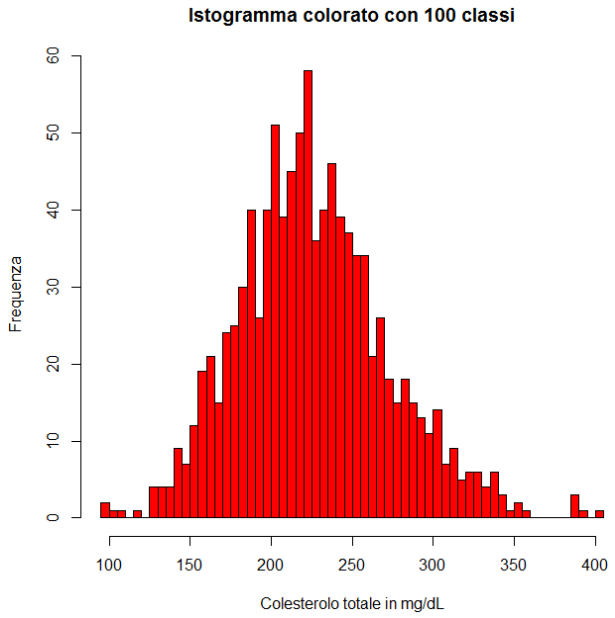


Figura 4.3 Istogramma della distribuzione della concentrazione del colesterolo nel siero. Dati suddivisi in 100 classi.

Potete anche tracciare l'istogramma e sovraimporre ad esso la distribuzione gaussiana teorica (**Figura 4.4**):

```
# traccia l'istogramma e vi sovrappone la curva gaussiana corrispondente
windows() # apre una nuova finestra
x <- mydata$Colest
h <-hist(x, breaks=33, col="red", main = "Istogramma con curva gaussiana", xlab="Colesterolo totale in mg/dL", ylab = "Frequenza")
xfit<-seq(min(x),max(x),length=40)
yfit<-dnorm(xfit,mean=mean(x),sd=sd(x))
yfit <- yfit*diff(h$mids[1:2])*length(x)
lines(xfit, yfit, col="blue", lwd=2)
#
```

In questo caso i dati sono stati suddivisi in 33 classi, seguendo la regola per cui il numero delle classi dovrebbe essere uguale alla radice quadrata del numero dei dati (qui abbiamo 1000 dati, la cui radice quadrata è circa 33). A questo punto, dato che con le tre istruzioni **windows()** avete aperto tre nuove finestre, avrete un totale di quattro finestre, con altrettanti grafici, sovrapposte. Spostate o iconizzate la finestra dell'ultimo grafico per vedere la finestra con il grafico precedente, e così via.

Il codice degli esempi riportati sopra per rappresentare gli istogrammi può essere interamente riutilizzato per le vostre specifiche esigenze, ricordando che dovete semplicemente:
→ sostituire il nome del file **"c:/R/Densplot.csv"** con quello del vostro file;
→ controllare il separatore di campo usato ed eventualmente correggere opportunamente il punto e virgola in **sep=";"**;

→ adattare il testo delle legende **main=””** (il titolo), **xlab=””** (asse x delle ascisse) e **ylab=””** (asse y delle ordinate);
→ cambiare il numero delle classi in cui suddividere i dati (**breaks=**)
→ personalizzare i colori del riempimento (**col=""**) e dei bordi (**border=""**).

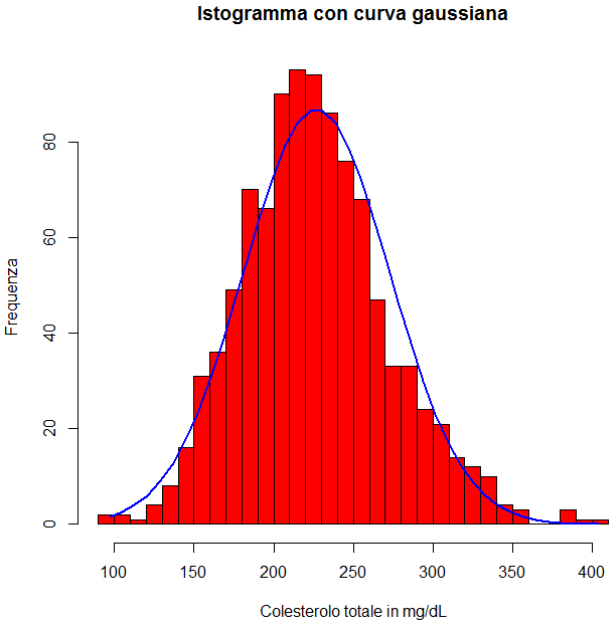


Figura 4.4 Istogramma della distribuzione della concentrazione del colesterolo nel siero, con sovraimposta la distribuzione gaussiana teorica corrispondente.

La tabella dei colori di **R** è riportata nell'appendice A1.

4.2. Kernel density plot

Il diagramma che rappresenta la distribuzione della densità delle osservazioni (kernel density plot) è una modalità di rappresentazione piuttosto interessante, che può essere utilizzata in alternativa al tradizionale istogramma. Digitando `kernel density plot` nella casella di ricerca di Google trovate una moltitudine di riferimenti tecnici su questo tema.

Scaricate e salvate nella cartella C:\R\ il file Densplot.csv. Si tratta dello stesso file utilizzato per gli istogrammi, e contiene sesso (M o F), età (in anni) e concentrazione del colesterolo nel siero (in mg/dL) di 1000 soggetti. Aperto con Excel o con OpenOffice.org Calc appare così:

Sesso	Eta	Colest
M	13	172
F	14	132
M	14	176
F	15	156
F	16	190
……	……	……

Copiate e incollate nella `Console di R` ed eseguite questo codice un blocco alla volta per familiarizzare con il linguaggio soffermandovi sui singoli passaggi:

```
# importa i dati in R
mydata <- read.table("c:/R/Densplot.csv", header=TRUE, sep=";")
# traccia il kernel density plot
```

```
d <- density(mydata$Colest) # dati di densità
plot(d, main = "Kernel density plot", xlab="Colesterolo totale in mg/dL", ylab = "Densità")
#
```

Ecco come appare il kernel density plot (**Figura 4.5**).

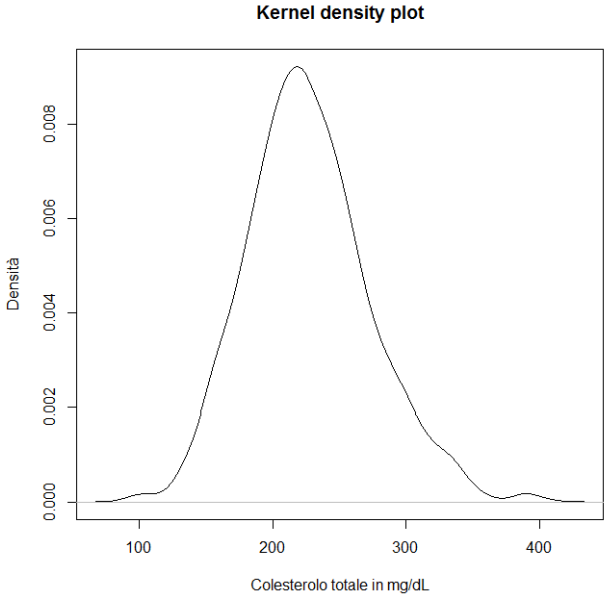


Figura 4.5 Kernel density plot semplice della distribuzione della concentrazione del colesterolo nel siero.

Potete migliorarne l'aspetto aggiungendo del colore:

```
# traccia il kernel density plot colorato
windows() # apre una nuova finestra
d <- density(mydata$Colest) # dati di densità
plot(d, main = "Kernel density plot colorato", xlab="Colesterolo totale in mg/dL", ylab = "Densità")
polygon(d, col="red", border="blue") # colora
#
```

Ecco come appare il kernel density plot, con un bordo di colore blu e il riempimento di colore rosso (**Figura 4.6**).

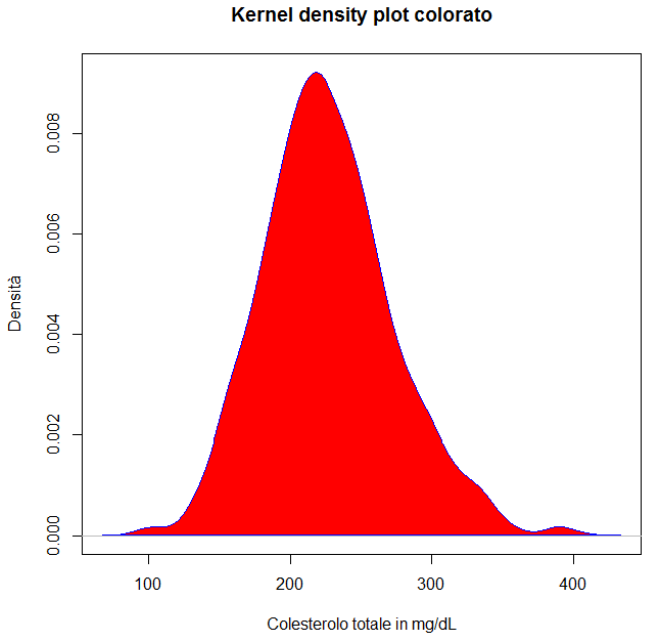


Figura 4.6 Kernel density plot colorato della distribuzione della concentrazione del colesterolo nel siero.

Dato che con l'istruzione **windows()** avete aperto una nuova finestra, spostatela o iconizzatela per vedere la finestra con il grafico precedente.

Ora copiate e incollate nella `Console di R` ed eseguite questo codice:

```
# importa i dati in R
mydata <- read.table("c:/R/Densplot.csv", header=TRUE, sep=";")
# traccia kernel density plot sovrapposti
# al termine fate click con il tasto sinistro del mouse nel punto in cui volete fare comparire la legenda
library(sm)
attach(mydata)
Sesso.f <- factor(Sesso, levels= c("F","M"), labels = c("Donna", "Uomo")) # crea la legenda
sm.density.compare(Colest, Sesso, xlab="Colesterolo totale in mg/dL", ylab="Densità") # traccia il grafico
title(main="Distribuzione del colesterolo totale per sesso") # aggiunge il titolo
colfill<-c(2:(2+length(levels(Sesso.f)))) # posiziona la legenda
legend(locator(1), levels(Sesso.f), fill=colfill) # posiziona la legenda
#
```

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 per farla comparire. Ed ecco il risultato definitivo (**Figura 4.7**).

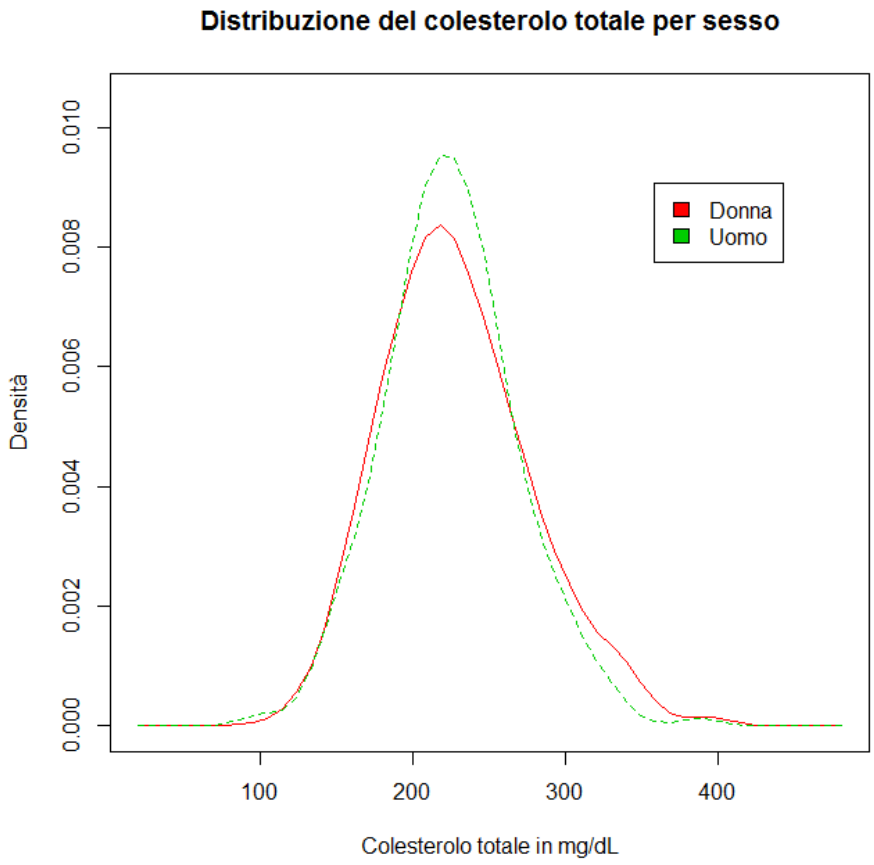


Figura 4.7 Kernel density plot sovrapposti della distribuzione della concentrazione del colesterolo nel siero in soggetti dei due sessi.

Se non siete soddisfatti della posizione delle legenda, sappiate che **R** non consente di muoverla. Dovete

rieseguire l'intero script e fare nuovamente click con il tasto sinistro del mouse nel punto in cui volete posizionare la legenda.

4.3. Box & whiskers plot

I box & whiskers plot (diagrammi a scatola e baffi) consentono di confrontare in modo immediato la distribuzione di più variabili. La scatola rappresenta la mediana (al centro), il primo quartile (margine inferiore della scatola) e il terzo quartile (margine superiore della scatola). La scatola include pertanto il 50% delle osservazioni. I baffi possono includono tutti i dati osservati oppure lasciare all'esterno i dati che presentano uno scostamento eccessivo (outliers). I box & whiskers plot forniscono una rappresentazione non-parametrica della distribuzione dei dati.

Scaricate e salvate nella cartella C:\R\ il file Boxplot.csv. I dati contenuti nel file sono i valori di concentrazione delle IgA (in g/L) in un gruppo di soggetti sani (Controlli) e di soggetti con cirrosi alcolica (AC), epatite cronica attiva (CAH), epatite cronica persistente (CPH), epatite alcolica non cirrotica (NCAH). Il contenuto del file aperto con un editor di testo come il Blocco note di Windows vi apparirà così:

```
Diagnosi;IgA
Controlli;1.22
......
Controlli;2.37
NCAH;7.44
......
NCAH;3.75
CPH;2.45
.....
CPH;3.47
CAH;2.35
......
CAH;2.93
AC;3.51
......
AC;6.22
```

Nella prima riga sono riportati i nomi delle due variabili contenute nel file, rispettivamente la diagnosi clinica (Diagnosi) e la concentrazione delle IgA in mg/dL (IgA). Nelle righe successive sono riportati i valori delle due variabili per ciascuno dei casi osservati. Come separatore di campo viene utilizzato il punto e virgola (;).

Copiate e incollate nella Console di R ed eseguite questo codice:

```
# con la prima riga sono importati i dati
mydata <- read.table("c:/R/Boxplot.csv", header=TRUE, sep=";")
# con la seconda riga sono tracciati i boxplot delle IgA per ciascuna diagnosi
boxplot(IgA~Diagnosi, data=mydata, main="IgA nelle malattie croniche del fegato", xlab="Diagnosi clinica", ylab="IgA in g/L", notch=FALSE, outline=TRUE, col="yellow")
#
```

Come vedete dal codice non è necessario specificare il numero di box & whiskers plot da tracciare. Il numero viene desunto direttamente dai dati, aggregando i valori di IgA per Diagnosi (**IgA~Diagnosi**) e quindi in questo caso è uguale al numero delle diverse diagnosi. Il parametro **outline=TRUE** indica di lasciare all'esterno dei baffi come punti separati i dati che presentano uno scostamento eccessivo (outliers) (**Figura 4.8 a sinistra**).

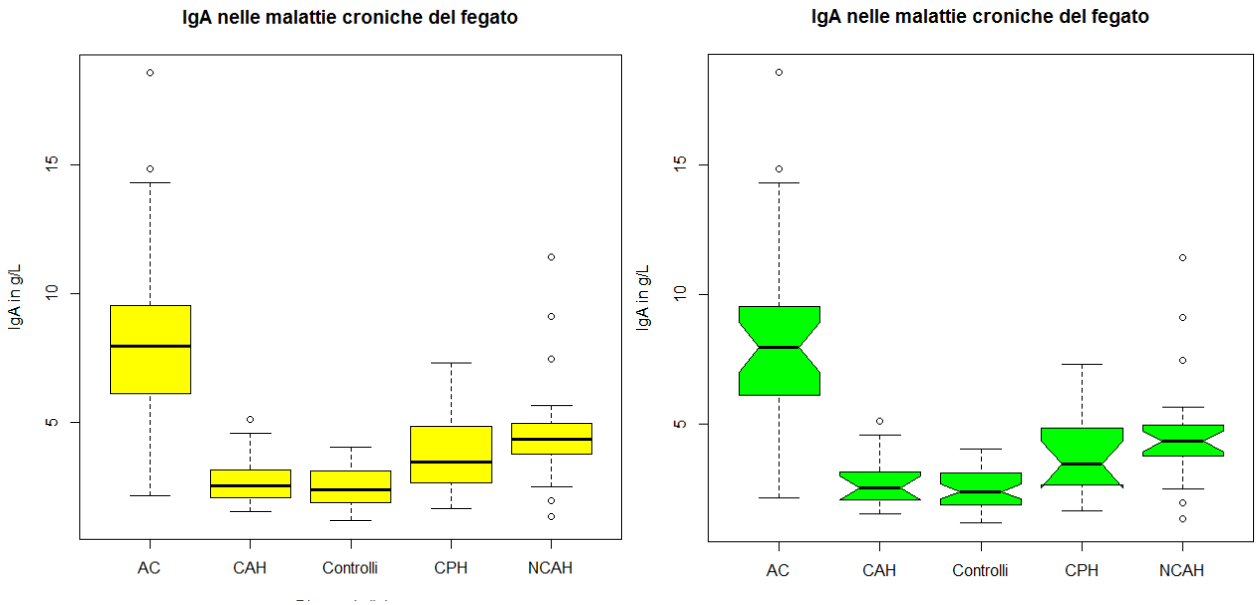


Figura 4.8 Box & whiskers plot della distribuzione delle IgA (in mg/dL) in alcune malattie epatiche, a sinistra (in giallo) senza incisure e a destra (in verde) con le incisure che indicano la significatività della differenza tra le mediane.

Ecco cosa accade riprendendo lo stesso codice ma questa volta con il parametro **notch=TRUE**:

```
# sono tracciati i boxplot delle IgA per ciascuna diagnosi con i notch (incisure)
windows() # apre una nuova finestra
boxplot(IgA~Diagnosi, data=mydata, main="IgA nelle malattie croniche del fegato",  xlab="Diagnosi clinica", ylab="IgA in g/L", notch=TRUE, col="green")
#
```

In questo caso sono tracciati i boxplot delle IgA per ciascuna diagnosi con una incisura (**notch=TRUE**) che rappresenta i limiti di confidenza al 95% della mediana (**Figura 4.8 a destra**). Questo corrisponde ad un test per la significatività della differenza tra le mediane. Se le incisure di due boxplot non si sovrappongono la mediana delle due distribuzioni è significativamente diversa.

Il fatto interessante è che vediamo qui utilizzata una rappresentazione grafica per effettuare un test statistico (un confronto tra mediane). Notate anche il messaggio che compare nella `Console di R`:

```
Warning message:
In bxp(list(stats = c(2.14, 6.115, 7.95, 9.55, 14.31, 1.51, 2.065,  :
  some notches went outside hinges ('box'): maybe set notch=FALSE
```

Il messaggio avverte che in alcuni casi le incisure sono uscite dai bordi della scatola (osservate i boxplot della CAH e della CPH). In questi casi il problema è determinato dal fatto che il numero delle osservazioni troppo ridotto determina un livello di incertezza che si estende al di la delle osservazioni. Vi sono solamente due modi per superare questo problema: rinunciare a trarre delle conclusioni da questi casi, o aumentare adeguatamente il numero delle osservazioni.

Ricordate infine che con l'istruzione **windows()** avete aperto una nuova finestra, quindi avrete due finestre, con altrettanti grafici, sovrapposte. Spostate o iconizzate la finestra del grafico con le incisure (box verdi) per vedere la finestra con il grafico precedente, quello senza incisure (box gialli).

4.4. Scatter plot

Potremmo chiamarlo semplicemente diagramma cartesiano, o diagramma di dispersione o grafico di dispersione (denominazione quest'ultima che rispecchia alla lettera quella inglese). Ma manteniamo anche questa volta per omogeneità la denominazione originale adottata in **R**.

Scaricate e salvate nella cartella C:\R\ il file Scatterplot.csv. Il file aperto con Excel o con OpenOffice.org Calc appare così, con una struttura molto tradizionale, una variabile per ogni colonna, nella prima riga i nomi delle variabili, nelle righe successive i loro valori:

GR	RGO	HB	HCT	HBA2	MCV	HBF	MCH	RDW	FERRO
4.90	97	13.3	40.6	1.8	82.8	0.6	27.1	17.3	106
4.66	81	10.8	34.3	2.6	73.6	1.6	23.2	21.5	148
5.43	57	11.5	36.1	4.8	66.5	2.5	21.1	21.0	104
5.41	63	10.8	39.7	2.5	73.4	1.8	20.0	19.9	74
4.94	60	10.4	32.3	1.4	65.0	0.7	21.1	23.7	17
4.30	97	12.1	35.8	1.9	83.3	0.7	28.2	18.3	43
……	……	……	……	……	……	……	……	……	……

Le variabili contenute nel file sono la concentrazione degli eritrociti (GR) espressa in 10^12/L, la resistenza globulare osmotica (RGO) in %, la concentrazione dell'emoglobina (HB), in g/dL, l'ematocrito (HCT) in %, l'emoglobina A2 (HBA2) espressa in % dell'emoglobina totale, il volume globulare medio (MCV) in fL, l'emoglobina F (HBF) espressa in % dell'emoglobina totale, l'emoglobina corpuscolare media (MCH) in pg, l'ampiezza della distribuzione dei globuli rossi (RDW) espressa in % (come coefficiente di variazione), e infine la concentrazione del ferro nel siero in µg/dL, misurati in 643 soggetti che includevano controlli sani, soggetti portatori di beta-talassemia, portatori di alfa-talassemia, e soggetti con anemia sideropenica.

Da notare che sono utilizzate la libreria **car** e la libreria **gclus** che, se non lo avete ancora fatto, dovete scaricare dal CRAN prima di eseguire l'esempio (in caso contrario si verificherà un errore nell'esecuzione del codice laddove è previsto l'utilizzo delle librerie). Copiate e incollate nella `Console di R` questo codice ed eseguitelo una blocco alla volta per familiarizzare con il linguaggio soffermandovi sui singoli passaggi:

```
# innanzitutto importiamo i dati
mydata <- read.table("c:/R/Scatterplot.csv", header=TRUE, sep=";")
# visualizziamo i dati
mydata
# traccia uno scatter plot semplice
attach(mydata)
plot(HBA2, FERRO, main="Scatter plot semplice con cerchi pieni", xlab="Emoglobina A2, % ", ylab="Ferro, µg/dL ", pch=19)
#
```

Viene prodotto un semplice diagramma cartesiano che rappresenta la concentrazione del ferro in funzione della concentrazione di emoglobina A2 (**Figura 4.9 a sinistra**). I singoli punti sono rappresentati mediante cerchi pieni.

```
# cambia lo stile dei punti
windows() # apre una nuova finestra
plot(HBA2, FERRO, main="Scatter plot semplice con cerchi vuoti", xlab="Emoglobina A2, % ", ylab="Ferro, µg/dL ", pch=1)
#
```

Viene prodotto lo stesso diagramma cartesiano del caso precedente, ma questa volta il simbolo per

rappresentare i dati è rappresentato da un cerchio vuoto (**Figura 4.9 a destra**).

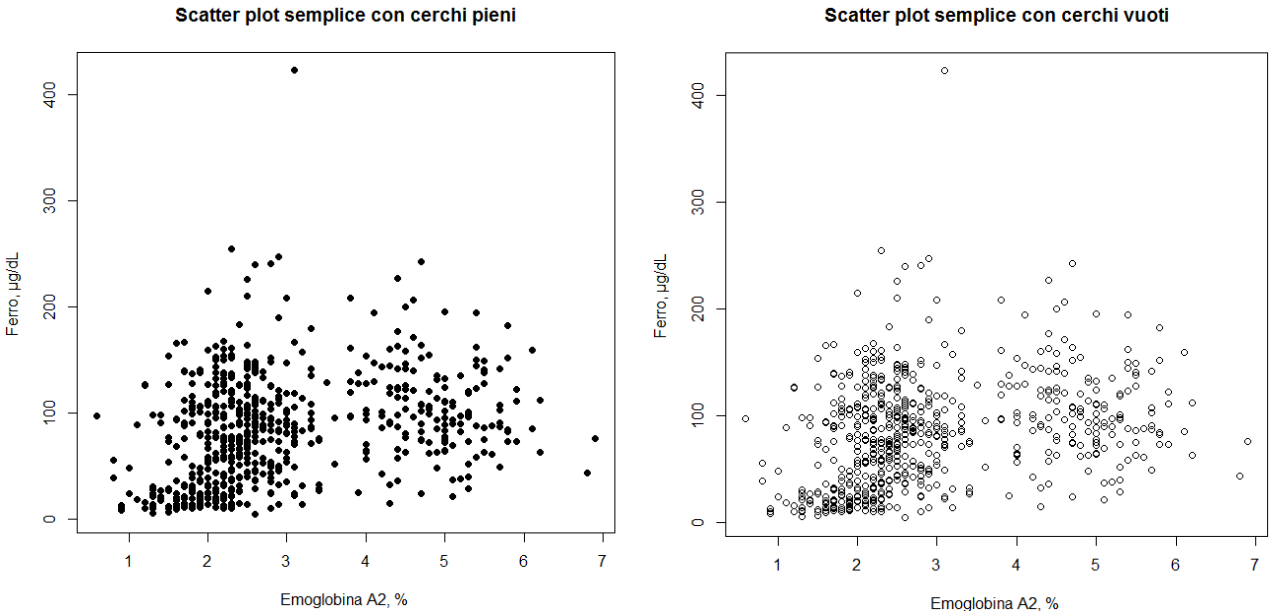


Figura 4.9 Due scatter plot della concentrazione dell'emoglobina A2 negli eritrociti (in % in ascisse) e del ferro nel siero (in ordinate in µg/dL). A sinsitra i punti sono rapresentati con cerchi pieni, a destra con cerchi vuoti.

Le potenzialità di **R** nella rappresentazione di scatter plot vanno però ben oltre:

una sola riga di codice traccia lo scatterplot con la matrice completa di tutte le variabili
windows() # apre una nuova finestra
pairs(~GR+RGO+HB+HCT+HBA2+MCV+HBF+MCH+RDW+FERRO, data=mydata, main="Matrice degli scatter plot di tutte le variabili")
#
Viene generata la matrice degli scatter plot incrociando tra di loro tutte le variabili, e per ogni coppia di variabili viene effettuata una duplice rappresentazione, prima con l'una e poi con l'altra variabile in ascisse (**Figura 4.10**):

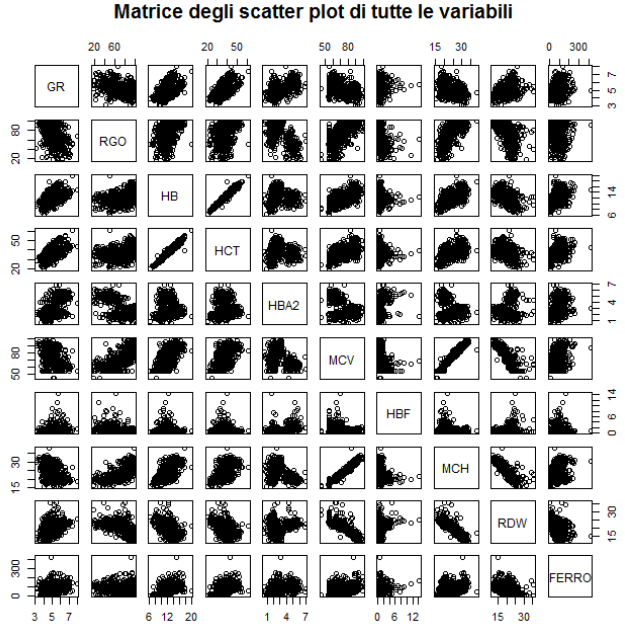


Figura 4.10 Matrice degli scatter plot di tutte le variabili contenute nel file di dati Scatterplot.csv che include controlli sani, soggetti portatori di beta-talassemia, portatori di alfa-talassemia, e soggetti con anemia sideropenica.

come al punto precedente, ma con matrice parziale limitata a quattro variabili
windows() # apre una nuova finestra
pairs(~GR+HBA2+MCV+MCH,data=mydata, main="Matrice degli scatter plot di GR, HBA2, MCV, MCH")
#
La matrice degli scatter plot viene limitata a eritrociti (GR), emoglobina A2 (HBA2), volume globulare medio (MCV) ed emoglobina corpuscolare media (MCH) (**Figura 4.11**):

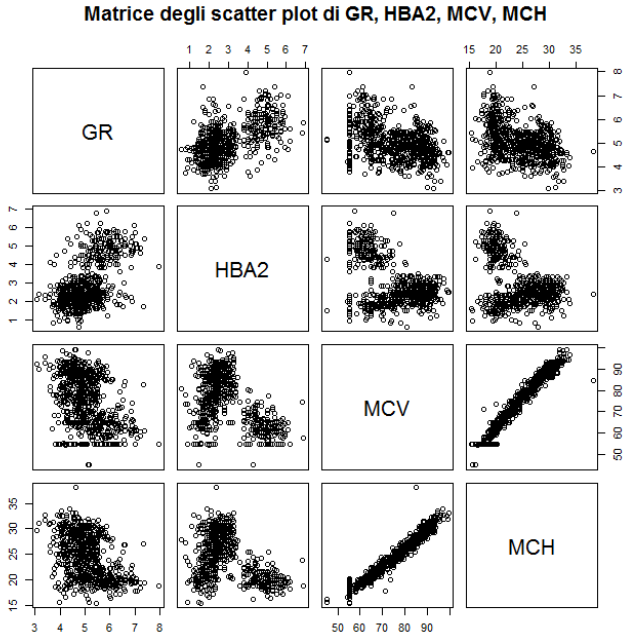


Figura 4.11 Matrice degli scatter plot di globuli rossi (GR), emoglobina A2 (HBA2), volume globulare medio (MCV) ed emoglobina corpuscolare media (MCH).

altra rappresentazione, notare la nuova libreria car e il parametro diagonal = "none"
library(car)
windows() # apre una nuova finestra
scatterplotMatrix(~GR+HBA2+MCV+MCH, reg.line=lm, smooth=TRUE, span=0.5, diagonal = "none", data=mydata, main="Matrice degli scatter plot con tendenze")
#
In questa rappresentazione ottenuta con l'impiego della libreria **car** sono riportate le curve che esprimono le tendenze medie dei dati a variare congiuntamente (**Figura 4.12**):

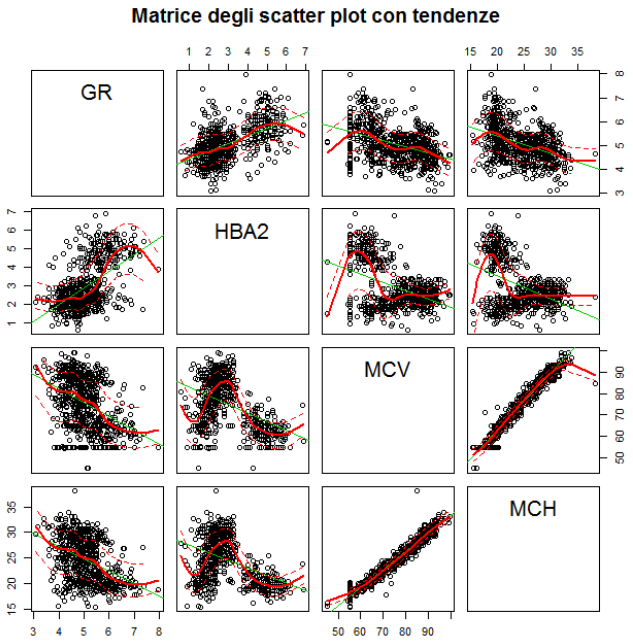


Figura 4.12 Matrice degli scatter plot di globuli rossi (GR), emoglobina A2 (HBA2), volume globulare medio (MCV) ed emoglobina corpuscolare media (MCH) con evidenziate le tendenze medie delle variabili a variare congiuntamente. La relazione tra MCV e MCH è chiaramente lineare.

come al punto precedente, notare il parametro diagonal = "histogram"
library(car)
windows() # apre una nuova finestra
scatterplotMatrix(~GR+HBA2+MCV+MCH, reg.line=lm, smooth=TRUE, span=0.5, diagonal = "histogram", data=mydata, main="Matrice degli scatter plot con istogrammi")
#
Nella diagonale sono ora rappresentati gli istogrammi delle distribuzioni (**Figura 4.13**):

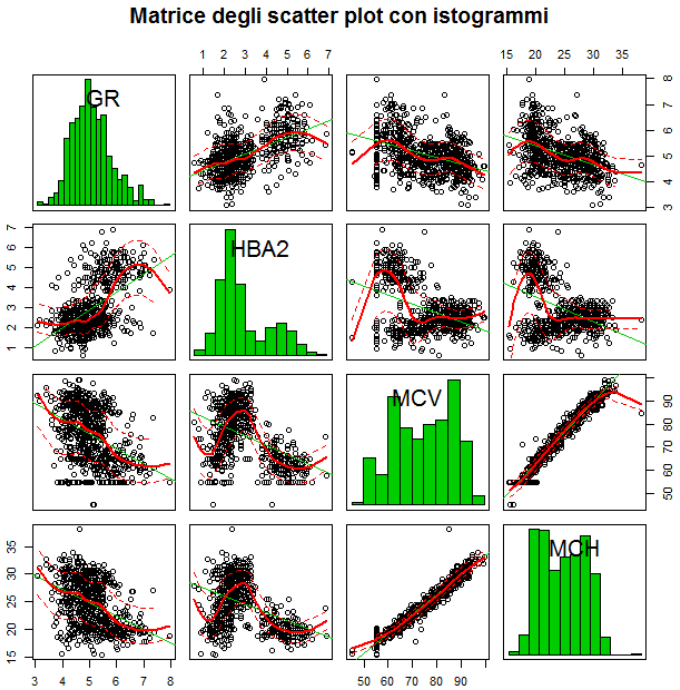


Figura 4.13 Matrice degli scatter plot di globuli rossi (GR), emoglobina A2 (HBA2), volume globulare medio (MCV) ed emoglobina corpuscolare media (MCH) con evidenziate le tendenze medie delle variabili a variare congiuntamente. La relazione tra MCV e MCH è chiaramente lineare. Nella diagonale sono stati riportati gli istogrammi.

Nel codice riportato sopra compare **diagonal = "histogram"**. Il parametro **diagonal** ammette, oltre al valore **"none"** (utilizzato per la rappresentazione della figura 4.12) e al valore **"histogram"** utilizzato nella figura precedente, anche i seguenti valori: **"boxplot"**, **"density"**,**"oned"**, **"qqplot"**. Provate ad utilizzarli modificando opportunamente il codice **R** riportato sopra per vedere cosa accade in questi casi.

Una nuova opportunità nella rappresentazione degli scatter plot sotto forma di matrici è offerta dalla libreria gclus con il codice che segue:

questo scatterplot necessita la libreria gclus
library(gclus)
windows() # apre una nuova finestra
dta <- mydata[c(1,2,3,4,5,6,7,8,9,10)] # recupera i dati dalle colonne
dta.r <- abs(cor(dta)) # calcola la correlazione
dta.col <- dmat.color(dta.r) # applica i colori
riordina le variabili in modo che quelle meglio correlate siano vicine alla diagonale
dta.o <- order.single(dta.r)
cpairs(dta, dta.o, panel.colors=dta.col, gap=.5, main="Variabili ordinate in base alla correlazione")
#
Le variabili sono colorate e ordinate in base alla maggiore o minore correlazione esistente tra di loro, quelle meglio correlate sono collocate accanto alla diagonale, le altre sono collocate andando dalla diagonale verso la periferia via via che la correlazione diminuisce (**Figura 4.14**):

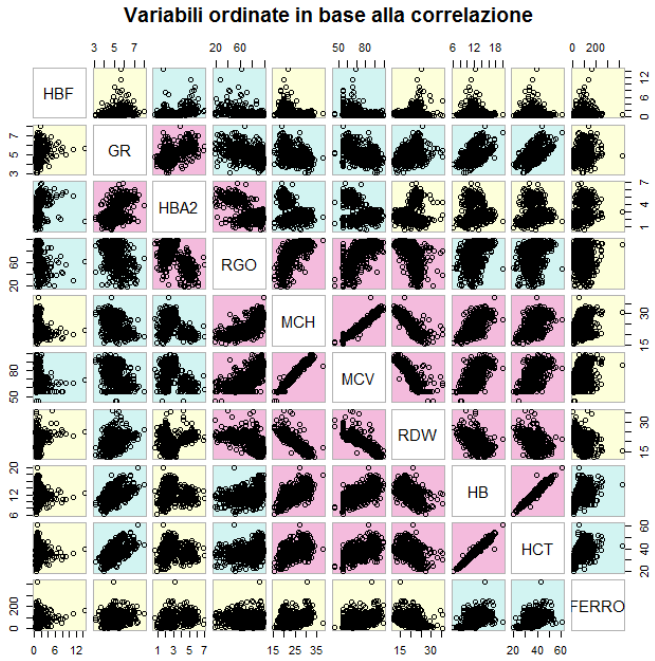


Figura 4.14 Matrice degli scatter plot di tutte le variabili contenute nel file di dati Scatterplot.csv che include controlli sani, soggetti portatori di beta-talassemia, portatori di alfa-talassemia, e soggetti con anemia sideropenica. Gli scatter plot sono ordinati in base alla correlazione tra le variabili, dalla diagonale (quelle meglio correlate e in rosa) verso l'esterno (quelle peggio correlate in giallo chiaro).

A questo punto, dato che con le molteplici istruzioni **windows()** avete aperto via via nuove finestre che si sono andate sovrapponendo, ciascuna con il proprio grafico, spostate o iconizzate la finestra dell'ultimo grafico per vedere la finestra con il grafico precedente, e così via.

4.5. Scatter plot 3D

Caricate e salvate nella cartella C:\R\ il file Scatterplot.csv. Si tratta degli stessi dati utilizzati nella precedente parte dedicata agli scatter plot (bidimensionali), nella quale trovate i dettagli sui dati contenuti nel file. Da notare che per gli scatte plot tridimensionali (3D) sono utilizzate la libreria **scatterplot3d**, la libreria **rgl**, e la libreria **Rcmdr** che, se non lo avete ancora fatto, dovete scaricare dal CRAN prima di eseguire l'esempio (in caso contrario si verificherà un errore nell'esecuzione del codice laddove è previsto l'utilizzo delle librerie).

```
# innanzitutto importiamo i dati
mydata <- read.table("c:/R/Scatterplot.csv", header=TRUE, sep=";")
# visualizziamo i dati
mydata
# scatter plot tridimensionale (3D) semplice, necessita libreria apposita
library(scatterplot3d)
attach(mydata)
scatterplot3d(HBA2,GR,MCV, main="Scatter plot 3d semplice")
#
```

Lo scatter plot 3D è qui rappresentato in modo molto semplice (**Figura 4.15 a sinistra**). Si rimanda alla documentazione della libreria **scatterplot3d** le informazioni sui molti argomenti con cui questa rappresentazione può essere migliorata e personalizzata[12]. Ad esempio con il codice che segue:

```
# scatter plot 3D a colori con linee verticali, necessita libreria apposita
windows() # apre una nuova finestra
```

[12] La documentazione delle librerie o package di R che risiedono sul CRAN è in genere molto ben indicizzata dai motori di ricerca. Ad esempio in questo caso se nella finestra di ricerca di Google digitate "package scatterplot3d pdf" (senza le virgolette) trovate immediatamente il file con la documentazione di questo pacchetto.

```
library(scatterplot3d)
attach(mydata)
scatterplot3d(HBA2,GR,MCV, pch=16, highlight.3d=TRUE, type="h", main="Scatter plot 3d con linee delle coordinate")
#
```

potete tracciare la proiezione dei punti sulle coordinate orizzontali per meglio identificare la loro posizione (**Figura 4.15 a destra**).

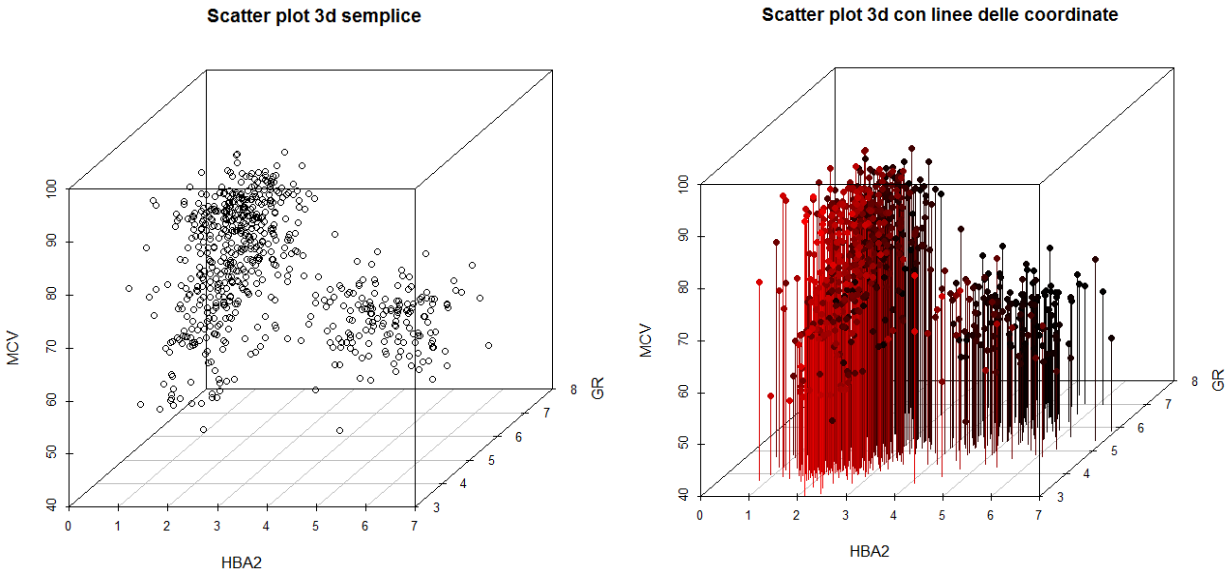


Figura 4.15 Scatter plot 3D (x,y,z) della concentrazione dell'emoglobina A2 negli eritrociti (HBA2 in %, asse x), dei globuli rossi (GR in 10^{12}/litro, asse y) e volume globulare medio (MCV in fL, asse z), con i soli punti (a sinistra) e con la proiezioni dei punti sul piano delle coordinate orizzontali x,y (a destra).

Con la libreria **rgl** è possibile realizzare un grafico 3D che può essere ruotato al fine di orientare i dati secondo la prospettiva che li coglie al meglio:

```
# spinning 3D scatter plot, necessita libreria apposita
mydata <- read.table("c:/R/Scatterplot.csv", header=TRUE, sep=";")
library(rgl)
attach(mydata)
axes3d()
bg3d("white")
plot3d(HBA2,GR,MCV, type="p", col="red", size=3)
#
```

Se "afferrate" il grafico 3D facendo click con il tasto sinistro del mouse e lo tenete premuto senza rilasciarlo, potete ruotarlo a vostro piacimento (**Figura 4.16**).

A questo punto, dato che con le due istruzioni **windows()** avete aperto due nuove finestre, avrete un totale di tre finestre, con altrettanti grafici, sovrapposte. Spostate o iconizzate la finestra dell'ultimo grafico per vedere la finestra con il grafico precedente, e così via.

Per eseguire lo script che segue dovete chiudere completamente **R** e riaprirlo per inizializzarlo. Lo script prevede l'utilizzo della libreria **Rcmdr** che consente anche in questo caso di realizzare in grafico 3D che può essere ruotato al fine di orientare i dati secondo la prospettiva che li coglie al meglio:

```
# spinning 3D scatter plot, necessita libreria apposita
# innanzitutto importiamo i dati
```

```
mydata <- read.table("c:/R/Scatterplot.csv", header=TRUE, sep=";")
library(Rcmdr)
attach(mydata)
scatter3d(HBA2,GR,MCV)
#
```

Se “afferrate” il grafico 3D facendo click con il tasto sinistro del mouse e lo tenetelo premuto senza rilasciarlo, potete ruotarlo a vostro piacimento (**Figura 4.17**).

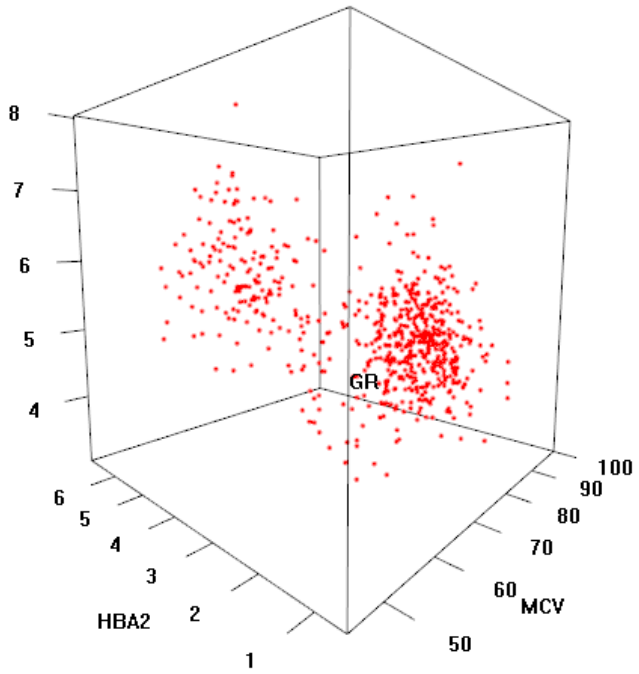


Figura 4.16 Scatter plot 3D realizzato con la libreria **rgl**. Se nella finestra grafica di **R** “afferrate” il grafico 3D facendo click con il tasto sinistro (tenete premuto il tasto senza rilasciarlo), potete ruotare il grafico a vostro piacimento

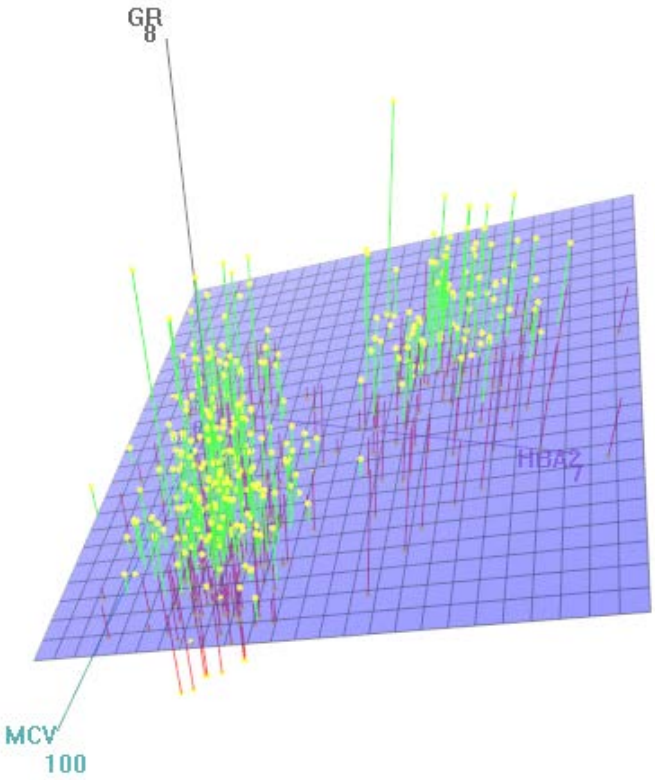


Figura 4.17 Scatter plot 3D realizzato con la libreria **Rcmdr**. Se nella finestra grafica di **R** “afferrate” il grafico 3D facendo click con il tasto sinistro (tenete premuto il tasto senza rilasciarlo), potete ruotare il grafico a vostro piacimento

4.6. Correlogrammi

Caricate e salvate nella cartella C:\R\ il file Scatterplot.csv. Si tratta degli stessi dati utilizzati nella precedente parte dedicata agli scatter plot (bidimensionali), nella quale trovate i dettagli sui dati contenuti nel file. Da notare che per realizzare i correlogrammi viene utilizzata la libreria **corrgram** che, se non lo avete ancora fatto, dovete scaricare dal CRAN prima di eseguire l'esempio (in caso contrario si verificherà un errore nell'esecuzione del codice).

Copiate e incollate nella `Console di R` ed eseguite questo codice:

```
# sono importati i dati
mydata <- read.table("c:/R/Statcorr.csv", header=TRUE, sep=";")
# correlogramma semplice
library(corrgram)
corrgram(mydata, order=TRUE, lower.panel=panel.shade, upper.panel=panel.pie, text.panel=panel.txt,
main="Correlogramma semplice")
#
```

Ecco come appare un correlogramma (**Figura 4.18**). L'ampiezza della colorazione della torta misura il coefficiente di correlazione (torta completamente bianca r = 0, torta completamente colorata r = 1), i valori dei coefficienti di correlazione vanno decrescendo dalla diagonale centrale verso la periferia, in blu sono riportati i valori positivi di r (le due grandezze aumentano e diminuiscono congiuntamente), in rosso i valori negativi di r (all'aumentare di una delle due grandezze l'altra diminuisce e viceversa).

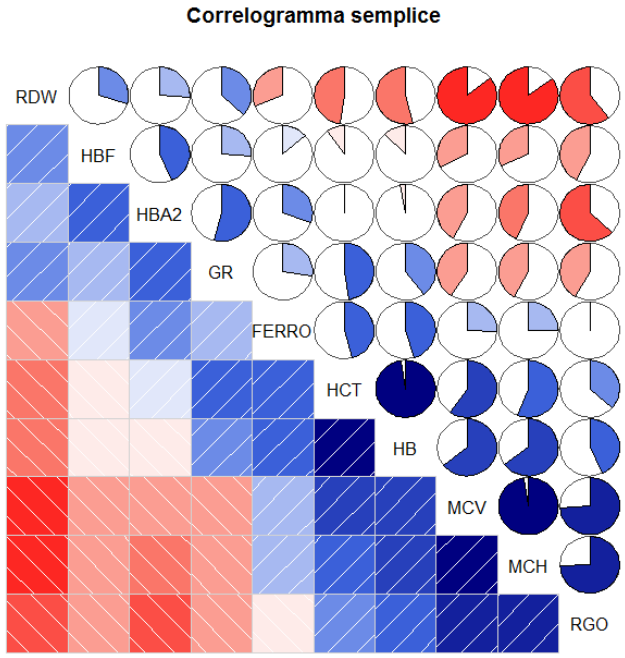


Figura 4.18 Correlogramma con intensità della colorazione e superfice colorata proporzionali al valore del coefficiente di correlazione, in blu i valori positivi e in rosso i valori negativi del coefficiente di correlazione.

```
# correlogramma con tendenze evidenziate
windows() # apre una nuova finestra
corrgram(mydata, order=TRUE, lower.panel=panel.ellipse, upper.panel=panel.pts, text.panel=panel.txt,
diag.panel=panel.minmax, main="Correlogramma con tendenze evidenziate")
#
```

In questo caso (**Figura 4.19**) nel quadrante superiore sono riportati i diagrammi di dispersione (scatter plot) e nel quadrante inferiore sono riportate le rette o le curve che esprimono le tendenze medie dei dati a variare congiuntamente.

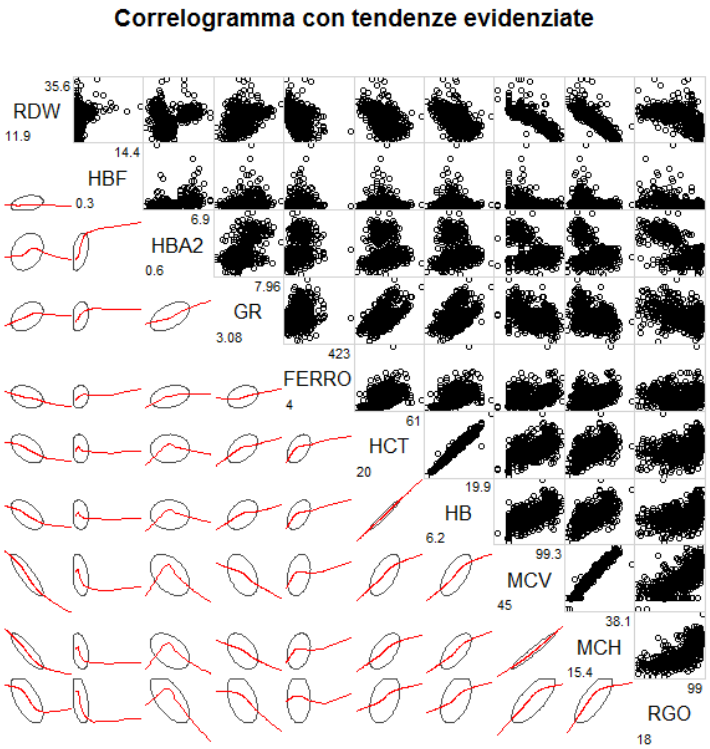


Figura 4.19 Correlogramma con evidenziate le tendenze medie delle variabili a variare congiuntamente.

```
# correlogramma con i coefficienti di correlazione e i loro limiti di confidenza
windows() # apre una nuova finestra
corrgram(mydata, lower.panel=panel.pts, upper.panel=panel.conf, diag.panel=panel.density,
main="Correlogramma con i coefficienti di correlazione r")
#
```

In questa forma di correlogramma (**Figura 4.20**) nella diagonale sono riportate le distribuzioni delle variabili sotto forma di kernel density plot, nel quadrante inferiore i diagrammi di distribuzione (scatter plot) e nel quadrante superiore il valore del coefficiente di correlazione r con i limiti di confidenza al 95%.

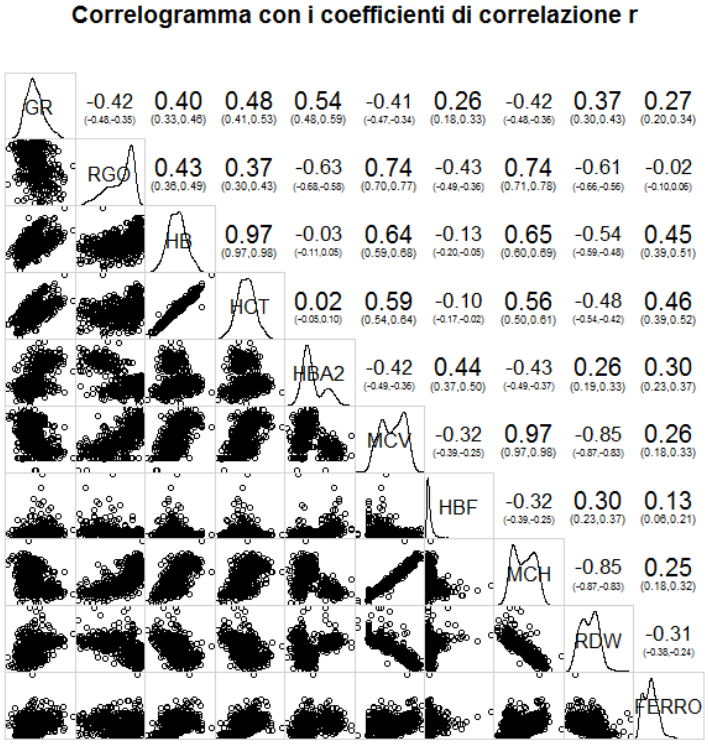


Figura 4.20 Correlogramma con il valore del coefficiente di correlazione e i suoi limiti di confidenza al 95%.

In realtà anche un più tradizionale scatter plot (**Figura 4.21**) aiuta a cogliere le forti correlazioni che intercorrono tra emoglobina (HB) ed ematocrito (HCT) e tra emoglobina corpuscolare media (MCH) e volume globulare medio (MCV):

```
# potete confermare le forti correlazioni tra HB/HCT e tra MCH/MCV anche con uno scatter plot
library(car)
windows() # apre una nuova finestra
scatterplotMatrix(~GR+RGO+HB+HCT+HBA2+MCV+HBF+MCH+RDW+FERRO, reg.line=lm, smooth=TRUE,
span=0.5, diagonal = "density", main="Matrice degli scatter plot", data=mydata)
#
```

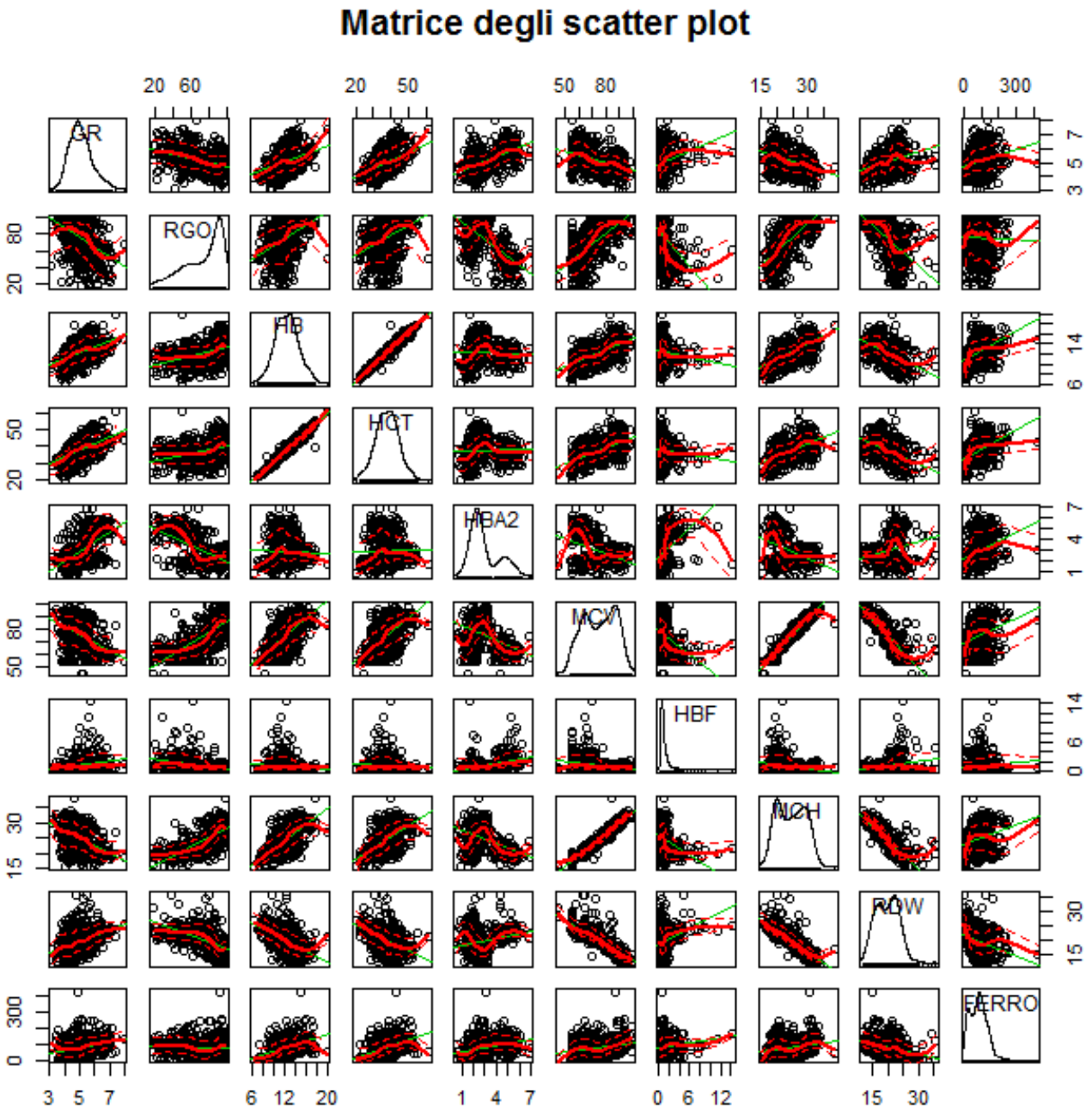


Figura 4.21 Una matrice degli scatter plot aiuta a valutare le possibili relazioni tra le variabili.

A questo punto, dato che con le tre istruzioni **windows()** avete aperto tre nuove finestre, avrete un totale di quattro finestre, con altrettanti grafici, sovrapposte. Spostate o iconizzate la finestra dell'ultimo grafico

per vedere la finestra con il grafico precedente, e così via.

Come ho evidenziato anche sul mio sito nella pagina su Teorema di Bayes e decisioni mediche[13] il fatto che due variabili siano correlate non ci dice nulla sui possibili rapporti causa-effetto. Anzi, è possibile che siano "evidentemente" correlati dal punto di vista statistico fatti che in realtà sono completamente slegati tra di loro. Nonostante ciò quando utilizzata in modo appropriato la correlazione può essere utile. Ed è quello che accade quando, come nel casi dei correlogrammi, il coefficiente di correlazione viene integrato con una rappresentazione grafica dei dati che aiuta a fare emergere i legami fra le variabili in esame.

4.7. Curve ROC

Scaricate e salvate nella cartella C:\R\ i file CurveROC.csv e CurveROCbis.csv. Il contenuto di entrambi i file aperto con un editor di testo come il Blocco note di Windows vi apparirà così (cambiano solamente i valori), con i nomi delle variabili nella prima riga e i dati dei singoli casi nelle righe successive:

```
predictions;labels
19;0
22;0
22;1
24;1
24;1
26;0
```

La variabile "`predictions`" contiene i valori misurati (in questo caso il risultato numerico di una analisi di laboratorio) mentre la variabile "`labels`" contiene la classificazione dei casi, e riporta 0 per i controlli (soggetti sani) e 1 per i soggetti malati. Come separatore di campo viene utilizzato il punto e virgola (`;`).

Da notare che sono utilizzate la libreria **pROC** e la libreria **sm** che, se non lo avete ancora fatto, dovete scaricare dal CRAN prima di eseguire l'esempio (in caso contrario si verificherà un errore nell'esecuzione del codice laddove è previsto l'utilizzo delle librerie). Copiate e incollate nella `Console di R` questo codice ed eseguitelo soffermandovi sui singoli passaggi:

```
# sono importati i dati
mydata <- read.table("c:/R/CurveROC.csv", header=TRUE, sep=";")
# nomi delle variabili in mydata
names(mydata)
# lista dei primi 10 casi di mydata
head(mydata, n=10)
# lista degli ultimi 5 casi di mydata
tail(mydata, n=5)
# utilizza la libreria pROC
library(pROC)
attach(mydata)
# traccia la curva ROC e calcola l'area sotto la curva (auc)
roc(mydata$labels, mydata$predictions, smooth = FALSE, auc = TRUE, ci = FALSE, plot = TRUE, identity = TRUE, main = "Curva ROC", xlab="1-specificità", ylab = "Sensibilità")
#
```

Dopo avere con la prima riga di codice importato i dati sono mostrati (**names(mydata)**) i nomi delle variabili:

```
[1] "predictions" "labels"
```

[13] http://www.bayes.it/html/decisioni_mediche.html

Quindi sono mostrati (**head(mydata, n=10)**) i primi 10 dati importati:

```
   predictions labels
1           19      0
2           22      0
3           22      1
4           24      1
5           24      1
6           26      0
7           27      1
8           28      0
9           29      0
10          29      0
```

E infine sono mostrati (**tail(mydata, n=5)**) gli ultimi cinque dati importati:

```
     predictions labels
1691         235      1
1692         237      1
1693         237      1
1694         242      1
1695         242      1
```

Infine viene tracciata la curva ROC e viene calcolata l'area sotto la curva (area under the curve ovvero a.u.c.)

```
Data: mydata$predictions in 853 controls (mydata$labels 0) < 842 cases
(mydata$labels 1).
Area under the curve: 0.9633
```

Ecco il grafico della curva ROC che viene prodotto (**Figura 4.22**):

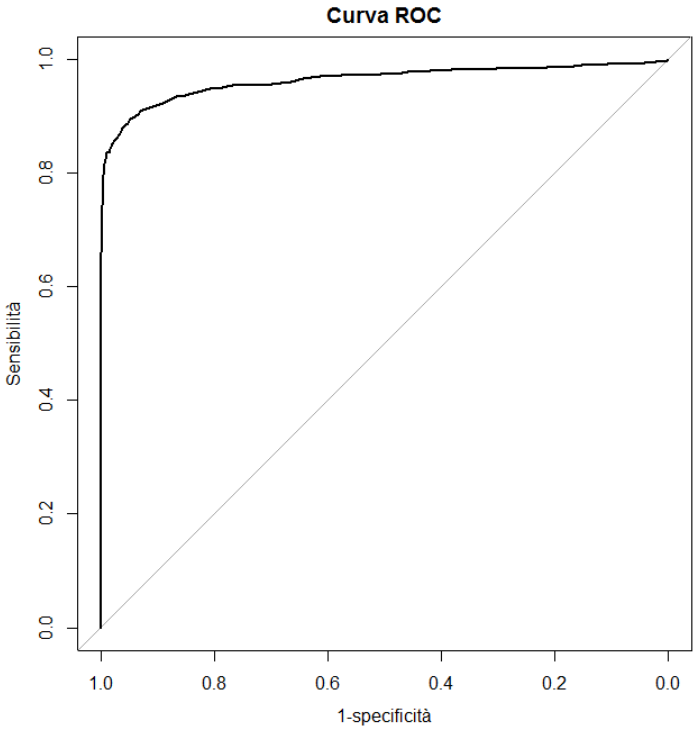


Figura 4.22 Curva ROC ricavata dalle distribuzioni dei risultati di un test di laboratorio in soggetti sani e malati. Il grafico è realizzato mediante la libreria **pROC**.

Con il blocco di codice che segue sono infine calcolate le statistiche della curva ROC:

```
# intervallo di confidenza al 95% dell'area sotto la curva, metodo di DeLong
ci.auc(mydata$labels, mydata$predictions, conf.level = 0.95)
```

intervallo di confidenza al 95% della sensibilità per valori di specificità da 0 a 1 con passo 0.1
ci.se(mydata$labels, mydata$predictions, specificities=seq(0,1,.1), conf.level = 0.95, boot.n = 100)
intervallo di confidenza al 95% della specificità per valori di sensibilità da 0 a 1 con passo .1
ci.sp(mydata$labels, mydata$predictions, sensitivities=seq(0,1,.1), conf.level = 0.95, boot.n = 100)
calcola il miglior valore soglia tra sani e malati e l'intervallo di confidenza al 95% della sensibilità e della specificità corrispondenti
ci.thresholds(mydata$labels, mydata$predictions, thresholds="best", conf.level = 0.95, boot.n = 100)
calcola per le principali grandezze i valori corrispondenti al valore soglia tra sani e malati
myroc <-roc(mydata$labels, mydata$predictions, plot = FALSE)
coords(myroc, "best", best.method = "youden", ret=c("threshold", "specificity", "sensitivity", "accuracy", "tn", "tp", "fn", "fp", "npv", "ppv"))
#
Innanzitutto viene calcolato l'intervallo di confidenza al 95% dell'area sotto la curva impiegando il metodo di DeLong:

```
95% CI: 0.9537-0.9729 (DeLong)
```

Quindi viene riportata una tabella con la mediana e gli intervalli di confidenza al 95% della sensibilità per valori di specificità che vanno da 0 a 1 con passo 0.1 (ovviamente è facile ripetere i calcoli cambiando passo a piacimento):

```
95% CI (100 stratified bootstrap replicates):
  sp se.low se.median se.high
 0.0 1.0000    1.0000  1.0000
 0.1 0.9844    0.9916  0.9964
 0.2 0.9780    0.9869  0.9923
 0.3 0.9734    0.9831  0.9912
 0.4 0.9692    0.9807  0.9882
 0.5 0.9628    0.9739  0.9831
 0.6 0.9596    0.9708  0.9810
 0.7 0.9437    0.9565  0.9695
 0.8 0.9350    0.9502  0.9648
 0.9 0.9035    0.9206  0.9370
 1.0 0.6288    0.6758  0.7732
```

Quindi viene riportata una tabella con la mediana e gli intervalli di confidenza al 95% della speciificità per valori di sensibilità che vanno da 0 a 1 con passo 0.1 (anche in questo caso è facile ripetere i calcoli cambiando passo a piacimento):

```
95% CI (100 stratified bootstrap replicates):
  se sp.low sp.median  sp.high
 0.0 1.0000  1.000000 1.000000
 0.1 1.0000  1.000000 1.000000
 0.2 1.0000  1.000000 1.000000
 0.3 1.0000  1.000000 1.000000
 0.4 1.0000  1.000000 1.000000
 0.5 1.0000  1.000000 1.000000
 0.6 1.0000  1.000000 1.000000
 0.7 0.9965  0.998800 1.000000
 0.8 0.9894  0.994100 0.999400
 0.9 0.9062  0.942500 0.964600
 1.0 0.0000  0.001172 0.005305
```

Successivamente sono calcolati la mediana e l'intervallo di confidenza al 95% della sensibilità e della specificità in corrispondenza del miglior valore soglia tra sani e malati:

```
95% CI (100 stratified bootstrap replicates):
 thresholds sp.low sp.median sp.high se.low se.median se.high
       74.5 0.9297    0.9484  0.9619  0.877    0.8955  0.9139
```

Infine calcola per le principali grandezze i valori corrispondenti al valore soglia tra sani e malati:

```
  threshold specificity sensitivity    accuracy          tn          tp
 74.5000000   0.9484174   0.8942993   0.9215339 809.0000000 753.0000000
fn          fp         npv         ppv
89.0000000  44.0000000   0.9008909   0.9447930
```

ove `tn` sono i veri negativi (true negative), `tp` sono i veri positivi (true positive), `fn` sono in falsi negativi (false negative), `fp` sono i falsi positivi (false positive), `npv` è il valore predittivo del test negativo (negative predictive value) e `ppv` è il valore predittivo del test positivo (positive predictive value).

Per ulteriori approfondimenti si rimanda alla documentazione della libreria **pROC**.

Un grafico che mostra, sovrapposte, le distribuzioni dei valori nei sani e nei malati aiuta certamente nella lettura dei dati (**Figura 4.23**):

```
# traccia kernel density plot sovrapposti dei valori osservati per controlli sani (0) e malati (1)
library(sm)
attach(mydata)
# attenzione il primo "labels" è la variabile che contiene i valori osservati il secondo "labels" sono le etichette da applicare come legenda
myplot <- factor(labels, levels= c("0","1"), labels = c("Sani", "Malati"))
# traccia i due grafici sovrapposti
windows() # apre una nuova finestra
sm.density.compare(predictions, labels, xlab="Valori osservati", ylab="Densità")
title(main="Distribuzione dei valori nei due gruppi")
# aggiunge la legenda: posizionarsi dove la si desidera fare comparire e fare click con tasto sinistro del mouse
colfill<-c(2:(2+length(levels(myplot))))
legend(locator(1), levels(myplot), fill=colfill)
#
```

Il codice traccia due kernel density plot indipendenti e sovrapposti dei valori osservati nei controlli sani e nei malati e rimane in attesa. A questo punto posizionate il mouse dove volete che compaia la legenda, e fate click con il tasto sinistro per farla comparire.

A questo punto ricordate che con l'istruzione windows() avete aperto una nuova finestra, quindi avete un totale di due finestre, con altrettanti grafici, sovrapposte. Spostate o iconizzate la finestra dell'ultimo grafico per vedere la finestra con il grafico precedente.

Ora copiate e incollate nella Console di R ed eseguite questo codice, con il quale sono importate due serie di dati, le cui curve ROC sono poi sovrapposte sullo stesso sistema di assi cartesiani (**Figura 4.24**):

```
# importa i dati per le due curve ROC
mydata <- read.table("c:/R/CurveROC.csv", header=TRUE, sep=";")
mydatabis <- read.table("c:/R/CurveROCbis.csv", header=TRUE, sep=";")
library(pROC)
# traccia la prima curva ROC
roc(mydata$labels, mydata$predictions, smooth = FALSE, auc = TRUE, ci = FALSE, plot = TRUE, identity = FALSE, main = "Curve ROC sovrapposte", xlab="1-specificità", ylab = "Sensibilità")
# traccia la seconda curva ROC
roc(mydatabis$labels, mydatabis$predictions, smooth = FALSE, auc = TRUE, ci = FALSE, plot = TRUE, add = TRUE, col = "red", lty = 4)
#
```

L'argomento add = TRUE consente, quando viene tracciata la seconda curva ROC, di sovrapporla alla prima.

Inoltre specificando il colore col = "red" e la linea tratteggiata lty = 4 le due curve ROC possono essere meglio distinte.

Rimando chi fosse interessato ad approfondire questo tema al paragrafo del sito che ho preparato per illustrare basi storiche e significato delle curve ROC.

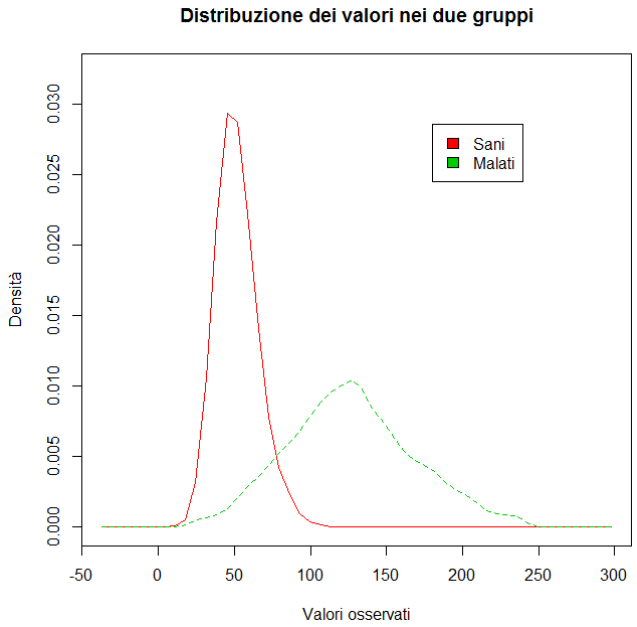


Figura 4.23 Kernel density plot sovrapposti delle distribuzioni dei risultati di un test di laboratorio in soggetti sani e malati. I dati sono quelli della curva ROC di figura **4.22**.

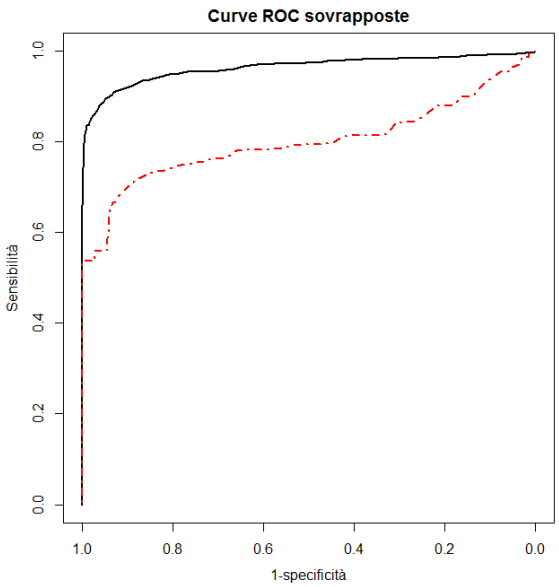


Figura 4.24 Due curve ROC sovrapposte consentono di evidenziare come il test con la curva in nero continuo fornisca una informazione maggiore di quello con la curva ROC in colore rosso trattegiato.

5. R problemi scelti

5.1. Confronto tra due metodi analitici

La regressione lineare standard non è adatta al confronto tra due variabili legate tra loro da una relazione lineare se anche la variabile indipendente è affetta da errore di misura. Ed è proprio quello che accade quando si confrontano tra di loro i risultati di due metodi analitici per la determinazione della stessa sostanza. In questo caso viene suggerito una approccio che prevede l'ispezione dei dati mediante il diagramma di Bland e Altman, e l'impiego della regressione lineare non parametrica di Passing e Bablok, che assume che entrambe le variabili siano affette da un errore di misura. L'approccio globale al confronto tra due metodi con **R** è stato sviluppato con la libreria **MethComp**. Se non la trovate sul CRAN, potete scaricarla dal sito di `R-Forge`[14] e installarla sul vostro PC copiando e incollando nella `Console di R` questa riga di comando:

```
install.packages("MethComp", repos="http://R-Forge.R-project.org")
```

Per utilizzare la libreria **MethComp** la struttura dei dati deve prevedere obbligatoriamente il campo **meth** (il metodo di analisi), il campo **item** (il numero progressivo del campione analizzato), il campo **repl** (il numero del replicato) e il campo **y** (il risultato numerico dell'analisi).

Create la cartella C:\R\ e salvate in questa cartella il file MethComp.csv. Come vedete contiene i dati relativi al confronto tra due metodi analitici organizzati esattamente come previsto dalla libreria, anche se in questo caso il numero del replicato è sempre uguale a 1 dato che non erano previsti analisi in replicato:

meth	item	repl	y
Metodo x	1	1	4
Metodo y	1	1	3
Metodo x	2	1	4
Metodo y	2	1	3.9
Metodo x	……	……	……
Metodo y	……	……	……
Metodo x	188	1	115.4
Metodo y	188	1	110.2
Metodo x	189	1	156
Metodo y	189	1	152

Copiate e incollate nella `Console di R` ed eseguite questo codice:

```
#
mydata <- read.table("c:/R/MethComp.csv", header=TRUE, sep=";")
library(MethComp)
newdata <- Meth(mydata) # crea un oggetto Meth per la libreria
plot.Meth(newdata)
#
```

Vedete la sintesi grafica dei dati del confronto tra metodi., con il diagramma di bland e Altman in alto a destra e la regressione lineare non parametrica di Passing e Bablok in basso a sinistra (**Figura 5.1**).

[14] https://r-forge.r-project.org/

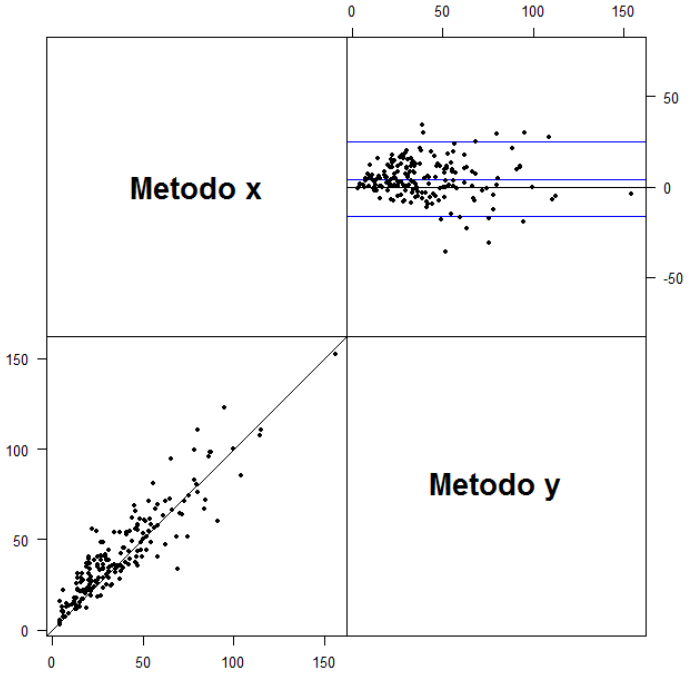


Figura 5.1 Grafico di Bland e Altman (in alto a destra) e regressione lineare non parametrica di Passing a Bablok (in basso a sinistra) in un confronto tra due metodi analitici realizzato mediante la libreria **MethComp**.

Ora copiate e incollate nella `Console di R` questo codice

```
#
predef <- par()$mar # salva i valori predefiniti dei margini
par(mar = c(5,5,5,4)) # imposta margini più ampi
BA.plot(newdata, main = "Grafico di Bland e Altman")
#
```

Per questo grafico è necessaria l'impostazione dei margini, il relativo problema viene illustrato al successivo paragrafo 5.4.

Al momento della creazione dell'oggetto `Meth` per la libreria viene fornita una breve sintesi dei dati:

```
> newdata <- Meth(mydata) # crea un oggetto Meth per la libreria

The following variables from the dataframe
"mydata" are used as the Meth variables:
meth: meth
item: item
repl: repl
   y: y
           #Replicates
Method              1 #Items #Obs: 378 Values:  min  med max
  Metodo x        189    189       189            4 31.0 156
  Metodo y        189    189       189            3 36.3 152
```

Cona la successiva e terza riga di codice viene tracciato il diagramma di Bland e Altman (**Figura 5.2**). Da notare un fatto che in genere viene tralasciato nella rappresentazione del diagramma, ma che invece è della massima importanza: i limiti di confidenza al 95% della media delle differenze, che invece sonoprevisti nella libreria MethComp.

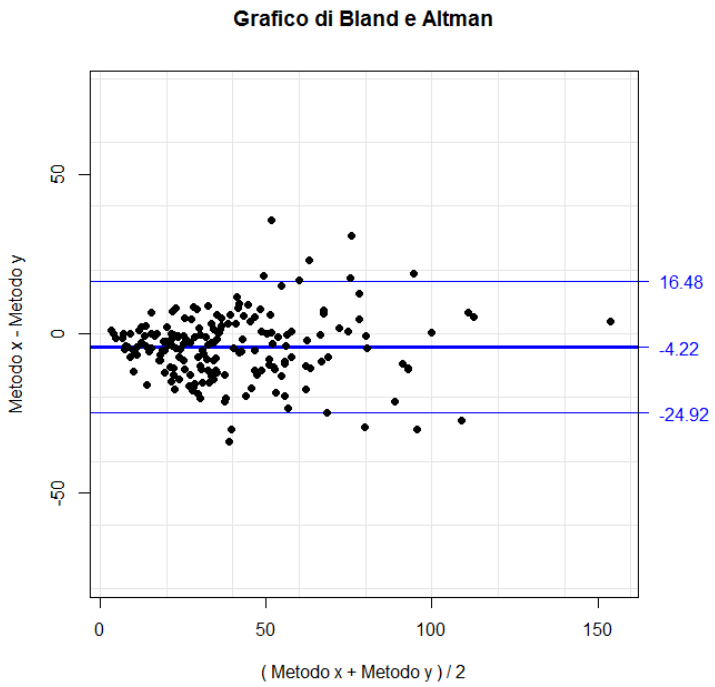


Figura 5.2 Diagramma di Bland e Altman con la media e i limiti di confidenza al 95% della media.

Infine copiate e incollate nella `Console di R` questo codice:

```
#
print(PBreg(newdata)) # statistiche della regressione di Passing e Bablok
par(mar = predef) # ripristina i valori predefiniti dei margini
plot(PBreg(newdata), main = "Regressione di Passing e Bablok") # traccia il grafico
#
```

Come prima cosa vedete le statistiche della regressione:

```
> print(PBreg(newdata)) # statistiche della regressione di Passing e
Bablok

Passing-Bablok linear regression of Metodo y on Metodo x

Observations read: 189, used: 189
Slopes calculated: 17766, offset: 1134

           Estimate    2.5%CI 97.5%CI
Intercept 3.9340857 2.1687117 5.37963
Slope     0.9888262 0.9296296 1.06135

Unadjusted summary of slopes:
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
   -335       0       1     Inf       1     Inf       1

Summary of residuals:
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-38.7600  -4.6960   0.0000   0.7072   7.4890  30.2100

Test for linearity: (passed)
```

Per l'intercetta e per il coefficiente angolare sono riportati i limiti di confidenza al 95%, che consentono di effettuare immediatamente la valutazione della significatività della differenza dell'intercetta da 0 (zero è il valore atteso dell'intercetta se tra i due metodi non vi è errore sistematico di tipo costante) e del coefficiente angolare da 1 (uno è il valore atteso del coefficiente angolare se tra i due metodi non vi è errore sistematico di tipo proporzionale).

Oltre alle statistiche vedete il grafico con la retta teorica di equivalenza metodo x = metodo y tratteggiata, e con la retta trovata che conferma graficamente l'esistenza tra i due metodi di una differenza sistematica di tipo costante (**Figura 5.3**), che ovviamene necessita di interpretazione dal punto di vista analitico.

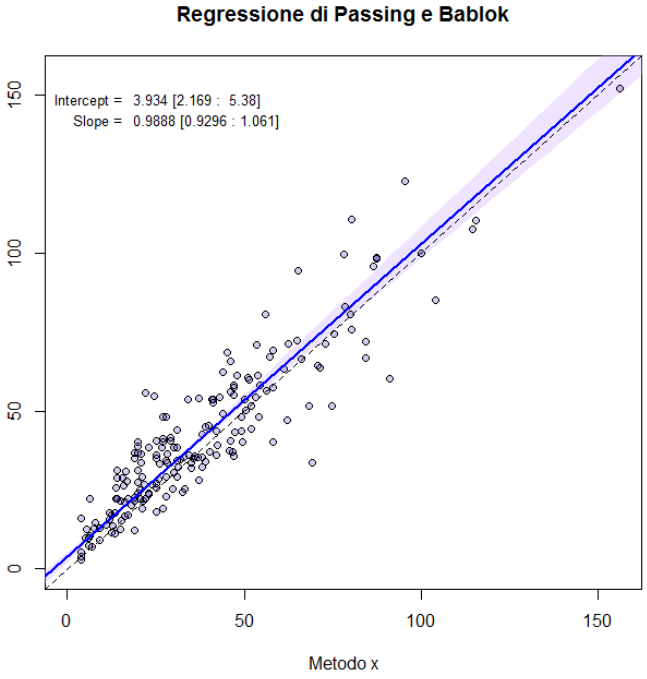


Figura 5.3 Regressione lineare non parametrica di Passing e Bablok con la retta di regressione e i limiti di confidenza al 95% della regressione lineare.

5.2. Kernel density plot sovrapposti

Abbiamo già visto (al paragrafo 4.2.) il codice **R** per tracciare due kernel density plot sovrapposti. Qui ne sovrapporremo cinque.

Utilizziamo gli stessi dati forniti per tracciare i box & whiskers plot (paragrafo 4.3). Se non l'avete già fatto, scaricate dal CRAN la libreria **sm**. Quindi create la cartella C:\R\ e salvate in questa cartella il file Boxplot.csv. Copiate e incollate nella `Console di` R ed eseguite questo codice:

```
#
mydata <- read.table("c:/R/Boxplot.csv", header=TRUE, sep=";")
library(sm)
attach(mydata)
myplot <- factor(Diagnosi, levels = c("AC","CAH","Controlli","CPH","NCAH"), labels = c("AC","CAH","Controlli","CPH","NCAH"))
sm.density.compare(IgA, Diagnosi, xlab="IgA in g/L", ylab="Frequenza (kernel density)")
title(main="IgA nelle malattie croniche del fegato")
# aggiunge la legenda: posizionarsi dove la si desidera fare comparire e fare click con tasto sinistro del mouse
colfill<-c(2:(2+length(levels(myplot))))
```

legend(locator(1), levels(myplot), fill=colfill)
#
Ecco i cinque kernel density plot sovrapposti che vengono tracciati (**Figura 5.4**). Anche questo codice come quelli che seguono è da tesaurizzare, per riadattarlo ad eventuali future esigenze.

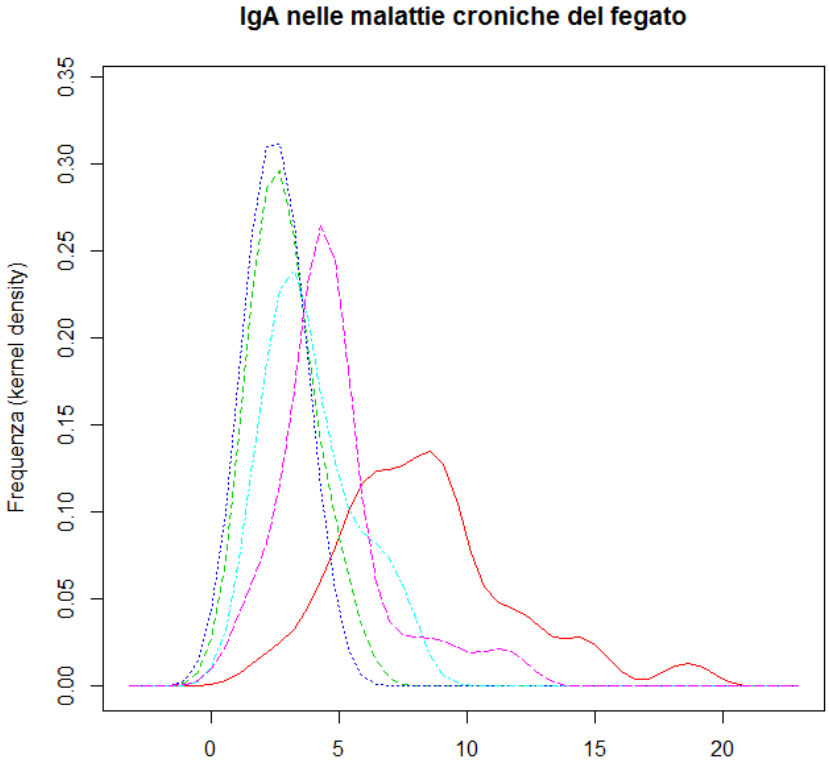


Figura 5.4 Kernel density plot sovrapposti possono aiutare nello studio comparativo di più distribuzioni.

5.3. Identificare i punti in uno scatter plot

Questo codice risponde a un problema banale, ma che si pone sovente: questo punto che si discosta così tanto dagli altri a quale dato corrisponde?

Create la cartella C:\R\ e salvate in questa cartella il file Scatterplot.csv. Copiate e incollate nella `Console di R` ed eseguite questo codice:
#
mydata <- read.table("c:/R/Scatterplot.csv", header=TRUE, sep=";")
attach(mydata)
plot(HB, HCT, main="Identifica punti in uno scatterplot", xlab="Emoglobina, mg/dL ", ylab="Ematocrito, %", pch=1)
identify(HB, HCT, plot = TRUE, atpen = FALSE, offset = 0.5, tolerance = 0.25, locatorBell = TRUE)
#
Posizionatevi nelle vicinanze del punto cui siete interessati e che volete identificare: poco sopra, appena sotto, un poco a sinistra o a destra, e fate click con il tasto sinistro del mouse: nella posizione prescelta comparirà il numero del dato. Per terminare selezionate `Stop` con il tasto destro del mouse. Il risultato è riportato nella **Figura 5.5**.

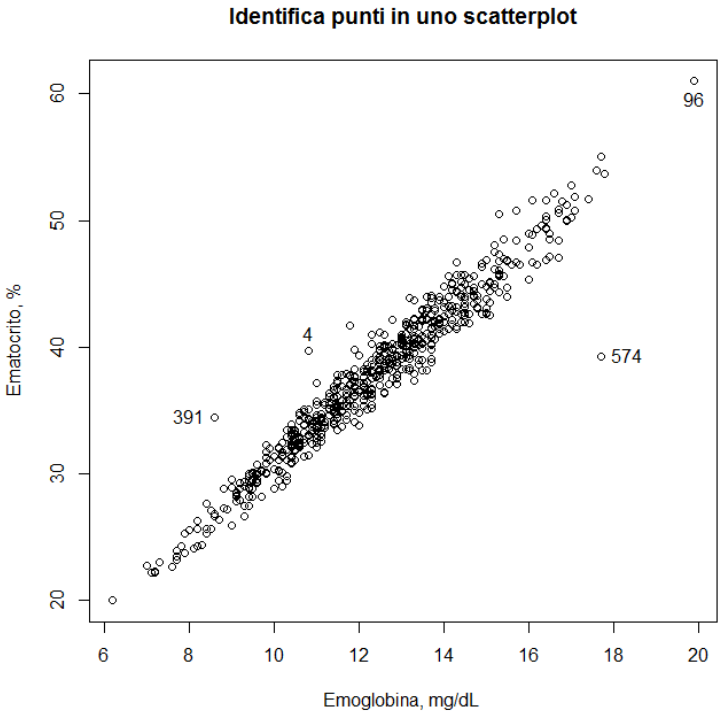


Figura 5.5 Un semplice script consente, dopo avere tracciato uno scatter plot, di identificare il numero del dato che corrisponde ad uno specifico punto con un click del mouse.

5.4. Adattare i margini a una immagine

Chi si è interessato al problema del confronto tra metodi (paragrafo 5.1.) avrà certamente notato questa strana riga di codice con il relativo commento:

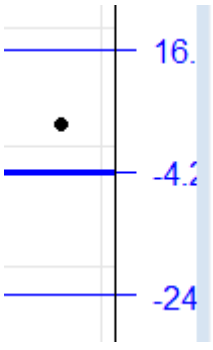
par(mar = c(5,5,5,4)) # imposta margini più ampi

Per avere la spiegazione dovete scaricate la libreria **MethComp**. Se non la trovate sul CRAN, potete scaricarla dal sito di `R-Forge`[15] e installarla sul vostro PC copiando e incollando nella `Console di R` questa riga di comando:

```
install.packages("MethComp", repos="http://R-Forge.R-project.org")
```

Create la cartella C:\R\ e salvate in questa cartella il file MethComp.csv. Copiate e incollate nella `Console di R` questo codice ed eseguitelo:

```
#
mydata <- read.table("c:/R/MethComp.csv", header=TRUE, sep=";")
library(MethComp)
newdata <- Meth(mydata) # crea un oggetto Meth per la libreria
BA.plot(newdata, main = "Grafico di Bland e Altman")
#
```



Come vedete i margini sono insufficienti e i valori sulla scala di destra risultano troncati.

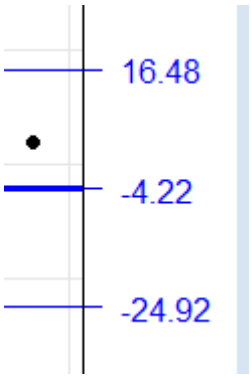
Ora copiate e incollate nella Console di R ed eseguite questo codice:

```
#
mydata <- read.table("c:/R/MethComp.csv", header=TRUE, sep=";")
```

[15] https://r-forge.r-project.org/

```
library(MethComp)
newdata <- Meth(mydata) # crea un oggetto Meth per la libreria
predef <- par()$mar # salva i valori predefiniti dei margini
par(mar = c(5,5,5,4)) #imposta i nuovi margini
BA.plot(newdata, main = "Grafico di Bland e Altman")
par(mar = predef) # ripristina i valori predefiniti dei margini
#
```

Come vedete i margini sono ora sufficienti a contenere i valori sulla scala di destra.



5.5. Inserire più grafici in una immagine

Anche questo è un problema banale ma che può portare a perdersi tra migliaia di librerie e i milioni di righe di codice di **R** senza trovare la soluzione.

Create la cartella C:\R\ e salvate in questa cartella il file Verigauss.csv. Contiene i dati di sesso, età e concentrazione di colesterolo totale, colesterolo HDL, colesterolo LDL e trigliceridi che abbiamo già incontrato:

Sesso	Eta	Colesterolo	HDL	LDL	Trigliceridi
M	33	56	44	9	19
M	62	60	5		
F	90	70	30		99
M	75	80	53		
F	32	82	51		23
M	71	84	25		
F	……	……	……	……	……
F	……	……	……	……	……

Copiate e incollate nella `Console diR` ed eseguite questo codice:

```
#
mydata <- read.table("c:/R/Verigauss.csv", header=TRUE, sep=";")
newdata <- na.omit(mydata) #esclude i casi con dati mancanti
tri <- newdata$Trigliceridi
par(mfrow=c(2,2))
hist(tri, main="Istogramma dei dati", xlab="Trigliceridi in mg/dL", ylab = "Frequenza")
plot(density(tri), main="Distribuzione di densità dei dati", xlab="Trigliceridi in mg/dL", ylab = "Frequenza")
plot(ecdf(tri), main="Distribuzione cumulativa empirica", xlab="Trigliceridi in mg/dL", ylab = "Frequenza cumulativa")
qqnorm((tri-mean(tri))/sd(tri), main="Quantili campionari vs. teorici", xlab="Quantili teorici", ylab = "Quantili campionari")
abline (0,1) # linea di allineamento teorico di dati gaussiani
#
```

Il comando chiave è "**par(mfrow=c(2,2))**" che predispone la matrice 2 righe x 2 colonne da riempire con i quattro grafici (**hist**, **plot(density**, **plot(ecdf** e **qqnorm**) per riga, ovvero da sinistra in alto a destra in basso (**Figura 5.6**).

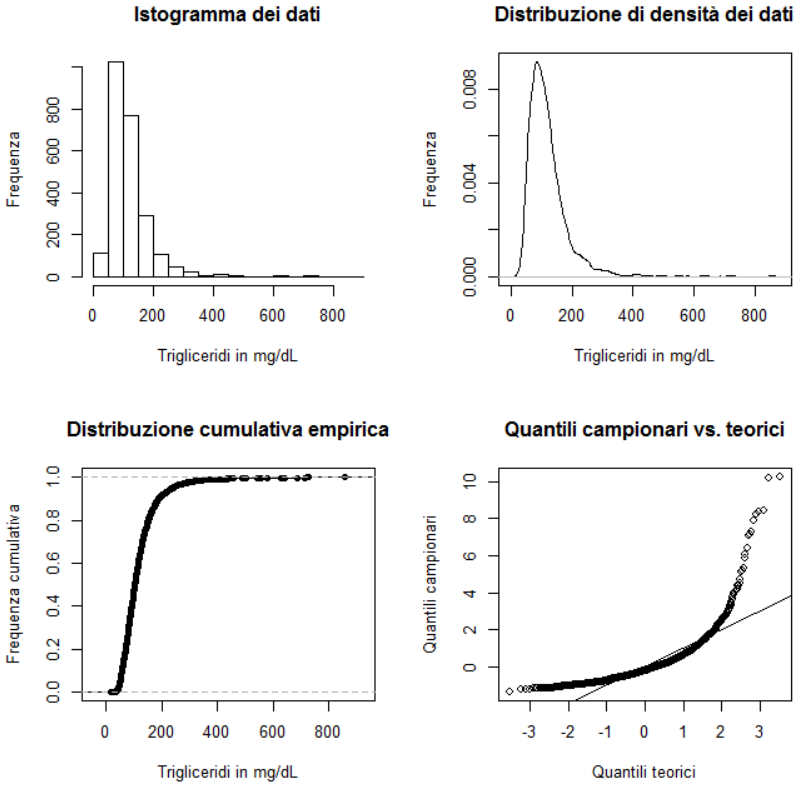


Figura 5.6 Con **R** è possibile raggruppare più grafici all'interno di una stessa figura.

Le seguenti tre varianti sul tema non sono rappresentate e sono lasciate come esercizio per familiarizzare con il tema e magari trovare altre soluzioni.

La prima variante inserisce due grafici in una riga e due colonne:

```
#
par(mfrow=c(1,2))
hist(tri, main="Istogramma dei dati", xlab="Trigliceridi in mg/dL", ylab = "Frequenza")
plot(density(tri), main="Distribuzione di densità dei dati", xlab="Trigliceridi in mg/dL", ylab = "Frequenza")
#
```

Questa seconda variante inserisce tre grafici, il primo occupa riga 1 / colonne 1 e 2, il secondo riga 2 / colonna 1, il terzo riga 2 / colonna 2:

```
#
layout(matrix(c(1,1,2,3), 2, 2, byrow = TRUE))
hist(tri, main="Istogramma dei dati", xlab="Trigliceridi in mg/dL", ylab = "Frequenza")
plot(density(tri), main="Distribuzione di densità dei dati", xlab="Trigliceridi in mg/dL", ylab = "Frequenza")
plot(ecdf(tri), main="Distribuzione cumulativa empirica", xlab="Trigliceridi in mg/dL", ylab = "Frequenza cumulativa")
#
```

Infine questa terza variante inserisce tre grafici, il primo occupa riga 1 / colonna 1, il secondo riga 2 / colonna 1, il terzo colonna 2 / righe 1 e 2:

```
#
layout(matrix(c(1,3,2,3), 2, 2, byrow = TRUE), widths=c(1,1), heights=c(1,1))
```

```
hist(tri, main="Istogramma dei dati", xlab="Trigliceridi in mg/dL", ylab = "Frequenza")
plot(density(tri), main="Distribuzione di densità dei dati", xlab="Trigliceridi in mg/dL", ylab = "Frequenza")
plot(ecdf(tri), main="Distribuzione cumulativa empirica", xlab="Trigliceridi in mg/dL", ylab = "Frequenza cumulativa")
#
```

APPENDICI

A1. Tabella dei colori di R

In **R** i colori sono ordinati secondo una numerazione progressiva da 1 a 657, e a ciascuno di essi viene associato un codice descrittivo: ad esempio il colore 8 è denominato `aquamarine1`.

A ciascun colore viene associato anche un codice in formato esadecimale, che per il colore `aquamarine1` è #7FFFD4. Caratteristica fondamentale della codifica dei colori con **R** è che sia il codice descrittivo sia il codice esadecimale possono essere utilizzati indifferentemente con il parametro **col=** per definire il colore, per esempio di un grafico.

Infine a ciascun colore viene associata una tripletta di numeri che indica, con un numero compreso tra 0 e 255 per ciascun componente della tripletta, la quantità dei tre colori fondamentali della sintesi additiva dei colori, ottenuta sommando rosso (Red), verde (Green) e blu (Blue). Indica cioè quanto ciascuno dei tre colori fondamentali contribuisce a formare il colore in questione. Per esempio per il colore `aquamarine1` ovvero #7FFFD4 la tripletta è 127 255 212 a indicare che il colore in questione è formato dalla somma di 127 parti di rosso, 255 parti di verde e di 212 parti di blu "puri" della tabella dei colori di **R**.

Nell'esempio che segue con la prima riga di codice sono generati 10000 valori di deviata normale standardizzata, corrispondenti a una distribuzione gaussiana con media 0 e deviazione standard 1. Con la seconda riga di codice viene tracciato un istogramma di colore **aquamarine1**, quindi viene aperta una nuova finestra grafica e con la quarta e ultima riga di codice viene tracciato un istogramma di colore **#7FFFD4**:

```
#
x=rnorm(10000) # genera 10000 valori distribuiti in modo gaussiano
hist(x, breaks=100, col="aquamarine1", main="colore = aquamarine1")
windows() # apre una nuova finestra
hist(x, breaks=100, col="#7FFFD4", main="colore = #7FFFD4")
#
```

Se spostate la seconda finestra e la affiancate alla prima potete constatare che il colore, a conferma di quanto si ricava dalla tabella dei colori di **R**, è identico.

Anche questo esempio dimostra come **col="orchid1"** (istogramma della seconda riga) e **col="#FF83FA"** (istogramma della quarta riga) sono espressioni equivalenti:

```
#
x=rnorm(10000) # genera 10000 valori distribuiti in modo gaussiano
hist(x, breaks=100, col="orchid1", main="colore = orchid1")
windows() # apre una nuova finestra
hist(x, breaks=100, col="#FF83FA", main="colore = #FF83FA")
#
```

Infatti se dopo averere eseguito il codice spostate la seconda finestra e la affiancate alla prima potete constatare che il colore dei due istogrammi, a conferma di quanto si ricava dalla tabella dei colori di **R**, è identico.

Nelle pagine seguenti è riportata la tabella dei colori di **R**.

R colori 1-100

1	white	#FFFFFF	255	255	255
2	aliceblue	#F0F8FF	240	248	255
3	antiquewhite	#FAEBD7	250	235	215
4	antiquewhite1	#FFEFDB	255	239	219
5	antiquewhite2	#EEDFCC	238	223	204
6	antiquewhite3	#CDC0B0	205	192	176
7	antiquewhite4	#8B8378	139	131	120
8	aquamarine	#7FFFD4	127	255	212
9	aquamarine1	#7FFFD4	127	255	212
10	aquamarine2	#76EEC6	118	238	198
11	aquamarine3	#66CDAA	102	205	170
12	aquamarine4	#458B74	69	139	116
13	azure	#F0FFFF	240	255	255
14	azure1	#F0FFFF	240	255	255
15	azure2	#E0EEEE	224	238	238
16	azure3	#C1CDCD	193	205	205
17	azure4	#838B8B	131	139	139
18	beige	#F5F5DC	245	245	220
19	bisque	#FFE4C4	255	228	196
20	bisque1	#FFE4C4	255	228	196
21	bisque2	#EED5B7	238	213	183
22	bisque3	#CDB79E	205	183	158
23	bisque4	#8B7D6B	139	125	107
24	black	#000000	0	0	0
25	blanchedalmond	#FFEBCD	255	235	205
26	blue	#0000FF	0	0	255
27	blue1	#0000FF	0	0	255
28	blue2	#0000EE	0	0	238
29	blue3	#0000CD	0	0	205
30	blue4	#00008B	0	0	139
31	blueviolet	#8A2BE2	138	43	226
32	brown	#A52A2A	165	42	42
33	brown1	#FF4040	255	64	64
34	brown2	#EE3B3B	238	59	59
35	brown3	#CD3333	205	51	51
36	brown4	#8B2323	139	35	35
37	burlywood	#DEB887	222	184	135
38	burlywood1	#FFD39B	255	211	155
39	burlywood2	#EEC591	238	197	145
40	burlywood3	#CDAA7D	205	170	125
41	burlywood4	#8B7355	139	115	85
42	cadetblue	#5F9EA0	95	158	160
43	cadetblue1	#98F5FF	152	245	255
44	cadetblue2	#8EE5EE	142	229	238
45	cadetblue3	#7AC5CD	122	197	205
46	cadetblue4	#53868B	83	134	139
47	chartreuse	#7FFF00	127	255	0
48	chartreuse1	#7FFF00	127	255	0
49	chartreuse2	#76EE00	118	238	0
50	chartreuse3	#66CD00	102	205	0
51	chartreuse4	#458B00	69	139	0
52	chocolate	#D2691E	210	105	30
53	chocolate1	#FF7F24	255	127	36
54	chocolate2	#EE7621	238	118	33
55	chocolate3	#CD661D	205	102	29
56	chocolate4	#8B4513	139	69	19
57	coral	#FF7F50	255	127	80
58	coral1	#FF7256	255	114	86
59	coral2	#EE6A50	238	106	80
60	coral3	#CD5B45	205	91	69
61	coral4	#8B3B2F	139	62	47
62	cornflowerblue	#6495ED	100	149	237
63	cornsilk	#FFF8DC	255	248	220
64	cornsilk1	#FFF8DC	255	248	220
65	cornsilk2	#EEE8CD	238	232	205
66	cornsilk3	#CDC8B1	205	200	177
67	cornsilk4	#8B8878	139	136	120
68	cyan	#00FFFF	0	255	255
69	cyan1	#00FFFF	0	255	255
70	cyan2	#00EEEE	0	238	238
71	cyan3	#00CDCD	0	205	205
72	cyan4	#008B8B	0	139	139
73	darkblue	#00008B	0	0	139
74	darkcyan	#008B8B	0	139	139
75	darkgoldenrod	#B8860B	184	134	11
76	darkgoldenrod1	#FFB90F	255	185	15
77	darkgoldenrod2	#EEAD0E	238	173	14
78	darkgoldenrod3	#CD950C	205	149	12
79	darkgoldenrod4	#8B6508	139	101	8
80	darkgray	#A9A9A9	169	169	169
81	darkgreen	#006400	0	100	0
82	darkgrey	#A9A9A9	169	169	169
83	darkkhaki	#BDB76B	189	183	107
84	darkmagenta	#8B008B	139	0	139
85	darkolivegreen	#556B2F	85	107	47
86	darkolivegreen1	#CAFF70	202	255	112
87	darkolivegreen2	#BCEE68	188	238	104
88	darkolivegreen3	#A2CD5A	162	205	90
89	darkolivegreen4	#6E8B3D	110	139	61
90	darkorange	#FF8C00	255	140	0
91	darkorange1	#FF7F00	255	127	0
92	darkorange2	#EE7600	238	118	0
93	darkorange3	#CD6600	205	102	0
94	darkorange4	#8B4500	139	69	0
95	darkorchid	#9932CC	153	50	204
96	darkorchid1	#BF3EFF	191	62	255
97	darkorchid2	#B23AEE	178	58	238
98	darkorchid3	#9A32CD	154	50	205
99	darkorchid4	#68228B	104	34	139
100	darkred	#8B0000	139	0	0

R colori 101-200

101	darksalmon	#E9967A	233	150	122
102	darkseagreen	#8FBC8F	143	188	143
103	darkseagreen1	#C1FFC1	193	255	193
104	darkseagreen2	#B4EEB4	180	238	180
105	darkseagreen3	#9BCD9B	155	205	155
106	darkseagreen4	#698B69	105	139	105
107	darkslateblue	#483D8B	72	61	139
108	darkslategray	#2F4F4F	47	79	79
109	darkslategray1	#97FFFF	151	255	255
110	darkslategray2	#8DEEEE	141	238	238
111	darkslategray3	#79CDCD	121	205	205
112	darkslategray4	#528B8B	82	139	139
113	darkslategrey	#2F4F4F	47	79	79
114	darkturquoise	#00CED1	0	206	209
115	darkviolet	#9400D3	148	0	211
116	deeppink	#FF1493	255	20	147
117	deeppink1	#FF1493	255	20	147
118	deeppink2	#EE1289	238	18	137
119	deeppink3	#CD1076	205	16	118
120	deeppink4	#8B0A50	139	10	80
121	deepskyblue	#00BFFF	0	191	255
122	deepskyblue1	#00BFFF	0	191	255
123	deepskyblue2	#00B2EE	0	178	238
124	deepskyblue3	#009ACD	0	154	205
125	deepskyblue4	#00688B	0	104	139
126	dimgray	#696969	105	105	105
127	dimgrey	#696969	105	105	105
128	dodgerblue	#1E90FF	30	144	255
129	dodgerblue1	#1E90FF	30	144	255
130	dodgerblue2	#1C86EE	28	134	238
131	dodgerblue3	#1874CD	24	116	205
132	dodgerblue4	#104E8B	16	78	139
133	firebrick	#B22222	178	34	34
134	firebrick1	#FF3030	255	48	48
135	firebrick2	#EE2C2C	238	44	44
136	firebrick3	#CD2626	205	38	38
137	firebrick4	#8B1A1A	139	26	26
138	floralwhite	#FFFAF0	255	250	240
139	forestgreen	#228B22	34	139	34
140	gainsboro	#DCDCDC	220	220	220
141	ghostwhite	#F8F8FF	248	248	255
142	gold	#FFD700	255	215	0
143	gold1	#FFD700	255	215	0
144	gold2	#EEC900	238	201	0
145	gold3	#CDAD00	205	173	0
146	gold4	#8B7500	139	117	0
147	goldenrod	#DAA520	218	165	32
148	goldenrod1	#FFC125	255	193	37
149	goldenrod2	#EEB422	238	180	34
150	goldenrod3	#CD9B1D	205	155	29
151	goldenrod4	#8B6914	139	105	20
152	gray	#BEBEBE	190	190	190
153	gray0	#000000	0	0	0
154	gray1	#030303	3	3	3
155	gray2	#050505	5	5	5
156	gray3	#080808	8	8	8
157	gray4	#0A0A0A	10	10	10
158	gray5	#0D0D0D	13	13	13
159	gray6	#0F0F0F	15	15	15
160	gray7	#121212	18	18	18
161	gray8	#141414	20	20	20
162	gray9	#171717	23	23	23
163	gray10	#1A1A1A	26	26	26
164	gray11	#1C1C1C	28	28	28
165	gray12	#1F1F1F	31	31	31
166	gray13	#212121	33	33	33
167	gray14	#242424	36	36	36
168	gray15	#262626	38	38	38
169	gray16	#292929	41	41	41
170	gray17	#2B2B2B	43	43	43
171	gray18	#2E2E2E	46	46	46
172	gray19	#303030	48	48	48
173	gray20	#333333	51	51	51
174	gray21	#363636	54	54	54
175	gray22	#383838	56	56	56
176	gray23	#3B3B3B	59	59	59
177	gray24	#3D3D3D	61	61	61
178	gray25	#404040	64	64	64
179	gray26	#424242	66	66	66
180	gray27	#454545	69	69	69
181	gray28	#474747	71	71	71
182	gray29	#4A4A4A	74	74	74
183	gray30	#4D4D4D	77	77	77
184	gray31	#4F4F4F	79	79	79
185	gray32	#525252	82	82	82
186	gray33	#545454	84	84	84
187	gray34	#575757	87	87	87
188	gray35	#595959	89	89	89
189	gray36	#5C5C5C	92	92	92
190	gray37	#5E5E5E	94	94	94
191	gray38	#616161	97	97	97
192	gray39	#636363	99	99	99
193	gray40	#666666	102	102	102
194	gray41	#696969	105	105	105
195	gray42	#6B6B6B	107	107	107
196	gray43	#6E6E6E	110	110	110
197	gray44	#707070	112	112	112
198	gray45	#737373	115	115	115
199	gray46	#757575	117	117	117
200	gray47	#787878	120	120	120

R colori 201-300

201	gray48	#7A7A7A	122	122	122
202	gray49	#7D7D7D	125	125	125
203	gray50	#7F7F7F	127	127	127
204	gray51	#828282	130	130	130
205	gray52	#858585	133	133	133
206	gray53	#878787	135	135	135
207	gray54	#8A8A8A	138	138	138
208	gray55	#8C8C8C	140	140	140
209	gray56	#8F8F8F	143	143	143
210	gray57	#919191	145	145	145
211	gray58	#949494	148	148	148
212	gray59	#969696	150	150	150
213	gray60	#999999	153	153	153
214	gray61	#9C9C9C	156	156	156
215	gray62	#9E9E9E	158	158	158
216	gray63	#A1A1A1	161	161	161
217	gray64	#A3A3A3	163	163	163
218	gray65	#A6A6A6	166	166	166
219	gray66	#A8A8A8	168	168	168
220	gray67	#ABABAB	171	171	171
221	gray68	#ADADAD	173	173	173
222	gray69	#B0B0B0	176	176	176
223	gray70	#B3B3B3	179	179	179
224	gray71	#B5B5B5	181	181	181
225	gray72	#B8B8B8	184	184	184
226	gray73	#BABABA	186	186	186
227	gray74	#BDBDBD	189	189	189
228	gray75	#BFBFBF	191	191	191
229	gray76	#C2C2C2	194	194	194
230	gray77	#C4C4C4	196	196	196
231	gray78	#C7C7C7	199	199	199
232	gray79	#C9C9C9	201	201	201
233	gray80	#CCCCCC	204	204	204
234	gray81	#CFCFCF	207	207	207
235	gray82	#D1D1D1	209	209	209
236	gray83	#D4D4D4	212	212	212
237	gray84	#D6D6D6	214	214	214
238	gray85	#D9D9D9	217	217	217
239	gray86	#DBDBDB	219	219	219
240	gray87	#DEDEDE	222	222	222
241	gray88	#E0E0E0	224	224	224
242	gray89	#E3E3E3	227	227	227
243	gray90	#E5E5E5	229	229	229
244	gray91	#E8E8E8	232	232	232
245	gray92	#EBEBEB	235	235	235
246	gray93	#EDEDED	237	237	237
247	gray94	#F0F0F0	240	240	240
248	gray95	#F2F2F2	242	242	242
249	gray96	#F5F5F5	245	245	245
250	gray97	#F7F7F7	247	247	247
251	gray98	#FAFAFA	250	250	250
252	gray99	#FCFCFC	252	252	252
253	gray100	#FFFFFF	255	255	255
254	green	#00FF00	0	255	0
255	green1	#00FF00	0	255	0
256	green2	#00EE00	0	238	0
257	green3	#00CD00	0	205	0
258	green4	#008B00	0	139	0
259	greenyellow	#ADFF2F	173	255	47
260	grey	#BEBEBE	190	190	190
261	grey0	#000000	0	0	0
262	grey1	#030303	3	3	3
263	grey2	#050505	5	5	5
264	grey3	#080808	8	8	8
265	grey4	#0A0A0A	10	10	10
266	grey5	#0D0D0D	13	13	13
267	grey6	#0F0F0F	15	15	15
268	grey7	#121212	18	18	18
269	grey8	#141414	20	20	20
270	grey9	#171717	23	23	23
271	grey10	#1A1A1A	26	26	26
272	grey11	#1C1C1C	28	28	28
273	grey12	#1F1F1F	31	31	31
274	grey13	#212121	33	33	33
275	grey14	#242424	36	36	36
276	grey15	#262626	38	38	38
277	grey16	#292929	41	41	41
278	grey17	#2B2B2B	43	43	43
279	grey18	#2E2E2E	46	46	46
280	grey19	#303030	48	48	48
281	grey20	#333333	51	51	51
282	grey21	#363636	54	54	54
283	grey22	#383838	56	56	56
284	grey23	#3B3B3B	59	59	59
285	grey24	#3D3D3D	61	61	61
286	grey25	#404040	64	64	64
287	grey26	#424242	66	66	66
288	grey27	#454545	69	69	69
289	grey28	#474747	71	71	71
290	grey29	#4A4A4A	74	74	74
291	grey30	#4D4D4D	77	77	77
292	grey31	#4F4F4F	79	79	79
293	grey32	#525252	82	82	82
294	grey33	#545454	84	84	84
295	grey34	#575757	87	87	87
296	grey35	#595959	89	89	89
297	grey36	#5C5C5C	92	92	92
298	grey37	#5E5E5E	94	94	94
299	grey38	#616161	97	97	97
300	grey39	#636363	99	99	99

R colori 301-400

301	grey40	#666666	102 102 102
302	grey41	#696969	105 105 105
303	grey42	#6B6B6B	107 107 107
304	grey43	#6E6E6E	110 110 110
305	grey44	#707070	112 112 112
306	grey45	#737373	115 115 115
307	grey46	#757575	117 117 117
308	grey47	#787878	120 120 120
309	grey48	#7A7A7A	122 122 122
310	grey49	#7D7D7D	125 125 125
311	grey50	#7F7F7F	127 127 127
312	grey51	#828282	130 130 130
313	grey52	#858585	133 133 133
314	grey53	#878787	135 135 135
315	grey54	#8A8A8A	138 138 138
316	grey55	#8C8C8C	140 140 140
317	grey56	#8F8F8F	143 143 143
318	grey57	#919191	145 145 145
319	grey58	#949494	148 148 148
320	grey59	#969696	150 150 150
321	grey60	#999999	153 153 153
322	grey61	#9C9C9C	156 156 156
323	grey62	#9E9E9E	158 158 158
324	grey63	#A1A1A1	161 161 161
325	grey64	#A3A3A3	163 163 163
326	grey65	#A6A6A6	166 166 166
327	grey66	#A8A8A8	168 168 168
328	grey67	#ABABAB	171 171 171
329	grey68	#ADADAD	173 173 173
330	grey69	#B0B0B0	176 176 176
331	grey70	#B3B3B3	179 179 179
332	grey71	#B5B5B5	181 181 181
333	grey72	#B8B8B8	184 184 184
334	grey73	#BABABA	186 186 186
335	grey74	#BDBDBD	189 189 189
336	grey75	#BFBFBF	191 191 191
337	grey76	#C2C2C2	194 194 194
338	grey77	#C4C4C4	196 196 196
339	grey78	#C7C7C7	199 199 199
340	grey79	#C9C9C9	201 201 201
341	grey80	#CCCCCC	204 204 204
342	grey81	#CFCFCF	207 207 207
343	grey82	#D1D1D1	209 209 209
344	grey83	#D4D4D4	212 212 212
345	grey84	#D6D6D6	214 214 214
346	grey85	#D9D9D9	217 217 217
347	grey86	#DBDBDB	219 219 219
348	grey87	#DEDEDE	222 222 222
349	grey88	#E0E0E0	224 224 224
350	grey89	#E3E3E3	227 227 227
351	grey90	#E5E5E5	229 229 229
352	grey91	#E8E8E8	232 232 232
353	grey92	#EBEBEB	235 235 235
354	grey93	#EDEDED	237 237 237
355	grey94	#F0F0F0	240 240 240
356	grey95	#F2F2F2	242 242 242
357	grey96	#F5F5F5	245 245 245
358	grey97	#F7F7F7	247 247 247
359	grey98	#FAFAFA	250 250 250
360	grey99	#FCFCFC	252 252 252
361	grey100	#FFFFFF	255 255 255
362	honeydew	#F0FFF0	240 255 240
363	honeydew1	#F0FFF0	240 255 240
364	honeydew2	#E0EEE0	224 238 224
365	honeydew3	#C1CDC1	193 205 193
366	honeydew4	#838B83	131 139 131
367	hotpink	#FF69B4	255 105 180
368	hotpink1	#FF6EB4	255 110 180
369	hotpink2	#EE6AA7	238 106 167
370	hotpink3	#CD6090	205 96 144
371	hotpink4	#8B3A62	139 58 98
372	indianred	#CD5C5C	205 92 92
373	indianred1	#FF6A6A	255 106 106
374	indianred2	#EE6363	238 99 99
375	indianred3	#CD5555	205 85 85
376	indianred4	#8B3A3A	139 58 58
377	ivory	#FFFFF0	255 255 240
378	ivory1	#FFFFF0	255 255 240
379	ivory2	#EEEEE0	238 238 224
380	ivory3	#CDCDC1	205 205 193
381	ivory4	#8B8B83	139 139 131
382	khaki	#F0E68C	240 230 140
383	khaki1	#FFF68F	255 246 143
384	khaki2	#EEE685	238 230 133
385	khaki3	#CDC673	205 198 115
386	khaki4	#8B864E	139 134 78
387	lavender	#E6E6FA	230 230 250
388	lavenderblush	#FFF0F5	255 240 245
389	lavenderblush1	#FFF0F5	255 240 245
390	lavenderblush2	#EEE0E5	238 224 229
391	lavenderblush3	#CDC1C5	205 193 197
392	lavenderblush4	#8B8386	139 131 134
393	lawngreen	#7CFC00	124 252 0
394	lemonchiffon	#FFFACD	255 250 205
395	lemonchiffon1	#FFFACD	255 250 205
396	lemonchiffon2	#EEE9BF	238 233 191
397	lemonchiffon3	#CDC9A5	205 201 165
398	lemonchiffon4	#8B8970	139 137 112
399	lightblue	#ADD8E6	173 216 230
400	lightblue1	#BFEFFF	191 239 255

R colori 401-500

401	lightblue2	#B2DFEE	178	223	238
402	lightblue3	#9AC0CD	154	192	205
403	lightblue4	#68838B	104	131	139
404	lightcoral	#F08080	240	128	128
405	lightcyan	#E0FFFF	224	255	255
406	lightcyan1	#E0FFFF	224	255	255
407	lightcyan2	#D1EEEE	209	238	238
408	lightcyan3	#B4CDCD	180	205	205
409	lightcyan4	#7A8B8B	122	139	139
410	lightgoldenrod	#EEDD82	238	221	130
411	lightgoldenrod1	#FFEC8B	255	236	139
412	lightgoldenrod2	#EEDC82	238	220	130
413	lightgoldenrod3	#CDBE70	205	190	112
414	lightgoldenrod4	#8B814C	139	129	76
415	lightgoldenrodyellow	#FAFAD2	250	250	210
416	lightgray	#D3D3D3	211	211	211
417	lightgreen	#90EE90	144	238	144
418	lightgrey	#D3D3D3	211	211	211
419	lightpink	#FFB6C1	255	182	193
420	lightpink1	#FFAEB9	255	174	185
421	lightpink2	#EEA2AD	238	162	173
422	lightpink3	#CD8C95	205	140	149
423	lightpink4	#8B5F65	139	95	101
424	lightsalmon	#FFA07A	255	160	122
425	lightsalmon1	#FFA07A	255	160	122
426	lightsalmon2	#EE9572	238	149	114
427	lightsalmon3	#CD8162	205	129	98
428	lightsalmon4	#8B5742	139	87	66
429	lightseagreen	#20B2AA	32	178	170
430	lightskyblue	#87CEFA	135	206	250
431	lightskyblue1	#B0E2FF	176	226	255
432	lightskyblue2	#A4D3EE	164	211	238
433	lightskyblue3	#8DB6CD	141	182	205
434	lightskyblue4	#607B8B	96	123	139
435	lightslateblue	#8470FF	132	112	255
436	lightslategray	#778899	119	136	153
437	lightslategrey	#778899	119	136	153
438	lightsteelblue	#B0C4DE	176	196	222
439	lightsteelblue1	#CAE1FF	202	225	255
440	lightsteelblue2	#BCD2EE	188	210	238
441	lightsteelblue3	#A2B5CD	162	181	205
442	lightsteelblue4	#6E7B8B	110	123	139
443	lightyellow	#FFFFE0	255	255	224
444	lightyellow1	#FFFFE0	255	255	224
445	lightyellow2	#EEEED1	238	238	209
446	lightyellow3	#CDCDB4	205	205	180
447	lightyellow4	#8B8B7A	139	139	122
448	limegreen	#32CD32	50	205	50
449	linen	#FAF0E6	250	240	230
450	magenta	#FF00FF	255	0	255
451	magenta1	#FF00FF	255	0	255
452	magenta2	#EE00EE	238	0	238
453	magenta3	#CD00CD	205	0	205
454	magenta4	#8B008B	139	0	139
455	maroon	#B03060	176	48	96
456	maroon1	#FF34B3	255	52	179
457	maroon2	#EE30A7	238	48	167
458	maroon3	#CD2990	205	41	144
459	maroon4	#8B1C62	139	28	98
460	mediumaquamarine	#66CDAA	102	205	170
461	mediumblue	#0000CD	0	0	205
462	mediumorchid	#BA55D3	186	85	211
463	mediumorchid1	#E066FF	224	102	255
464	mediumorchid2	#D15FEE	209	95	238
465	mediumorchid3	#B452CD	180	82	205
466	mediumorchid4	#7A378B	122	55	139
467	mediumpurple	#9370DB	147	112	219
468	mediumpurple1	#AB82FF	171	130	255
469	mediumpurple2	#9F79EE	159	121	238
470	mediumpurple3	#8968CD	137	104	205
471	mediumpurple4	#5D478B	93	71	139
472	mediumseagreen	#3CB371	60	179	113
473	mediumslateblue	#7B68EE	123	104	238
474	mediumspringgreen	#00FA9A	0	250	154
475	mediumturquoise	#48D1CC	72	209	204
476	mediumvioletred	#C71585	199	21	133
477	midnightblue	#191970	25	25	112
478	mintcream	#F5FFFA	245	255	250
479	mistyrose	#FFE4E1	255	228	225
480	mistyrose1	#FFE4E1	255	228	225
481	mistyrose2	#EED5D2	238	213	210
482	mistyrose3	#CDB7B5	205	183	181
483	mistyrose4	#8B7D7B	139	125	123
484	moccasin	#FFE4B5	255	228	181
485	navajowhite	#FFDEAD	255	222	173
486	navajowhite1	#FFDEAD	255	222	173
487	navajowhite2	#EECFA1	238	207	161
488	navajowhite3	#CDB38B	205	179	139
489	navajowhite4	#8B795E	139	121	94
490	navy	#000080	0	0	128
491	navyblue	#000080	0	0	128
492	oldlace	#FDF5E6	253	245	230
493	olivedrab	#6B8E23	107	142	35
494	olivedrab1	#C0FF3E	192	255	62
495	olivedrab2	#B3EE3A	179	238	58
496	olivedrab3	#9ACD32	154	205	50
497	olivedrab4	#698B22	105	139	34
498	orange	#FFA500	255	165	0
499	orange1	#FFA500	255	165	0
500	orange2	#EE9A00	238	154	0

R colori 501-600

501	orange3	#CD8500	205	133	0
502	orange4	#8B5A00	139	90	0
503	orangered	#FF4500	255	69	0
504	orangered1	#FF4500	255	69	0
505	orangered2	#EE4000	238	64	0
506	orangered3	#CD3700	205	55	0
507	orangered4	#8B2500	139	37	0
508	orchid	#DA70D6	218	112	214
509	orchid1	#FF83FA	255	131	250
510	orchid2	#EE7AE9	238	122	233
511	orchid3	#CD69C9	205	105	201
512	orchid4	#8B4789	139	71	137
513	palegoldenrod	#EEE8AA	238	232	170
514	palegreen	#98FB98	152	251	152
515	palegreen1	#9AFF9A	154	255	154
516	palegreen2	#90EE90	144	238	144
517	palegreen3	#7CCD7C	124	205	124
518	palegreen4	#548B54	84	139	84
519	paleturquoise	#AFEEEE	175	238	238
520	paleturquoise1	#BBFFFF	187	255	255
521	paleturquoise2	#AEEEEE	174	238	238
522	paleturquoise3	#96CDCD	150	205	205
523	paleturquoise4	#668B8B	102	139	139
524	palevioletred	#DB7093	219	112	147
525	palevioletred1	#FF82AB	255	130	171
526	palevioletred2	#EE799F	238	121	159
527	palevioletred3	#CD6889	205	104	137
528	palevioletred4	#8B475D	139	71	93
529	papayawhip	#FFEFD5	255	239	213
530	peachpuff	#FFDAB9	255	218	185
531	peachpuff1	#FFDAB9	255	218	185
532	peachpuff2	#EECBAD	238	203	173
533	peachpuff3	#CDAF95	205	175	149
534	peachpuff4	#8B7765	139	119	101
535	peru	#CD853F	205	133	63
536	pink	#FFC0CB	255	192	203
537	pink1	#FFB5C5	255	181	197
538	pink2	#EEA9B8	238	169	184
539	pink3	#CD919E	205	145	158
540	pink4	#8B636C	139	99	108
541	plum	#DDA0DD	221	160	221
542	plum1	#FFBBFF	255	187	255
543	plum2	#EEAEEE	238	174	238
544	plum3	#CD96CD	205	150	205
545	plum4	#8B668B	139	102	139
546	powderblue	#B0E0E6	176	224	230
547	purple	#A020F0	160	32	240
548	purple1	#9B30FF	155	48	255
549	purple2	#912CEE	145	44	238
550	purple3	#7D26CD	125	38	205
551	purple4	#551A8B	85	26	139
552	red	#FF0000	255	0	0
553	red1	#FF0000	255	0	0
554	red2	#EE0000	238	0	0
555	red3	#CD0000	205	0	0
556	red4	#8B0000	139	0	0
557	rosybrown	#BC8F8F	188	143	143
558	rosybrown1	#FFC1C1	255	193	193
559	rosybrown2	#EEB4B4	238	180	180
560	rosybrown3	#CD9B9B	205	155	155
561	rosybrown4	#8B6969	139	105	105
562	royalblue	#4169E1	65	105	225
563	royalblue1	#4876FF	72	118	255
564	royalblue2	#436EEE	67	110	238
565	royalblue3	#3A5FCD	58	95	205
566	royalblue4	#27408B	39	64	139
567	saddlebrown	#8B4513	139	69	19
568	salmon	#FA8072	250	128	114
569	salmon1	#FF8C69	255	140	105
570	salmon2	#EE8262	238	130	98
571	salmon3	#CD7054	205	112	84
572	salmon4	#8B4C39	139	76	57
573	sandybrown	#F4A460	244	164	96
574	seagreen	#2E8B57	46	139	87
575	seagreen1	#54FF9F	84	255	159
576	seagreen2	#4EEE94	78	238	148
577	seagreen3	#43CD80	67	205	128
578	seagreen4	#2E8B57	46	139	87
579	seashell	#FFF5EE	255	245	238
580	seashell1	#FFF5EE	255	245	238
581	seashell2	#EEE5DE	238	229	222
582	seashell3	#CDC5BF	205	197	191
583	seashell4	#8B8682	139	134	130
584	sienna	#A0522D	160	82	45
585	sienna1	#FF8247	255	130	71
586	sienna2	#EE7942	238	121	66
587	sienna3	#CD6839	205	104	57
588	sienna4	#8B4726	139	71	38
589	skyblue	#87CEEB	135	206	235
590	skyblue1	#87CEFF	135	206	255
591	skyblue2	#7EC0EE	126	192	238
592	skyblue3	#6CA6CD	108	166	205
593	skyblue4	#4A708B	74	112	139
594	slateblue	#6A5ACD	106	90	205
595	slateblue1	#836FFF	131	111	255
596	slateblue2	#7A67EE	122	103	238
597	slateblue3	#6959CD	105	89	205
598	slateblue4	#473C8B	71	60	139
599	slategray	#708090	112	128	144
600	slategray1	#C6E2FF	198	226	255

R colori 601-657

601	slategray2	#B9D3EE	185	211	238
602	slategray3	#9FB6CD	159	182	205
603	slategray4	#6C7B8B	108	123	139
604	slategrey	#708090	112	128	144
605	snow	#FFFAFA	255	250	250
606	snow1	#FFFAFA	255	250	250
607	snow2	#EEE9E9	238	233	233
608	snow3	#CDC9C9	205	201	201
609	snow4	#8B8989	139	137	137
610	springgreen	#00FF7F	0	255	127
611	springgreen1	#00FF7F	0	255	127
612	springgreen2	#00EE76	0	238	118
613	springgreen3	#00CD66	0	205	102
614	springgreen4	#008B45	0	139	69
615	steelblue	#4682B4	70	130	180
616	steelblue1	#63B8FF	99	184	255
617	steelblue2	#5CACEE	92	172	238
618	steelblue3	#4F94CD	79	148	205
619	steelblue4	#36648B	54	100	139
620	tan	#D2B48C	210	180	140
621	tan1	#FFA54F	255	165	79
622	tan2	#EE9A49	238	154	73
623	tan3	#CD853F	205	133	63
624	tan4	#8B5A2B	139	90	43
625	thistle	#D8BFD8	216	191	216
626	thistle1	#FFE1FF	255	225	255
627	thistle2	#EED2EE	238	210	238
628	thistle3	#CDB5CD	205	181	205
629	thistle4	#8B7B8B	139	123	139
630	tomato	#FF6347	255	99	71
631	tomato1	#FF6347	255	99	71
632	tomato2	#EE5C42	238	92	66
633	tomato3	#CD4F39	205	79	57
634	tomato4	#8B3626	139	54	38
635	turquoise	#40E0D0	64	224	208
636	turquoise1	#00F5FF	0	245	255
637	turquoise2	#00E5EE	0	229	238
638	turquoise3	#00C5CD	0	197	205
639	turquoise4	#00868B	0	134	139
640	violet	#EE82EE	238	130	238
641	violetred	#D02090	208	32	144
642	violetred1	#FF3E96	255	62	150
643	violetred2	#EE3A8C	238	58	140
644	violetred3	#CD3278	205	50	120
645	violetred4	#8B2252	139	34	82
646	wheat	#F5DEB3	245	222	179
647	wheat1	#FFE7BA	255	231	186
648	wheat2	#EED8AE	238	216	174
649	wheat3	#CDBA96	205	186	150
650	wheat4	#8B7E66	139	126	102

651	whitesmoke	#F5F5F5	245	245	245
652	yellow	#FFFF00	255	255	0
653	yellow1	#FFFF00	255	255	0
654	yellow2	#EEEE00	238	238	0
655	yellow3	#CDCD00	205	205	0
656	yellow4	#8B8B00	139	139	0
657	yellowgreen	#9ACD32	154	205	50