STATISTICA E ANALISI DEI DATI (PRIMA PARTE)

Amelia Giuseppina Nobile[1]

(a.a. 2022/2023)

3 novembre 2022

[1]Dipartimento di Informatica, Università degli Studi di Salerno

Capitolo 1

L'ambiente integrato R

1.1 Introduzione e note storiche

R è contemporaneamente un linguaggio di programmazione ed un software, costituito da una varietà di strumenti per l'analisi statistica dei dati e per la loro visualizzazione. R è quindi un ambiente integrato che permette di elaborare dati, eseguire calcoli e produrre grafici.

R deriva dal linguaggio di programmazione di statistica S, sviluppato negli anni '80 da AT&T Bell Laboratories da Rick A. Becker, John Chambers e Allan Wilks[1], che ha dato origine ad un noto software commerciale S-Plus, prodotto da Insightful Corporation e successivamente da TIBCO Software Inc. [2]

La versione iniziale di R è stata realizzata nel 1996 da Ross Ihaka e Robert Gentleman[3] del Dipartimento di Statistica dell'Università di Auckland in Nuova Zelanda. Successivamente, un nutrito gruppo di ricercatori operanti in ambito statistico e informatico hanno iniziato a fornire il loro contributo, dando così vita al "R Development Core Team", che dal 1997 si occupa dei codici sorgenti di R. Nel 2003 è stata costituita l'organizzazione non profit "R Foundation for Statistical Computing" avente come obiettivo quello di promuovere lo sviluppo e la diffusione del software, di gestire e tutelare il copyright di R e della relativa documentazione.

R è un software *open source* scaricabile gratuitamente da Internet sotto la licenza GPL (General Public License). Sul sito del progetto "The R Project for Statistical Computing", la cui home page è *http://www.r-project.org/*, è possibile trovare ogni tipo di supporto per l'utilizzo di R. Inoltre dal sito di "The Comprehensive R Archive Network (CRAN)", il cui indirizzo Internet è *http://cran.r-project.org/*, è possibile accedere ai numerosi mirror del CRAN per effettuare il download del software e di tutta la documentazione collegata

[1] Becher, R.A., Chambers, J.M., Wilks, A.R. *The New S Language*, The Blue Book, Chapman & Hall, London, 1988.

[2] Venables W.N., Ripley, B.D. *Modern Applied Statistics with S* (Fourth edition), Springer, 2002.

[3] Ihaka R., Gentleman R. *R: A Language for Data Analysis and Graphics*, Journal of Computational and Graphical Statistics, **5(3)**, 299-314, 1996.

per diversi sistemi operativi: Linux, Windows, MacOS. Da tale sito è possibile scaricare, oltre che il modulo base, anche una vasta gamma di package aggiuntivi utilizzabili per la risoluzione di specifici problemi o per particolari analisi statistiche.

Il linguaggio R è stato sviluppato in modo tale da mantenere la massima compatibilità con il software commerciale S-Plus e, sempre più spesso, soprattutto in ambienti universitari, si preferisce installare licenze gratuite di R.

Le caratteristiche principali di R possono essere così sintetizzate:

1. è un software *open source*;
2. è un linguaggio di programmazione *object-oriented* (come C++ e Java) e quindi l'utente finale può avere accesso al codice interno di R ed eventualmente proporne modifiche;
3. è un linguaggio di programmazione *interpretato* e quindi il sistema è in grado di elaborare le frasi inserite immediatamente, senza dover passare attraverso un processo di compilazione;
4. è un software *multipiattaforma*, osia può essere installato su Linux, Windows o Mac;
5. è semplice da utilizzare nella gestione e nella manipolazione dei dati;
6. è dotato di notevoli e particolarmente flessibili potenzialità grafiche 2D e 3D consentendo la rappresentazione grafica di dati;
7. dispone di un insieme di strumenti per il calcolo su vettori, matrici, data frame e per altre operazioni complesse;
8. consente l'accesso a un vasto insieme di strumenti integrati per analisi statistiche;
9. fornisce la possibilità di programmare, creando funzioni e programmi ad hoc definiti dall'utente;
10. è dotato di una funzione di *help in linea* per ciascun comando facilmente richiamabile dal programma;
11. possiede numerosi data frame di esempio e ottimi manuali di riferimento (in lingua inglese) consultabili o scaricabili direttamente da Internet (*http://www.cran.r-project.org/manuals.html*).

R eredita da S le caratteristiche di essere un linguaggio interpretato e di tipo object-oriented. Queste due caratteristiche rendono il linguaggio molto flessibile e facilmente estendibile attraverso la creazione di nuove funzioni definite dall'utente. R consente di programmare calcoli matematici e statistici e di effettuare analisi e computazioni anche molto complesse.

Nel linguaggio R tutto viene rappresentato mediante *oggetti*. Ogni oggetto (vettore, dataset, tabella, grafico, ...) è trattato dalle funzioni di R con

uno specifico metodo ed è possibile implementare nuovi metodi per ampliare le possibilità delle stesse funzioni.

Il cuore di R è rappresentato dal *modulo base* che offre gli strumenti fondamentali per effettuare le usuali operazioni di lettura e scrittura dei dati da e su file, le operazioni su matrici e vettori e le elaborazioni statistiche connesse alla statistica descrittiva e inferenziale, alla regressione, all'analisi esplorativa dei dati, alla produzione di grafici e alla simulazione di variabili aleatorie. Alcune librerie sono già comprese nel modulo base, mentre altre librerie possono essere aggiunte e installate successivamente in base alle necessità. Alcuni *package aggiuntivi*, disponibili sul sito del CRAN (CRAN Task Views), sono i seguenti 39:

- Bayesian (Bayesian Inference);
- CausalInference (Causal Inference);
- ChemPhys (Chemometrics and Computational Physics);
- ClinicalTrials (Clinical Trial Design, Monitoring, and Analysis);
- Cluster (Cluster Analysis & Finite Mixture Models);
- Databases (Databases with R);
- DifferentialEquations (Differential Equations);
- Distributions (Probability distributions);
- Econometrics (Econometrics);
- Environmetrics (Analysis of Ecological and Environmental Data);
- Epidemiology (Epidemiology);
- ExperimentalDesign (Design of Experiments (DoE) & Analysis of Experimental Data);
- ExtremeValue (Extreme Value Analysis);
- Finance (Empirical Finance);
- FunctionalData (Functional Data Analysis);
- GraphicalModels (Graphical Models);
- HighPerformanceComputing (High-Performance and Parallel Computing with R);
- Hydrology (Hydrological Data and Modeling);
- MachineLearning (Machine Learning & Statistical Learning);
- MedicalImaging (Medical Image Analysis);

- MetaAnalysis (Meta-Analysis);

- Missing Data (Missing Data);

- ModelDeployment (Model Deployment with R);

- NaturalLanguageProcessing (Natural Language Processing);

- NumericalMathematics (Numerical Mathematics);

- OfficialStatistics (Official Statistics & Survey Methodology);

- Optimization (Optimization and Mathematical Programming);

- Pharmacokinetics (Analysis of Pharmacokinetic Data);

- Psychometrics (Psychometric Models and Methods);

- ReproducibleResearch (Reproducible Research);

- Robust (Robust Statistical Methods);

- Spatial (Analysis of Spatial Data);

- SpatioTemporal (Handling and Analyzing Spatio-Temporal Data)

- SportsAnalytics (Sports Analytics);

- Survival (Survival Analysis);

- Teaching Statistics (Teaching Statistics);

- TimeSeries (Time Series Analysis);

- Tracking (Processing and Analysis of Tracking Data);

- WebTechnologies (Web Technologies and Services).

Anche se il linguaggio è fornito con un'interfaccia a linea di comando, sono disponibili diverse interfacce grafiche che consentono di integrare R, tra cui *RStudio* il cui download può essere effettuato da *https://www.rstudio.com/*. Dal 2009 è presente in rete una rivista elettronica dal titolo "The R Journal".

1.2 Alcune differenze tra R e Phyton

R e Phyton sono linguaggi di programmazione open source utilizzati in *Data Analysis* e *Machine Learning*. R è un linguaggio di programmazione orientato alla statistica, alla probabilità, alla simulazione e all'analisi statistica dei dati, mentre Python è un linguaggio di programmazione general-purpose.

Lo sviluppo di Python è iniziato nel 1989 al *National Research Institute for Mathematics and Computer Science (CWI)* di Amsterdam ad opera di *Guido Van Rossum*, che vi lavorava come ricercatore. Guido van Rossum, spesso

indicato come *Benevolent Dictator for Life (BDFL)*, è l'autore principale del linguaggio di programmazione Python. Il nome che Van Rossum attribuì al linguaggio è ispirato da una serie televisiva che la BBC trasmetteva agli inizi degli anni 70 "The Monty Python's Flying Circus ". Python è supportato dalla "Python Software Foundation" (https://www.python.org/psf/).

R può essere utilizzato per implementare concetti statistici come modellizzazione lineare e non lineare, analisi di serie temporali, regressione, clustering, metodi di stima e test statistici. Nell'implementazione delle varie tecniche statistiche e probabilistiche, il linguaggio R dedica particolare attenzione alla matematica alla base del modello. R è quindi maggiormente utilizzato nella Data Analysis e si basa su modelli statistici e analisi statistiche specializzate. Consente di creare grafici di alta qualità permettendo agli utenti di modificare l'estetica della grafica e personalizzarla. Invece, Python è un linguaggio di programmazione general-purpose molto utilizzato nell'ambito del Machine Learning ed è utile in una varietà di applicazioni. Può essere utilizzato per l'apprendimento automatico, lo sviluppo web, il calcolo scientifico, l'automazione, l'elaborazione del linguaggio naturale e molti altri ambiti. Python è più veloce nel gestire insiemi di dati di grandi dimensioni, rendendo tale linguaggio più appropriato nella gestione di Big Data.

In questo insegnamento si è scelto di utilizzare R, linguaggio sviluppato specificamente per l'analisi dei dati e il calcolo in ambito statistico.

1.3 Come interagire con R

L'attivazione di una sessione di lavoro viene effettuata agendo con un doppio click del mouse sull'icona che identifica l'applicazione R presente sul desktop. Dopo qualche istante sullo schermo compare la tipica console di R che presenta un'immagine simile a quella illustrata in figura:

```
R version 4.2.1 (2022-06-23)
Copyright (C) 2022 The R Foundation for Statistical Computing
R è un software libero ed è rilasciato SENZA ALCUNA GARANZIA.
Siamo ben lieti se potrai redistribuirlo, ma sotto certe condizioni.
Scrivi 'license()' o 'licence()' per dettagli su come distribuirlo.

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

Scrivi 'demo()' per una dimostrazione, 'help()' per la guida
oppure 'help.start()' per la guida nel browser HTML.
Scrivi 'q()' per uscire da R.
>
```

Nella console di R sono disponibili alcuni menù con un gruppo limitato di tasti di comando in grado di attivare velocemente le operazioni di

- Interrompi l'attuale computazione (Stop current computation)

- Apri script (Open Script)
- Carica area di lavoro (Load workspace)
- Salva area di lavoro (Save workspace)
- Copia (Copy)
- Incolla (Paste)
- Copia e Incolla (Copy and paste)
- Stampa (Print)

Tutti i comandi debbono essere inseriti dopo il segnale di prompt ">", che indica la disponibilità del sistema ad accettare nuovi comandi. Scopo della console è quello di fornire all'utente un ambiente dove scrivere i comandi che desidera eseguire. Dopo aver premuto il tasto Invio, il corrispondente comando scritto nella riga, viene immediatamente verificato per controllarne la correttezza ed eseguito se l'esito della verifica è positivo.

All'avvio R ricorda all'utente che digitando q() e premendo il tasto Invio si attiva la procedura di chiusura della sessione di lavoro; prima di concludere questa operazione, l'utente deve scegliere se desidera salvare un'immagine del workspace (spazio di lavoro), che contiene tutti i risultati delle elaborazioni, i dati, le variabili e molte altre informazioni. Se si salva il workspace questo verrà caricato in modo automatico la successiva volta che si utilizza R, in modo da riutilizzare le elaborazioni.

Per visualizzare tutti gli oggetti creati si può utilizzare il comando ls(). Inoltre, per cancellare un oggetto obj basta utilizzare il comando rm(obj). Per salvare alcuni elementi del workspace si può utilizzare l'istruzione:

$> \mathsf{save}(\mathsf{obj}_1, \ldots, \mathsf{obj}_n, \mathsf{file} = \text{"dati.rda"})$

che permette di salvare $\mathsf{obj}_1, \ldots, \mathsf{obj}_n$ nel file dati.rda. Per ricaricare quanto abbiamo salvato nel file dati.rda, si utilizza l'istruzione:

$> \mathsf{load}(\text{"dati.rda"})$

R usa la directory di lavoro corrente per salvare i dati in modo automatico. La funzione getwd() restituisce il percorso dell'attuale directory di lavoro. Se l'utente lavora a più progetti contemporaneamente e usa differenti directory per ogni progetto. È possibile spostarsi da una directory all'altra usando il comando $\mathsf{setwd}(\text{"C:/nomeDirectory"})$.

Il sistema mette a disposizione dell'utente una guida sul comando di cui si stanno richiedendo informazioni tramite il comando help(nomeComando) o equivalentemente ?nomeComando. Appare una nuova finestra che contiene una descrizione completa del comando con una serie di esempi. È possibile richiedere l'esecuzione di questi esempi scrivendo da console example(nomeComando). Altri tipici help sugli operatori e la sintassi sono: ?Syntax, ?Arithmetic, ?Logic, ?Comparison, ?Extract, ?Control.

1.4 Principali operatori e funzioni matematiche in R

I comandi elementari del linguaggio R sono costituiti da istruzioni di assegnazioni e da chiamate di funzioni. La forma di un'istruzione di assegnazione è

nomeVariabile <− espressione

e la sua esecuzione comporta la valutazione dell'espressione alla destra dell'operatore di assegnazione (costituito dalla coppia di caratteri < e −, scritti uno di seguito all'altro senza spazi intermedi di separazione) e la memorizzazione del risultato all'interno dell'oggetto il cui nome è indicato alla sinistra. Invece le chiamate di funzione hanno la forma del nome della funzione seguito immediatamente da una coppia di parentesi rotonde che racchiudono gli eventuali parametri; essi sono separati tra loro da virgole e sono passati come argomento alla funzione.

Gli *operatori* di R più frequentemente utilizzati sono elencati nella Tabella 1.1.

Tabella 1.1: Gli operatori di R più frequentemente usati

Descrizione	Simbolo di R	Esempio
Commento	#	> # Questo è un commento
Addizione	+	> 15 + 5.6 [1] 20.6
Sottrazione	−	> 15 − 5.6 [1] 9.4
Moltiplicazione	∗	> 15 ∗ 5.6 [1] 84
Esponenziazione	^	> 3^1.8 [1] 7.224674
x mod y	x%%y	> 19%%3 [1] 1
Divisione intera	%/%	> 19%/%3 [1] 6
Divisione reale	/	> 19/3 [1] 6.333333
Operatore di concatenazione	c()	> c(3, 2, 4) [1] 3 2 4
Sequenza da a a b di h	seq()	> seq(2, 12, 3) [1] 2 5 8 11
Operatore di sequenza	:	> 0 : 8 [1] 0 1 2 3 4 5 6 7 8

Gli *operatori relazionali* sono $<, <=, >, >=, ==, !=$. Gli operatori logici sono & per AND, | per OR e ! per la negazione.

I calcoli possono condurre a risposte che sono $+\infty$, rappresentato in R con Inf, oppure $-\infty$, rappresentato in R con −Inf. Altri calcoli possono condurre a quantità che non sono numeri. Questi sono rappresentati in R con Nan (not a

number). Quando, invece, si è in presenza di un dato mancante, R assegna il valore NA (not available).

In R è anche possibile eseguire operazioni che coinvolgono i numeri complessi. Per lavorare con i numeri complessi occorre esplicitare la parte complessa come mostra l'esempio seguente:

```
> (2-4i)*(2+4i)
[1] 20+0i
```

in cui i denota l'unità immaginaria. Se si preme Invio, R risponde visualizzando il risultato presente nella seconda riga, ossia 20. Il simbolo [1] indica che il risultato visualizzato è il primo elemento di un vettore.

ll linguaggio è sensibile all'uso dei caratteri maiuscoli e minuscoli; oltre ai caratteri alfabetici e numerici è consentito l'uso all'interno dei nomi, del carattere punto, ma non sono invece ammessi altri caratteri, quali lo spazio bianco o il trattino di underscore.

Alcuni simboli sono già definiti nel modulo base di R o nei pacchetti aggiuntivi ed è quindi preferibile che non siano utilizzati nuovamente l'utente. Ad esempio, i simboli q, T, F sono già usati da R. Altri nomi riservati sono:

```
FALSE Inf NA NaN NULL TRUE
break else for function if in next repeat while
diff mean pi range rank var
```

Per scoprire se un simbolo è già stato utilizzato basta digitare tale simbolo sul prompt e premere il tasto Invio. Se il simbolo non è stato ancora utilizzato apparirà un messaggio di errore che indica che l'oggetto non è stato trovato.

Il linguaggio R mette inoltre a disposizione un largo insieme di *funzioni matematiche* di cui le più usate sono riportate in Tabella 1.2. Le funzioni trigonometriche in R misurano gli angoli in radianti e la costante π è identificata con il nome pi.

Nei prossimi paragrafi considereremo le principali strutture dati (vettori, array, matrici, liste, data frame, fattori) su cui le funzioni di R lavorano.

1.5 Vettori

In R i vettori sono oggetti che mantengono al loro interno un insieme indicizzato di elementi tutti dello stesso tipo (numeri, stringhe, caratteri, valori logici, funzioni). Occorre sottolineare che in R uno scalare è rappresentato semplicemente mediante un vettore di lunghezza unitaria. Le più comuni classi di vettori in R sono:

- "*character*" un vettore di stringhe di caratteri di varie lunghezze;
- "*numeric*" un vettore di numeri reali;
- "*integer*" un vettore di numeri interi con segno;
- "*logical*" un vettore di valori logici (true o false);

Tabella 1.2: Alcune funzioni matematiche più frequentemente usate

Descrizione	Simbolo di R	Esempio
Radice quadrata di x	sqrt(x)	> sqrt(16) [1] 4
Valore assoluto di x	abs(x)	> abs(10 − 15 ∗ 2) [1] 20
Funzione esponenziale e^x	exp(x)	> exp(15 − 2 ∗ 7) [1] 2.718282
Logaritmo in base e di x	log(x)	> log(2.718282) [1] 1
Logaritmo in base 2 di x	log2(x)	> log2(8) [1] 3
Logaritmo in base 10 di x	log10(x)	> log10(1000) [1] 3
Logaritmo in base n di x	log(x, n)	> log(9, 3) [1] 2
$\lfloor x \rfloor$ (il più grande intero $\leq x$)	floor(x)	> floor(5.5) [1] 5
$\lceil x \rceil$ (il più piccolo intero $\geq x$)	ceiling(x)	> ceiling(5.5) [1] 6
Troncamento di x	trunc(x)	> trunc(5.5) [1] 5
Arrotondamento di x	round(x, digits = n)	> round(5.5, digits = 0) [1] 6
Seno di x in radianti	sin(x)	> sin(pi/4) [1] 0.7071068
Coseno di x in radianti	cos(x)	> cos(pi/4) [1] 0.7071068
Tangente di x in radianti	tan(x)	> tan(pi/4) [1] 1
Arcoseno di y in radianti	asin(y)	> asin(1) [1] 1.570796
Arcocoseno di y in radianti	acos(y)	> acos(−1) [1] 3.141593
Arcotangente di y in radianti	atan(y)	> atan(1) [1] 0.7853982
Fattoriale di x	factorial(x)	> factorial(5) [1] 120
Coefficiente binomiale $n!/(x!(n-x)!)$	choose(n, x)	> choose(5, 2) [1] 10

- "*complex*" un vettore di numeri complessi.

I valori possono essere inseriti in un vettore in differenti modi. Ad esempio, la prima riga successiva

```
> z<-8:12
> z # visualizza il vettore z
[1]  8  9 10 11 12
>
> class(z)
[1] "integer"
```

crea un vettore il cui nome è z costituito da valori interi da 8 fino a 12 utilizzando l'operatore : che crea una sequenza di interi successivi. Come mostrato nella seconda riga, se digitiamo z e premiamo Invio, R risponderà visualizzando gli elementi di z come mostrato nella terza riga. Il comando class(obj) individua la classe dell'oggetto obj.

Il modo più diretto per generare un vettore consiste nell'utilizzare la funzione di concatenazione c(), che può prendere un numero arbitrario di argomenti dello stesso tipo e il cui valore è un vettore ottenuto concatenando i suoi argomenti. Ad esempio, la prima riga successiva

```
> z<-c(8.5,9.4,10.3,11.5,12.6)
> z # visualizza il vettore z
[1]  8.5  9.4 10.3 11.5 12.6
>
> class(z)
[1] "numeric"
```

crea un vettore il cui nome è z costituito da valori reali utilizzando l'operatore di concatenazione. Ad esempio, nel vettore

```
> x<-c(-1,0,1)/0
> x # visualizza x
[1] -Inf  NaN  Inf
>
> class(x)
[1] "numeric"
```

si effettua la divisione per 0: fornisce $-$Inf se il numeratore è negativo, NAN (not a number) se il numeratore è zero e Inf se il numeratore è positivo. Il simbolo $[i]$ all'inizio di una riga, che per ora abbiamo visto come [1], fornisce in quale posizione nel vettore si trova il primo elemento visualizzato nella stessa riga.

È possibile anche inserire i numeri nel vettore uno alla volta usando la funzione scan(), Ad esempio, i primi sette righi successivi

```
> z<-scan()
1: 8
2: 9
3: 10
4: 11
5: 12
6:
Read 5 items
```

```
> z # visualizza il vettore z
[1]  8  9 10 11 12
```

creano un vettore il cui nome è z costituito da valori interi da 8 fino a 12 utilizzando la funzione scan(). Un'altra funzione che genera sequenze è seq(). Questa funzione può avere tre argomenti: il primo rappresenta l'inizio e il secondo la fine della sequenza, il terzo specifica l'ampiezza del passo. Ad esempio

```
> z<-seq(8,16,2)
> z # visualizza il vettore z
[1]  8 10 12 14 16
```

La funzione rep() è utile per replicare uno o più valori in un vettore. Ad esempio

```
> x<-rep(3,times=5)
> x # visualizza il vettore x
[1] 3 3 3 3 3
>
> y<-rep(c(4,3),times=5)
> y # visualizza il vettore y
 [1] 4 3 4 3 4 3 4 3 4 3
>
> z<-rep(c(4,3),times=5,each=2)
> z # visualizza il vettore z
 [1] 4 4 3 3 4 4 3 3 4 4 3 3 4 4 3 3 4 4 3 3
```

Uno dei più importanti attributi di un vettore è la sua lunghezza length(). Ad esempio, riferendoci all'esempio precedente:

```
> length(x)
[1] 5
>
> length(y)
[1] 10
>
> length(z)
[1] 20
```

In Tabella 1.3 sono elencate alcune funzioni che operano sui vettori.

L'ordinamento e la ricerca si possono effettuare facilmente in R tramite la funzione sort() e which().

Se si desidera *ordinare* gli elementi di un vettore z in ordine crescente basta utilizzare la funzione sort(z, decreasing = FALSE), mentre se si desiderano ordinare gli elementi di un vettore z in ordine decrescente basta utilizzare la funzione sort(z, decreasing = TRUE). Ad esempio,

```
> z<-c(8,15,16,2,4,8,2,3,10,9)
> sort(z, decreasing=FALSE)
 [1]  2  2  3  4  8  8  9 10 15 16
 >
> sort(z, decreasing=TRUE)
 [1] 16 15 10  9  8  8  4  3  2  2
```

Tabella 1.3: Alcune funzioni matematiche che operano sui vettori

Operazione	Significato
length(x)	numero di elementi del vettore x
max(x)	massimo valore contenuto nel vettore x
min(x)	minimo valore contenuto nel vettore x
sum(x)	somma di tutti gli elementi del vettore x
prod(x)	prodotto di tutti gli elementi del vettore x
diff(x)	vettore con le differenze esistenti tra i valori del vettore x
cumsum(x)	vettore con le somme parziali degli elementi del vettore x
cumprod(x)	vettore con i prodotti parziali degli elementi del vettore x
cummax(x)	vettore con i massimi parziali degli elementi del vettore x
cummin(x)	vettore con i minimi parziali degli elementi del vettore x
mean(x)	media aritmetica dei valori presenti nel vettore x: $\overline{x}$
median(x)	mediana dei valori presenti nel vettore x
range(x)	vettore contenente il minimo e il massimo del vettore x
unique(x)	vettore che contiene i valori distinti presenti nel vettore x
quantile(x)	vettore contenente il minimo, il quantile più basso, la mediana, il quantile più alto e il massimo del vettore x
var(x)	varianza campionaria dei valori del vettore x: s_x^2
sd(x)	deviazione standard campionaria dei valori del vettore x: s_x
cor(x, y)	correlazione campionaria tra i valori dei vettori x e y
sort(x)	vettore contenente gli elementi di x in ordine crescente
rev(x)	rovescia gli elementi del vettore x
rev(sort(x))	vettore contenente gli elementi di x in ordine decrescente

Per conoscere le posizioni assunte dagli elementi di un vettore che soddisfano una particolare condizione è possibile utilizzare la funzione which(), che richiede come argomento un vettore di tipo logico. Ad esempio,

```
> z<-c(8,9,10,11,12)
> which(z>10)
[1] 4 5
```

che fornisce la posizione degli elementi maggiori di 10 nel vettore.

I vettori presenti nella stessa espressione possono essere di lunghezza differente. In questo caso, il valore dell'espressione è un vettore con la stessa lunghezza del vettore più lungo presente in quell'espressione. I vettori con meno elementi sono riconsiderati tante volte fino ad arrivare alla stessa lunghezza del vettore più lungo. Ad esempio

```
> y<-c(1,2)
> z<-c(8,9,10,11,12)
> y*z
[1]  8 18 10 22 12
Warning message:
In y * z : longer object length is not a multiple of shorter object
     length
>
> y+z
[1]  9 11 11 13 13
Warning message:
In y + z : longer object length is not a multiple of shorter object
     length
```

I vettori di stringhe sono spesso utilizzati in R, ad esempio come etichette nei grafici. Le stringhe sono indicate da una sequenza di caratteri delimitati da una coppia di doppi apici (o una coppia di singoli apici). Ad esempio

```
> x<-c("Ascissa","Ordinata")
> x # visualizza il vettore x
[1] "Ascissa"  "Ordinata"
>
> class(x)
[1] "character"
```

Se si desidera accedere all'elemento i-esimo del vettore x occorre utilizzare x[i], mentre con x[−i] si crea un nuovo vettore contenente tutti gli elementi di x escluso l'elemento i-esimo. Invece, con x[i : j] si visualizzano gli elementi del vettore x dalla posizione i alla posizione j e infine con x[−(i : j)] si crea un nuovo vettore contenente tutti gli elementi di x esclusi gli elementi dalla posizione i alla posizione j

1.6 Array e Matrici

Un array è una collezione di elementi ognuno univocamente determinato da indici multipli. Gli elementi di un array sono tutti dello stesso tipo (numeri, stringhe, caratteri,...). In particolare, in un array tridimensionale a, a[i, j, k] è

l'elemento nella posizione (i, j, k) dell'array. In particolare, un array tridimensionale a di dimensione $n \times m \times r$ è visto come una sovrapposizione di r array bidimensionali di dimensione $n \times m$.

Un array tridimensionale a di dimensioni $n \times m \times r$ può essere ad esempio inizializzato nel seguente modo:

$> \mathsf{a} <- \mathsf{array}(1 : \mathsf{h}, \mathsf{dim} = \mathsf{c}(\mathsf{n}, \mathsf{m}, \mathsf{r}))$

che crea un array con i numeri da 1 fino a $h = n \times m \times r$.

Il comando dim(a) ritorna sotto forma di vettore le dimensioni dell'array mentre length(a) calcola il numero di elementi dell'array a considerato. Ad esempio, se si considera un array di dimensione $4 \times 3 \times 2$ si ha:

```
> a<-array(1:24,dim=c(4,3,2))
> a # visualizza l'array a
, , 1

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

, , 2

     [,1] [,2] [,3]
[1,]   13   17   21
[2,]   14   18   22
[3,]   15   19   23
[4,]   16   20   24
>
 dim(a)
[1] 4 3 2
>
> length(a)
[1] 24
```

Gli array sono utili quando è necessario fornire differenti informazioni per identificare un elemento. Un esempio tipico di array contiene le temperature minime e massime, il giorno, il mese e l'anno di registrazione in diverse città (con tutti i valori espressi da numeri).

Una matrice è un array bidimensionale di elementi univocamente determinati da una coppia di numeri interi, che costituiscono l'indice di riga e di colonna. Un array bidimensionale può essere inizializzato nel seguente modo:

$> \mathsf{a} <- \mathsf{array}(1 : \mathsf{h}, \mathsf{dim} = \mathsf{c}(\mathsf{n}, \mathsf{m}))$

che crea un array $n \times m$ con i numeri da 1 a $h = n \times m$. L'array bidimensionale può anche essere creato nel seguente modo:

$> \mathsf{a} <- \mathsf{matrix}(1 : \mathsf{h}, \mathsf{nrow} = \mathsf{n}, \mathsf{ncol} = \mathsf{m})$

dove n è il numero di righe della matrice e m il numero di colonne con $h = n \times m$. Ad esempio, per creare un vettore 3×4 usando l'array bidimensionale si ha:

```
> a<-array(1:12,dim=c(3,4))
> a # visualizza l'array bidimensionale a
```

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

mentre usando la matrice si ha:

```
> a<-matrix(1:12,nrow=3,ncol=4)
> a # visualizza la matrice a
     [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12
```

Un altro modo per creare la precedente matrice è quello di utilizzare l'operatore di concatenazione, ossia:

```
> a<-matrix(c(1,2,3,4,5,6,7,8,9,10,11,12),nrow=3,ncol=4)
> a # visualizza la matrice a
     [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12
```

Come si può osservare dai due esempi precedenti, R utilizzando i comandi array() e matrix() riempie le matrici per colonna. Se si desidera forzare R a riempire la matrice per riga si deve specificare l'opzione byrow = TRUE come ultimo argomento di matrix().

Se si vuole costruire una matrice $n \times m$ con tutti elementi nulli si può utilizzare il comando matrix() nel seguente modo

$>$ a $<-$ matrix(0, nrow = n, ncol = m)

mentre se si desidera riempire la matrice con elementi unitari si ha:

$>$ a $<-$ matrix(1, nrow = n, ncol = m)

Se si desidera creare una matrice $n \times m$ senza assegnare valori precisi agli elementi si può scrivere

$>$ a $<-$ matrix(, nrow = n, ncol = m)

Ad esempio, per una matrice 2×3 si ha:

```
>  a<-matrix(,nrow=2,ncol=3)
> a # visualizza la matrice a
     [,1] [,2] [,3]
[1,]   NA   NA   NA
[2,]   NA   NA   NA
```

dove NA è utilizzato in R per definire i valori non disponibili.

Se si desidera accedere all'elemento (i, j) della matrice a occorre utilizzare a[i, j], mentre con a[, j] si selezionano gli elementi della colonna j-esima e infine con a[i,] si selezionano gli elementi della riga i-esima.

→ Composizione di vettori e matrici

Le funzioni cbind() e rbind() permettono di creare opportune matrici componendo vettori di uguale lunghezza e matrici delle stesse dimensioni. La prima

funzione usa i vettori per creare le colonne mentre la seconda per creare le righe. Ad esempio, usando cbind() si ha

```
> a<-cbind(c(1,2,3),4:6,matrix(7:12,nrow=3,ncol=2))
> a # visualizza la matrice a
     [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12
```

mentre usando rbind() risulta:

```
> b<-rbind(c(1,2,3),4:6,matrix(7:12,nrow=2,ncol=3,byrow=TRUE))
> b # visualizza la matrice b
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
[4,]   10   11   12
```

Altre funzioni molto utilizzate nell'algebra matriciale sono:

diag(v), con v vettore, usata per creare una matrice diagonale con gli elementi del vettore sulla diagonale e i restanti elementi nulli;

diag(a), con a matrice, fornisce un vettore costituito dagli elementi della diagonale principale della matrice;

diag(n), con n intero, fornisce la matrice identità $n \times n$.

È possibile cambiare i nomi assegnati alle intestazioni delle righe e delle colonne di una matrice a utilizzando i comandi rownames(a) e colnames(a) come mostrato nel seguente esempio:

```
> a<-matrix(1:6,nrow=2,ncol=3)
> rownames(a)<-c("X1","X2")
> colnames(a)<-c("Y1","Y2","Y3")
> a # visualizza la matrice a
   Y1 Y2 Y3
X1  1  3  5
X2  2  4  6
```

→ Operazioni matriciali

R permette di eseguire le usuali operazioni sui vettori e le matrici. R interpreta i suoi vettori come vettori colonna. Per effettuare la *trasposizione di un vettore o di una matrice* occorre utilizzare l'operatore t() di trasposizione. Nell'esempio seguente, creiamo una matrice 2×2 e calcoliamo la sua trasposta

```
> a<-matrix(c(-1,-2,1,2),nrow=2,ncol=2)
> a # visualizza la matrice a
     [,1] [,2]
[1,]   -1    1
[2,]   -2    2
> t(a) # visualizza la matrice trasposta
     [,1] [,2]
[1,]   -1   -2
[2,]    1    2
```

Sui vettori e sulle matrici delle stesse dimensioni sono applicabili le usuali operazioni aritmetiche $+, -, *, :$ che agiscono elemento per elemento. Occorre osservare che in tal caso il prodotto di matrici utilizzando l'operatore $*$ non corrisponde al prodotto matriciale.

R fornisce la possibilità di effettuare il *prodotto matriciale* tra due matrici a di dimensione $n \times m$ e b di dimensione $m \times r$ utilizzando l'operatore $\%*\%$. L'esempio seguente considera il prodotto di una matrice a di dimensione 2×3 e di una matrice b di dimensione 3×4 ottenendo una matrice di dimensioni 2×4:

```
> a<-matrix(c(1,2,3,4,5,6),nrow=2,ncol=3)
> a # visualizza la matrice a
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
> b<-matrix(1:12,nrow=3,ncol=4)
> b # visualizza la matrice b
     [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12
>
> c<-a%*%b
> c # visualizza la matrice c
     [,1] [,2] [,3] [,4]
[1,]   22   49   76  103
[2,]   28   64  100  136
```

È possibile calcolare il *determinante* di una matrice quadrata a con il comando det(a), mentre per calcolare l'*inversa* di una matrice a quadrata non singolare (ossia con determinante diverso da zero) si usa il comando solve(a). Un esempio è il seguente:

```
> a <- matrix(c(1,2,1,1,1,3,1,1,2),nrow=3,ncol=3)
> a # visualizza la matrice a
     [,1] [,2] [,3]
[1,]    1    1    1
[2,]    2    1    1
[3,]    1    3    2
>
> det(a) # calcola il determinante della matrice a
[1] 1
>
> b<-solve(a) # calcola la matrice inversa della matrice a
> b # visualizza la matrice inversa b
     [,1] [,2] [,3]
[1,]   -1    1    0
[2,]   -3    1    1
[3,]    5   -2   -1
>
> c<-a%*%b # calcola il prodotto delle due matrici
> round(c,digits= 2) # visualizza la matrice identita'
     [,1] [,2] [,3]
[1,]    1    0    0
[2,]    0    1    0
[3,]    0    0    1
```

→ Autovalori e autovettori di una matrice
R consente anche di determinare gli *autovalori* e gli *autovettori* di una matrice quadrata A di ordine n. Occorre determinare una costante λ tale che $A\,x = \lambda\,x$, dove x individua un vettore colonna di lunghezza n. Ciò equivale a richiedere che $(A-\lambda\,I)x = 0$, dove I è la matrice identità. La soluzione non banale del problema corrisponde a richiedere che $\det(A - \lambda\,I) = 0$, il che corrisponde a risolvere un'equazione algebrica di grado n in λ, detta equazione caratteristica. Ognuna delle radici $\lambda_1, \lambda_2, \ldots, \lambda_n$ è chiamata autovalore e il corrispondente vettore soluzione x_i $(i = 1, 2, \ldots, n)$ è chiamato autovettore corrispondente all'autovalore λ_i della matrice A.

In R la funzione eigen(a) calcola gli autovalori e gli autovettori di una matrice a. Il risultato è una lista di due componenti: un vettore di nome values contenente gli autovalori e una matrice di nome vectors che contiene i corrispondenti autovettori sulle sue colonne. Questi due componenti possono essere usati in istruzioni di assegnazione utilizzando eigen(a)$values e eigen(a)$vectors, rispettivamente. Ad esempio,

```
> a<-matrix(1:4,nrow=2,ncol=2)
> a # visualizza la matrice a
     [,1] [,2]
[1,]    1    3
[2,]    2    4
> eigen(a)
$values
[1]  5.3722813 -0.3722813

$vectors
           [,1]       [,2]
[1,] -0.5657675 -0.9093767
[2,] -0.8245648  0.4159736
```

Per selezionare l'autovalore j-esimo di una matrice quadrata a basta utilizzare il comando eigen(a)$values[j], mentre digitando eigen(a)$vectors[, j] si ottiene il corrispondente autovettore.

Nell'esempio precedente, $\det(A-\lambda\,I) = (1-\lambda)(4-\lambda) - 6 = \lambda^2 - 5\lambda - 2 = 0$ conduce agli autovalori $\lambda_1 = (5 + \sqrt{33})/2 \simeq 5.37$ e $\lambda_2 = (5 - \sqrt{33})/2 \simeq -0.37$.

1.7 Liste

A differenza dei vettori, degli array e delle matrici, in R una lista è un oggetto che consiste in una raccolta di altri oggetti, che possono essere anche differenti tra loro, detti componenti della lista. Ad esempio, i risultati creati dalla funzione eigen() si presentano come una lista costituita da un oggetto (primo componente) di tipo vettore, che contiene gli autovalori, e un oggetto (secondo componente) di tipo matrice, che contiene i corrispondenti autovettori.

In generale, una lista può essere creata con il comando

$>$ nomeLista $<-$ list($\mathsf{name}_1 = \mathsf{obj}_1, \mathsf{name}_2 = \mathsf{obj}_2, \ldots, \mathsf{name}_n = \mathsf{obj}_n$)

Ai componenti di una lista è sempre associato un indice che ne individua la posizione nella lista e si può anche associare un nome che contraddistingue

gli elementi. I componenti della lista possono infatti essere individuati in due modi diversi: tramite la loro posizione nella lista oppure utilizzando i loro nomi (se disponibili). Se i nomi sono omessi, i componenti della lista possono essere riferiti solo tramite l'indice che individua la loro posizione. La funzione length(nomeLista) fornisce la dimensione della lista, ossia il numero di componenti che costituiscono la lista considerata.

Ci si può riferire al componente j-esimo della lista mediante nomeLista[j] (singole parentesi quadre) o nomeLista[[j]] (doppie parentesi quadre), ed anche tramite il loro nome (se presente) con il comando nomeLista$name$_j$. Se si utilizzano singole parentesi quadre il componente estratto è ancora una lista e si conservano i nomi degli oggetti componenti la lista, mentre se si usano le parentesi quadre doppie si accede al componente della lista senza includere l'eventuale nome. L'esempio seguente contiene una lista con tre componenti: un vettore di caratteri, un vettore numerico e una matrice.

```
> lst<-list(c("X1","X2"),c(0,0),matrix(1:6,nrow=2,ncol=3))
> lst #visualizza la lista lst
[[1]]
[1] "X1" "X2"

[[2]]
[1] 0 0

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

>
> length(lst) # visualizza la lunghezza della lista lst
[1] 3
>
> lst[[3]] [,2] # visualizza la seconda colonna della matrice
[1] 3 4
```

Riconsideriamo il precedente esempio inserendo ora i nomi ai componenti della lista.

```
> lst<-list(n1=c("X1","X2"),n2=c(0,0),n3=matrix(1:6,nrow=2,ncol=3))
> lst #visualizza la lista lst
$n1
[1] "X1" "X2"

$n2
[1] 0 0

$n3
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
>
> lst[2] # visualizza il secondo componente della lista (con
    singole parentesi)
$n2
```

```
[1] 0 0
>
> lst[[2]] # visualizza il secondo componente della lista (con
    doppie parentesi)
[1] 0 0
>
> lst$n2 # visualizza il secondo componente della lista (usando il
    nome n2)
[1] 0 0
>
> lst$n3[,2] # visualizza la seconda colonna della matrice
[1] 3 4
```

L'esempio precedente mostra che se si assegnano i nomi ai singoli componenti della lista è possibile estrarre ogni elemento della lista invocando direttamente il loro nome attraverso il simbolo $ oppure utilizzando le parentesi quadre. Si nota che le singole parentesi quadre creano ancora una lista e si preserva il nome dell'oggetto estratto, mentre con le doppie parentesi non si preserva il nome dell'oggetto estratto.

Dai due esempi precedenti si nota che l'oggetto creato con list() mantiene al suo interno altri oggetti, come vettori, matrici, caratteri, liste o altre strutture.

La funzione is.list(name) verifica se l'argomento name è una lista restituendo TRUE o FALSE; tale funzione può essere utilizzata anche per i vettori, le matrici e gli array. Riferendoci all'esempio precedente si ha:

```
> lst<-list(n1=c("X1","X2"),n2=c(0,0),n3=matrix(1:6,nrow=2,ncol=3))
>
> is.list(lst[[2]])
[1] FALSE
> is.list(lst[2])
[1] TRUE
> is.list(lst$n2)
[1] FALSE
>
> is.list(lst[[3]])
[1] FALSE
> is.list(lst[3])
[1] TRUE
> is.list(lst$n3)
[1] FALSE
```

che mostra che *l'uso delle singole parentesi quadre crea ancora una lista.*

La funzione str() restituisce informazioni sulla struttura di un oggetto. Riferendoci all'esempio precedente si ha:

```
> lst<-list(n1=c("X1","X2"),n2=c(0,0),n3=matrix(1:6,nrow=2,ncol=3))
> str(lst)
List of 3
 $ n1: chr [1:2] "X1" "X2"
 $ n2: num [1:2] 0 0
 $ n3: int [1:2, 1:3] 1 2 3 4 5 6
```

È possibile aggiungere e cancellare componenti alla lista. Per aggiungere un componente alla fine di una lista di $n-1$ componenti basta utilizzare l'istruzione di assegnazione

$> \mathsf{nomeLista[n]} <- \mathsf{list(name_n = obj_n)}$

Per aggiungere un componente in una posizione j di una lista con n componenti, basta utilizzare la funzione di concatenazione nell'istruzione di assegnazione

$> \mathsf{nomeLista} <- \mathsf{c(nomeLista[1, j-1], name_j = obj_j, nomeLista[j, n])}$

che crea una lista con $n+1$ componenti. I componenti in posizione successiva a quello inserito vengono automaticamente spostati di una posizione. Per cancellare il componente j-esimo di una lista basta invece utilizzare l'istruzione di assegnazione:

$> \mathsf{nomeLista[j]} <- \mathsf{NULL}$

```
> lst1<-list(c1=c(0,1),c2=c("si","no"))
>
> lst1$new<-"NEW"
> lst1 # visualizza la lista con un elemento aggiunto alla fine
$c1
[1] 0 1
$c2
[1] "si" "no"
$new
[1] "NEW"
>
lst1[2]<- NULL # il secondo elemento della lista e' eliminato
> lst1 # visualizza la nuova lista ottenuta
$c1
[1] 0 1
$new
[1] "NEW"
>
> lst1<-list(sinistra=c(0,1)) #  lista di sinistra
> lst2<-list(destra=c("A","B")) # lista di destra
> lst3<-c(lst1,dentro="NEW",lst2)
> lst3
$sinistra
[1] 0 1
$dentro
[1] "NEW"
$destra
[1] "A" "B"
```

1.8 Fattori

I dati di tipo statistico si possono suddividere in due gruppi principali: *quantitativi* e *qualitativi*. I dati quantitativi sono rappresentati tramite valori numerici, mentre i dati qualitativi sono rappresentati attraverso stringhe di caratteri oppure mediante opportune classi (sottointervalli numerici). In R si usa il termine *fattore* per indicare variabili di tipo vettore in grado di rappresentare informazioni di tipo qualitativo. Ad esempio, per una variabile individuo i valori possono essere indicati mediante i nomi "bambino", "giovane", "adulto", "anziano" oppure utilizzando i rispettivi codici numerici $1, 2, 3, 4$. Nelle variabili qualitative

può anche essere presente un ordinamento tra i valori. Riferendosi all'esempio precedente, i valori della variabile individuo possono essere così ordinati: "bambino" < "giovane" < "adulto" < "anziano". Per la creazione di fattori possono essere usate anche le variabili numeriche nel caso in cui si desidera considerare dei raggruppamenti o degli intervalli di valori. Ad esempio, l'età degli individui può essere suddivisa nelle seguenti classi: $(0, 12]$, $(12, 18]$, $(18, 60]$, $(60, 100]$. In R per creare dati di tipo qualitativo è conveniente utilizzare la funzione factor(). Nell'esempio seguente

```
> individuo<-c("bambino","giovane","adulto","anziano","bambino","
    giovane")
> individuo #visualizza il vettore individuo
[1] "bambino" "giovane" "adulto"  "anziano" "bambino" "giovane"
>
> persona<-factor(individuo)
> persona # visualizza il fattore persona
[1] bambino giovane adulto  anziano bambino giovane
Levels: adulto anziano bambino giovane
>
> str(individuo)
 chr [1:6] "bambino" "giovane" "adulto" "anziano" "bambino" "
     giovane"
> str (persona)
 Factor w/ 4 levels "adulto","anziano",..: 3 4 1 2 3 4
```

si crea un vettore individuo contenente stringhe di caratteri; ciascuna stringa è racchiusa tra coppie di doppi apici. La funzione factor(individuo) trasforma il vettore individuo nel fattore persona. Si nota che i valori della variabile persona si presentano senza doppi apici poiché sono stati trasformati da stringhe di caratteri in nuovi valori memorizzati sotto forma di codici numerici. L'attributo Levels mantiene i valori assunti utilizzando 4 livelli (corrispondenti agli indici 1 2 3 4) ordinati alfabeticamente.

Se si desidera imporre un particolare ordinamento si può usare la funzione ordered(). Relativamente all'esempio precedente, si può procedere senza utilizzare la funzione factor() nel seguente modo:

```
> individuo<-c("bambino","giovane","adulto","anziano","bambino","
    giovane")
> persona1<-ordered(individuo,levels=c("bambino","giovane","adulto"
    ,"anziano"))
> persona1 # visualizza il fattore persona1
[1] bambino giovane adulto  anziano bambino giovane
Levels: bambino < giovane < adulto < anziano
>
> str(persona1)
 Ord.factor w/ 4 levels "bambino"<"giovane"<..: 1 2 3 4 1 2
```

La funzione ordered() quindi crea una variabile di tipo fattore che consente un ordinamento interno dei valori permettendo così l'uso degli operatori relazionali. Se invece, si desidera modificare l'ordinamento alfabetico ottenuto con la funzione factor(), si può specificare l'ordinamento nel modo seguente:

```
> individuo<-c("bambino","giovane","adulto","anziano","bambino","
    giovane")
> persona<-factor(individuo)
> ordered(persona,levels=c("bambino","giovane","adulto","anziano"))
[1] bambino giovane adulto  anziano bambino giovane
Levels: bambino < giovane < adulto < anziano
```

È possibile anche stabilire l'ordinamento direttamente nella funzione factor() nel seguente modo:

```
> individuo<-c("bambino","giovane","adulto","anziano","bambino","
    giovane")
> persona1<-factor(individuo,levels=c("bambino","giovane","adulto",
    "anziano"),ordered=TRUE)
> persona1 # visualizza il fattore ordinato persona1
[1] bambino giovane adulto  anziano bambino giovane
Levels: bambino < giovane < adulto < anziano
```

In R si utilizza la funzione cut() per trasformare dati numerici in dati qualitativi mediante la loro collocazione in opportune classi sulla base di quanto indicato nel parametro breaks. Nel parametro breaks si inseriscono gli estremi degli intervalli che sono aperti a sinistra e chiusi a destra. Se si desidera ottenere intervalli chiusi a sinistra e aperti a destra occorre specificare in cut() l'opzione aggiuntiva right = FALSE. Nell'esempio seguente si considera un vettore numerico contenente i voti di 8 studenti e si considerano le seguenti classi: $(17, 21]$, $(21, 24]$, $(24, 27]$, $(27, 31]$.

```
> votiStudenti<-c(18,21,25,28,30,25,24,25)
> votiStudenti # visualiza il vettore votiStudenti
[1] 18 21 25 28 30 25 24 25
>
> voti<-cut(votiStudenti,breaks=c(17,21,24,27,31))
> voti # visualiza il vettore voti
[1] (17,21] (17,21] (24,27] (27,31] (27,31] (24,27] (21,24] (24,27]
Levels: (17,21] (21,24] (24,27] (27,31]
```

È inoltre possibile anche aggiungere delle etichette alle classi sulla base di quanto specificato nel parametro label. L'ordine con cui questi nomi sono indicati è importante poiché corrisponde all'ordine di associazione dei valori. Ponendo label = FALSE vengono utilizzati gli interi come etichette. Riferendosi all'esempio precedente si ha:

```
> votiStudenti<-c(18,21,25,28,30,25,24,25)
> giudizio<-cut(votiStudenti,breaks=c(17,21,24,27,31),
+ label=c("sufficiente","buono","distinto","ottimo"))
> giudizio # visualizza il fattore giudizio
[1] sufficiente sufficiente distinto    ottimo      ottimo
    distinto    buono       distinto
Levels: sufficiente buono distinto ottimo
>
> str(giudizio)
 Factor w/ 4 levels "sufficiente",..: 1 1 3 4 4 3 2 3
```

L'esempio precedentemente mostra che alla classe $(17, 21]$ è associato sufficiente, alla classe $(21, 24]$ è associato buono, alla classe $(24, 27]$ è associato distinto e alla classe $(27, 31]$ è associato ottimo.

Le funzioni as.integer() e as.numeric() aventi come argomento un fattore effettuano la trasformazione di un fattore in un vettore di interi o numerico. Tali funzioni ritornano un vettore numerico che associa ad ogni elemento del fattore un codice indicante la sua posizione all'interno di levels. Ciò è mostrato nell'esempio seguente:

```
> as.integer(giudizio)
[1] 1 1 3 4 4 3 2 3
> as.numeric(giudizio)
[1] 1 1 3 4 4 3 2 3
```

1.9 Data frame

In R un data frame è un oggetto di tipo lista che si presenta in forma di tabella (*matrice di dati*) ed è costituito da righe e colonne; ogni riga del data frame individua un'*osservazione* e ad ogni colonna corrisponde una *variabile*. Come per le matrici, le colonne di un data frame hanno tutte la stessa lunghezza e i valori contenuti in ogni singola colonna sono omogenei (numeri, caratteri, valori logici, ...). Invece, a differenza delle matrici, un data frame può avere colonne di tipo diverso. Le righe e le colonne possono avere delle *etichette* costituite da nomi o da valori numerici. Il numero di colonne è individuato da length(nomeDataFrame), mentre dim(nomeDataFrame) fornisce un vettore contenente il numero di righe e di colonne del data frame. Il comando names(nomeDataFrame) restituisce i nomi delle etichette delle colonne. I nomi delle righe possono essere modificati attraverso il comando row.names(nomeDataFrame). Nell'esempio seguente creiamo un data frame costituito da due variabili (la prima di tipo numerico e la seconda costituita da stringhe di caratteri).

```
> df<-data.frame(Y1=c(100,200,400),Y2=c("si","no","si"))
> df # visualizza il data frame df
   Y1 Y2
1 100 si
2 200 no
3 400 si
>
> length(df) # visualizza il numero di variabili
[1] 2
> dim(df) # visualizza il numero di osservazioni e variabili (righe
     e colonne)
[1] 3 2
>
> row.names(df)=c("X1","X2","X3") # cambia le etichette delle righe
> df # visualizza il data frame df
    Y1 Y2
X1 100 si
X2 200 no
X3 400 si
>
```

```
> df$Y1
[1] 100 200 400
> df$Y2
[1] "si" "no" "si"
```

Il comando rm(nomeDataFrame) permette di eliminare l'intero data frame, ma non consente di eliminare le variabili interne. Se si vuole eliminare la variabile j-esima di un data frame basta utilizzare il comando

> nomeDataFrame[j] <− NULL

In Tabella 1.4 sono elencate alcune funzioni utilizzabili per esaminare un data frame.

Tabella 1.4: Alcune funzioni che operano sui data frame

Operazione	Significato
length(df)	numero di colonne del data frame df
dim(df)	numero di righe e colonne (osservazioni e variabili) del data frame df
names(df)	nomi delle etichette delle colonne del data frame df
row.names(df)	assegna/modifica le etichette delle righe del data frame df
df[i, j]	individua l'elemento in posizione (i, j) nel data frame df
df[i : j, k]	seleziona nella k-esima colonna gli elementi da i a j
df[i : j, k : r]	seleziona una parte del data frame df avente le righe da i a j e le colonne da k a r
df[i,]	seleziona la parte del data frame df costituita dalla i-esima riga
df[, j]	seleziona la parte del data frame df costituita dalla j-esima colonna

Il comando class(obj) individua la classe dell'oggetto obj. Ad esempio,

```
> df<-data.frame(Y1=c(100,200,400),Y2=c("si","no","si"))
>
> class(df)
[1] "data.frame"
>
> class(df[1,1])
[1] "numeric"
>
> class(df[1,2])
[1] "factor"
>
> class(df[1:2,1:2])
[1] "data.frame"
```

Riferendosi all'esempio precedente, il comando class(df) restituisce "data.frame", il comando class(df[1, 1]) restituisce "numeric", il comando class(df[1, 2]) restituisce "factor", il comando class(df[1 : 2, 1 : 2]) restituisce "data.frame".

In R esistono molti insiemi di dati contenuti nel package di base e altri sono disponibili in altri package. Per vedere l'elenco degli insiemi di dati disponibili nel package di base basta utilizzare il comando data(); per caricare un particolare insieme di dati, ad esempio PlantGrowth, basta utilizzare il comando:

> data(PlantGrowth)

Per vedere l'elenco degli insiemi di dati contenuti in un altro package di nome pkg si può utilizzare il comando data(package = "pkg"), mentre per caricare un file di nome growth presente in tale package si può usare il comando

> data(growth, package = "pkg")

Se invece il package di nome pkg era già stato caricato tramite il comando library(pkg), allora data() serve per visualizzare gli insiemi di dati disponibili e data(growth) serve per caricare l'insieme di dati growth contenuto nel package pkg.

1.10 Definizione di nuove funzioni

Il sistema R è dotato di un linguaggio di programmazione evoluto che consente la progettazione e la realizzazione di nuove funzioni definite dall'utente.

La definizione di una funzione in R deve rispettare la seguente sintassi:

> nomeFunzione <− function(arg1, arg2, ...) corpo della funzione

Il nome della funzione è indicato alla sinistra dell'operatore di assegnazione. Gli argomenti, se presenti, sono inseriti separati da virgole nella coppia di parentesi tonde e sono usati per passare i valori iniziali alla funzione. Alla fine si definisce il corpo della funzione, ossia le istruzioni necessarie per ottenere il risultato atteso. Se il corpo è formato da più di un'istruzione occorre racchiuderlo tra una coppia di parentesi graffe. Una funzione ritorna il valore calcolato nell'ultima espressione oppure, se è presente l'istruzione return(value) ritorna i valori contenuti nella variabile value indicata nel suo argomento. Occorre osservare che il risultato ottenuto da una funzione può essere costituito da un unico valore oppure da una struttura che racchiude al suo interno più valori. Una chiamata di funzione è fatta con il comando > nomeFunzione(arg1, arg2, ...). Un esempio di funzione che calcola la misura della circonferenza è

```
> circonf<-function(r)
+ 2*pi*r
>
> circonf(4) # esecuzione della funzione per r = 4
[1] 25.13274
```

mentre un esempio di funzione che calcola il discriminante e le radici reali di un'equazione di secondo grado $a\,x^2 + b\,x + c = 0$ $(a \neq 0)$ è:

```
> radici<-function(a,b,c){
+ delta<-b^2-4*a*c
+ if(delta>=0){
+ x1<-(-b-sqrt(delta))/(2*a)
+ x2<-(-b+sqrt(delta))/(2*a)
+ return(c(delta,x1,x2))
+ }
+ else
+ "Non esistono radici reali" }
>
> radici(1,0,-1) # esecuzione della funzione per a=1, b=0, c=-1
[1]  4 -1  1
```

```
>
> radici(1,2,3) # esecuzione della funzione per a=1, b=2, c=3
[1] "Non esistono radici reali"
```

Si nota che nel primo esempio il risultato della funzione è un unico valore (misura della circonferenza), mentre nel secondo esempio il risultato è un vettore con tre componenti (discriminante e le due radici reali).

Spesso in R si utilizza la funzione sapply(v, funz) che permette di applicare la funzione indicata nel parametro funz a tutti gli elementi di una sequenza o di un vettore v restituendo un vettore di valori. Ad esempio,

```
> r<-c(1,2,3,4,5)
> circonf<-function(r) 2*pi*r
>
> sapply(r,circonf)
[1]  6.283185 12.566371 18.849556 25.132741 31.415927
```

restituisce il valore della misura della circonferenza per tutti gli elementi del vettore r.

Inoltre in R si utilizza anche la funzione apply(df, num, funz) per applicare la funzione alle righe (num = 1) oppure alle colonne (num = 2) di una matrice o di un data frame restituendo un vettore di valori. Applichiamo la funzione apply() ad una matrice calcolando il valore minimo delle righe ed il valore massimo delle colonne.

```
> a <- matrix (c(5, 2, 7, 1, 2, 8, 4, 5, 6), nrow=3, ncol=3)
> rownames(a)<-c("X1","X2","X3")
> colnames(a)<-c("Y1","Y2","Y3")
> a # visualizza la matrice a
   Y1 Y2 Y3
X1  5  1  4
X2  2  2  5
X3  7  8  6
>
> apply(a,1,min)
X1 X2 X3
 1  2  6
 >
> apply(a,2,max)
Y1 Y2 Y3
 7  8  6
```

Calcoliamo ora la somma e la media aritmetica degli elementi delle prime due colonne del seguente data frame:

```
> df<-data.frame(Y1=c(5.1,4.9,5.0),Y2=c(0.2,0.3,0.5),Y3=c("si","no"
    ,"no"))
> row.names(df)<-c("X1","X2","X3")
> df # visualizza il data frame
    Y1  Y2 Y3
X1 5.1 0.2 si
X2 4.9 0.3 no
X3 5.0 0.5 no
>
> apply(df[1:2],2,sum)
```

```
Y1 Y2
15  1
>
> apply(df[1:2],2,mean)
       Y1        Y2
5.0000000 0.3333333
```

Il primo argomento della funzione apply() seleziona le prime due colonne del data frame df, il secondo indice indica che la funzione deve essere applicata alle colonne selezionate della matrice (2), il terzo argomento è la funzione da utilizzare. Se, dopo aver selezionato le prime due colonne, si desidera invece applicare la funzione agli elementi delle righe basta utilizzare 1 come secondo argomento, come mostrato nel seguito:

```
> apply(df[1:2],1,sum)
 X1  X2  X3
5.3 5.2 5.5
>
> apply(df[1:2],1,mean)
  X1   X2   X3
2.65 2.60 2.75
```

1.11 Espressioni condizionali

Nelle espressioni condizionali si può far uso di diversi operatori relazionali $<$, $>$, $<=$, $>=$, $==$, $!=$ e logici NOT (!), AND (& e &&) e OR (| e ||). Se si utilizzano gli operatori logici & e | il controllo è eseguito termine a termine tra gli elementi di due vettori e si ottiene un corrispondente vettore contenente i risultati logici via via ottenuti dal confronto degli elementi. Invece, gli operatori logici && e || eseguono il controllo in sequenza da sinistra a destra esaminando così soltanto il primo elemento di ogni vettore. Il seguente esempio

```
> v<-c(-3,-2,-1,0,1,2,3)
>
> (v<=1)&(v>=-1)
[1] FALSE FALSE  TRUE  TRUE  TRUE FALSE FALSE
>
> (v<=1)&&(v>=-1)
[1] FALSE
```

mostra che l'operatore & è applicato a tutti i singoli elementi del vettore mentre l'operatore && esegue il controllo soltanto sul primo elemento.

1.12 Strutture di selezione

Le strutture di selezione messe a disposizione dal linguaggio R sono if e if . . . else. La sintassi usata in R per la prima struttura di selezione è:

> if(espressione condizionale){
>
> blocco di istruzioni da eseguire se l'espressione è vera}

mentre per la seconda struttura di selezione è

> if (espressione condizionale){

blocco di istruzioni da eseguire se l'espressione condizionale è vera}

else{

blocco di istruzioni da eseguire se l'espressione condizionale è falsa}

La parte di codice da eseguire dopo if e else necessita di coppie di parentesi graffe nel caso siano presenti più istruzioni. Le strutture di selezione possono avere altre strutture di selezione if e if . . . else annidate all'interno.

Esiste una versione vettorizzata di if . . . else che ha la forma:

> ifelse(espressione condizionale, a, b)

che è capace di operare su tutti i valori presenti in un vettore e non solo su di un singolo valore. Per chiarire il funzionamento del costrutto ifelse() consideriamo il seguente esempio:

```
> v<-c(-3,-2,-1,0,1,2,3)
> a<-ifelse( (v<=1)&(v>=-1),1,0)
> a # visualizza a
[1] 0 0 1 1 1 0 0
```

che mostra che con ifelse() viene valutata l'espressione condizionale per ogni elemento del vettore e si ottiene alla fine un vettore di 0 e 1 che può essere assegnato al vettore a.

1.13 Strutture iterative

Le strutture usate in R per realizzare costrutti iterativi sono simili a quelle disponibili in altri linguaggi di programmazione, ossia for, while e repeat.

La sintassi della struttura for è la seguente:

> for(variabile che assume valori in un insieme) {

corpo del ciclo }

Il corpo del ciclo può essere costituito da un'unica istruzione oppure da più istruzioni racchiuse tra una coppia di parentesi graffe. La variabile indicata può assumere tutti i valori nell'insieme che può essere costituito da una sequenza oppure dagli elementi di un vettore. Alcuni esempi di applicazione della struttura for sono riportati nel seguito.

```
> for(n in 1:4) print(n^2)
[1] 1
[1] 4
[1] 9
[1] 16
>
> for(n in seq(2,-4,by=-2)) print(n)
[1] 2
[1] 0
[1] -2
[1] -4
```

```
>
> for(v in c("sufficiente","buono","distinto","ottimo")) print(v)
[1] "sufficiente"
[1] "buono"
[1] "distinto"
[1] "ottimo"
```

Con il primo ed il secondo for i valori che la variabile n può assumere sono specificati da una sequenza di valori, mentre nel terzo caso sono gli elementi del vettore v. Nell'esempio seguente usiamo lla struttura iterativa for per scrivere una funzione che calcola il fattoriale di x, ossia valuta x!

```
> fact1<-function(x){
+ f<-1
+ if(x<2) return(1)
+ for(n in 2:x){
+       f<-f*n}
+ return (f)
+ }
>
> sapply(0:6,fact1)
[1]   1   1   2   6  24 120 720
```

Nell'esempio precedente è stata usata la funzione sapply(v, funz) che permette di applicare la funzione indicata nel parametro funz a tutti gli elementi di una sequenza o di un vettore v restituendo un vettore di valori.

La sintassi della struttura while è:

> while(condizione) {
>
> corpo del ciclo }

che opera attivando l'esecuzione delle istruzioni indicate nel corpo del ciclo fino a quando la condizione espressa tra le parentesi tonde è verificata. Le istruzioni del corpo del ciclo possono essere scritte su più righe per agevolarne la lettura oppure su un'unica riga separando con un punto e virgola (;) le istruzioni. Un esempio di applicazione della struttura while è il seguente:

```
> n<-1 # condizione iniziale
> while(n<5){
+ print(n^2)
+ n<-n+1
+ }
[1] 1
[1] 4
[1] 9
[1] 16
```

Invece di stampare sullo schermo i risultati del ciclo, è possibile salvarli in un vettore che può essere utilizzato successivamente procedendo come segue:

```
> n<-1
> v<-vector(length=0) # oggetto iniziale
> while(n<5){
+ v[length(v)+1]<-n^2
+ n<-n+1
```

```
+ }
>
> v # visualizza il vettore v
[1]  1  4  9 16
```

L'istruzione v <− vector(length = 0) consente di creare un vettore v di dimensione 0, il cui utilizzo nel ciclo while consente di inserire le potenze nelle posizioni desiderate del vettore. Usiamo ora la struttura iterativa while per scrivere una funzione che calcola x!

```
> fact2<-function(x){
+ f<-1
+ y<-x
+ while(y>1){
+ f<-f*y
+ y<-y-1}
+ return(f)
+ }
>
> sapply(0:6,fact2)
[1]   1   1   2   6  24 120 720
```

La sintassi della struttura repeat è la seguente:

> repeat {
    corpo del ciclo }

Se si utilizza questa struttura di iterazione, la terminazione della ripetizione del ciclo deve essere espressamente indicata all'interno delle istruzioni che costituiscono il corpo del ciclo. Il comando break può essere usato per uscire dal ciclo in cui questa istruzione è racchiusa ed è il solo modo per terminare un ciclo con repeat. L'istruzione next invece provoca il salto immediato all'inizio dell'iterazione successiva. Usiamo ora la struttura iterativa repeat per scrivere una funzione che calcola x!:

```
 > fact3<-function(x){
+ f<-1
+ y<-x
+ repeat{
+ if(y<2) break
+ f<-f*y
+ y<-y-1}
+ return(f)
+ }
>
> sapply(0:6,fact3)
[1]   1   1   2   6  24 120 720
```

R passa l'intera copia dell'oggetto alle funzioni invece di passare un riferimento all'oggetto, come accade in altri linguaggi di programmazione. In R è quindi sconsigliata la programmazione di cicli iterativi tramite una qualsiasi istruzione che fa usa esplicito di cicli a causa della lentezza con cui le iterazioni vengono realizzate dal sistema dovuta alla sequenza di copie e assegnazioni ripetute un numero elevato di volte. Quando è possibile, si consiglia l'uso di

forme alternative per realizzare i cicli che si appoggiano su funzioni, messe a disposizione dal linguaggio R e che lavorano vettorialmente. Ad esempio per calcolare x! per $x = 1, 2, \ldots, 6$ basta utilizzare la funzione prodotto cumulativo cumprod() nel seguente modo:

```
> cumprod(1:6)
[1]   1   2   6  24 120 720
```

Una funzione che calcola x! si può quindi così scrivere:

```
> fact4<-function(x) max( cumprod(1:x) )
>
> sapply(0:6,fact4)
[1]   1   1   2   6  24 120 720
```

1.14 Script, output e directory

Spesso occorre scrivere procedure, funzioni e insiemi di comandi progettati dall'utente da utilizzare in diverse sessioni di R. A tal fine è necessario inserire queste procedure in un file di testo. Nel menù è presente una voce che permette di aprire una finestra in cui scrivere le procedure che si desiderano. Ad esempio supponiamo di scrivere la procedura per il calcolo del fattoriale:

```
> # funzione per il calcolo dei fattoriali
> fact4<-function(x) max( cumprod(1:x) )
```

e di salvare tale file con il nome funzioniNuove.R nella directory corrente dell'utente. Nel menù è presente anche una voce che permette di modificare una procedura precedentemente inserita dall'utente e successivamente di salvarla. Per poter eseguire le funzioni inserite in tale file occorre in primo luogo in primo luogo aprire una nuova sessione di R e caricarle in memoria utilizzando il comando source("funzioniNuove.R") e successivamente eseguirle passando gli opportuni parametri come mostrato nell'esempio seguente:

```
> source("funzioniNuove.R")
> sapply(1:6, fact4)
[1]  1 2 6 24 120 720
```

Alcune volte si rende necessario poter memorizzare l'output di una sessione di R utilizzando la funzione sink(). Ad esempio, la sequenza di comandi

```
> sink("outputFunzioniNuove.R")
> fact4 <- function(x) max( cumprod(1:x) )
> sapply(1:8, fact4)
> sink()
```

crea un file outputFunzioniNuove.R nella directory corrente dell'utente in cui è inserito il testo

```
[1]  1 2 6 24 120 720 5040 40320
```

che può essere successivamente visualizzato. La funzione sink() serve per chiudere il file di output.
Il menù consente di salvare l'area di lavoro su un file di testo outputSessione.history contenente l'intero elenco dei comandi utilizzati in una sessione e che può essere di aiuto nella creazione degli script. Da un file outputSessione.history è anche possibile caricare le istruzioni, che potranno essere richiamate dalla console di R.

In generale la directory può essere differente da quella in cui si trova l'eseguibile di R. È possibile verificare la directory corrente di lavoro attraverso la funzione getwd() e visionarne il contenuto attraverso la funzione dir() che mostra i nomi dei file e delle directory contenute all'interno della directory corrente. Poiché spesso si utilizzano differenti directory di lavoro, per spostarsi da una directoty all'altra si utilizzano i comandi

```
> wd < −"C : \\Magistrale\\SAD\\ProgettoSAD"
> setwd(wd)
```

Spesso occorre importare dati che sono disponibili in un file ottenuto da un qualsiasi altro programma. La funzione read.table() permette di leggere l'intero data frame. Se la prima riga del file contiene l'intestazione delle variabili, allora read.table() interpreterà la prima riga del file come una riga in cui sono contenuti i nomi delle variabili, assegnando ciascun nome alle variabili (colonne) del data frame. Ad esempio, i seguenti comandi:

```
> tabella <− read.table(file = "Tabella.txt",
+      header = TRUE, sep = " ", na.strings = "NA", dec = ".")
```

permettono di importare il file Tabella.txt e automaticamente convertirlo in un data frame tabella. L'argomento header = TRUE specifica che nella prima riga del file Tabella.txt sono contenuti i nomi delle variabili. L'argomento sep = " " indica che i diversi campi sono separati da uno spazio. Inoltre l'argomento na.strings = "NA" è utile se nel file importato sono presenti valori mancanti individuati da NA ed infine l'argomento dec = "." specifica il tipo di carattere utilizzato nel file per separare i numeri decimali.

Tale tabella può essere poi trasformata in una matrice attraverso l'istruzione

```
> dataMatrix < −as.matrix(tabella)
```

Su questa matrice dei dati si può successivamente iniziare ad operare utilizzando il linguaggio R.

Capitolo 2

Grafici e tabelle di frequenza

2.1 Introduzione

Il sistema R è dotato di un sofisticato ambiente grafico che permette di creare grafici per illustrare i risultati di elaborazioni statistiche. Con R è possibile impostare parametri per modificare ogni aspetto di un grafico, simboli, colori ed esportare nei più comuni formati, sia vettoriali (quali .ps, .pdf) che bitmap (.bmp, .jpg). Per avere un'idea delle potenzialità offerte dall'ambiente grafico di R, basta visualizzare la dimostrazione riassuntiva che può essere ottenuta digitando il comando demo(graphics) e digitando ripetutamente il tasto "Invio" per procedere alla visualizzazione delle immagini in successione.

In questo capitolo vedremo come è possibile in R rappresentare i dati contenuti in vettori, fattori, matrici, data frame e serie temporali (serie storiche). Inoltre, vedremo come costruire tabelle di frequenza e grafici di alta qualità. Le tabelle di frequenza univariate e bivariate rivestono un ruolo fondamentale nell'analisi della distribuzione dei dati riguardanti un determinato fenomeno. Con R è semplice costruire tabelle di frequenza univariate e bivariate e analizzare i dati in esse contenuti.

In statistica descrittiva le variabili possono essere di tre tipi: *variabili qualitative*, *variabili quantitative* e *variabili ordinabili*. Nel seguito analizzeremo le variabili univariate e bivariate.

2.2 Grafici per vettori numerici

Consideriamo un vettore $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ contenente dati di tipo quantitativo, ossia un vettore con dati numerici. In questo caso la funzione plot(x) illustra l'andamento del valori assunti dal vettore rispetto ai relativi indici. Ad esempio, consideriamo i voti di 30 studenti universitari che inseriamo nel vettore voti. Il seguente codice

```
> voti<-c(18,19,20,30,29,28,21,22,23,27,26,25,24,25,
+ 26,24,23,22,27,28,21,24,25,25,27,19,21,28,29,28)
>
```

```
> plot(voti,ylab="Voti di 30 studenti universitari",col="red")
```

produce il grafico in Figura 2.1, che illustra il voto riportato da ognuno dei 30 studenti individuati dalle loro posizioni nel vettore.

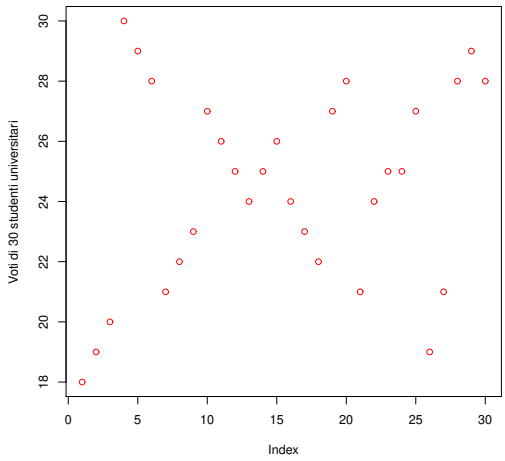

Figura 2.1: Rappresentazione dei valori assunti dal vettore numerico "voti" in corrispondenza dei propri indici.

Vogliamo ora connettere mediante linee i punti della Figura 2.1 creando una *serie storica* dei voti. A tal fine, utilizziamo la funzione plot() con l'opzione type = l, che permette di creare delle linee interconnesse. Quindi, il comando

```
> plot(voti,type="l",ylab="Voti di 30 studenti universitari",col="
    blue")
```

produce il grafico illustrato in Figura 2.2.

Si può anche procedere in modo diverso utilizzando il comando lines(). Infatti i comandi

```
> plot(voti,ylab="Voti di 30 studenti universitari",col="red")
> lines(voti,col="blue")
```

producono il grafico mostrato in Figura 2.3.

2.3 Serie temporali

Una serie temporale (serie storica) può essere memorizzata in un vettore se i dati sono univariati oppure in una matrice o in un data frame nel caso di dati multivariati. R dispone di una funzione specifica, denominata ts(), ossia *time series*, per rappresentare i valori di una serie temporale insieme ad alcune altre

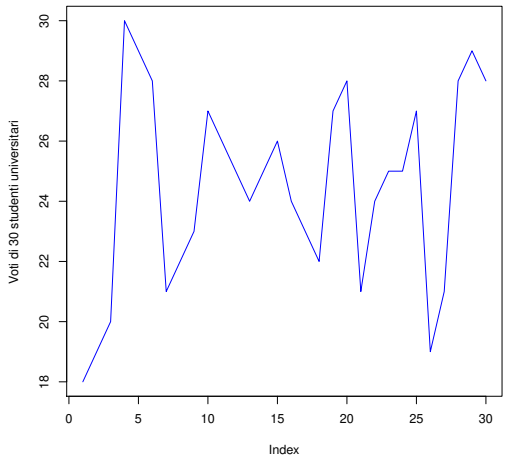

Figura 2.2: Rappresentazione dei valori assunti dal vettore numerico "voti" in corrispondenza dei propri indici mediante linee interconnesse.

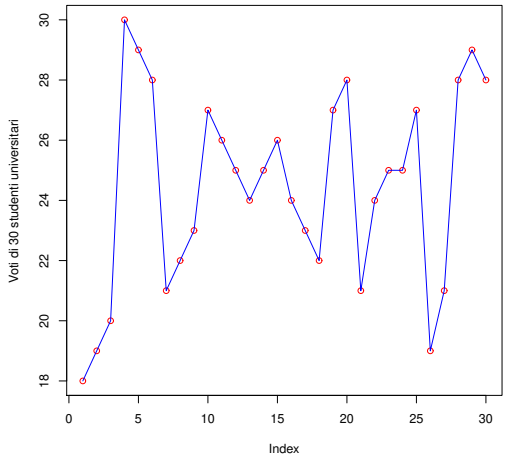

Figura 2.3: Rappresentazione dei valori assunti dal vettore numerico "voti" in corrispondenza dei propri indici mediante linee interconnesse.

caratteristiche (inizio, fine, eventuale periodicità stagionale, ...). La funzione che permette di definire una serie temporale è:

```
y<- ts(x, start=, frequency=, deltat=, end=)
```

dove

x : valori della serie (vettore nel caso univariato, matrice o data frame nel caso multivariato);

start : istante iniziale temporale (ad esempio, 2010);

frequency : numero di osservazioni nell'unità di tempo;

deltat : la distanza temporale tra le osservazioni;

end : istante finale.

La funzione plot(y) permette successivamente di rappresentare il grafico della serie temporale creata.

Il parametro opzionale start serve per stabilire l'istante temporale a cui deve essere riferita la prima osservazione. Ad esempio, supponiamo che i valori contenuti in x siano una *serie storica annuale* e che la prima osservazione sia da riferire al 2010. Allora, si può utilizzare

```
y <- ts(x, start=2010, frequency=1)
```

R calcola automaticamente l'anno finale in base alla lunghezza del vettore x e gli anni sono rappresentati sull'asse delle ascisse nel grafico realizzato con la funzione plot(y). Ad esempio, il codice seguente

```
> osservazioni <- c(10,20,50,60,90,50,80,40,20,25,70,25)
> time <- ts(osservazioni,start = 2010, frequency = 1)
>
> time # visualizza la serie temporale
Time Series:
Start = 2010
End = 2021
Frequency = 1
 [1] 10 20 50 60 90 50 80 40 20 25 70 25
>
> plot(time, main = "Serie temporale", col="red", type="o",
+ ylab="Osservazioni annuali")
```

produce il grafico in Figura 2.4 in cui la frequenza delle osservazioni è annuale.

Il parametro opzionale deltat serve per indicare la distanza nel tempo tra le osservazioni e dipende dall'unità temporale scelta. Ad esempio, supponiamo che le osservazioni nel vettore x iniziano nel 2010, abbiano *cadenza semestrale* e l'anno è scelto come unità temporale. Allora, si può utilizzare

```
y <- ts(x, start=2010, deltat=0.5)
```

o equivalentemente

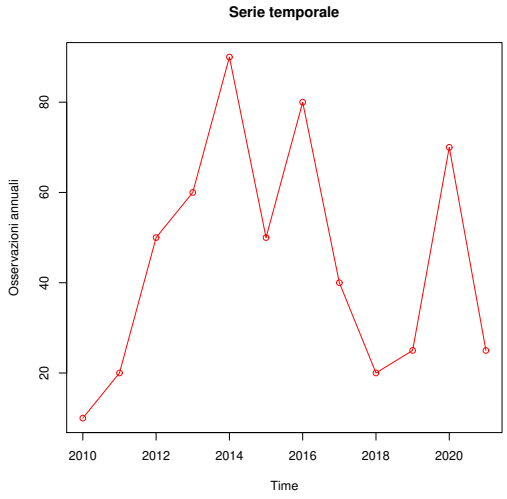

Figura 2.4: Osservazioni annuali a partire dal 2010.

```
y <- ts(x, start=2010, frequency=2)
```

Si nota che frequency rappresenta il reciproco di deltat, ossia fornisce il numero di osservazioni per unità di tempo. Nel caso semestrale delle osservazioni, la frequenza è 2 e la distanza temporale tra le osservazioni è 0.5. Ad esempio, il codice seguente

```
> osservazionisemestrali <- c(10, 20,50,60,90,50, 80,40,20,25,70,25)
> time <- ts(osservazioni,start = 2010, frequency = 2)
>
> time # visualizza la serie temporale
Time Series:
Start = c(2010, 1)
End = c(2015, 2)
Frequency = 2
 [1] 10 20 50 60 90 50 80 40 20 25 70 25
>
> plot(time, main = "Serie temporale", col="blue", type="o",
+ ylab="Osservazioni semestrali")
```

o equivalentemente il codice

```
> osservazionisemestrali <- c(10, 20,50,60,90,50, 80,40,20,25,70,25)
> time <- ts(osservazioni,start = 2010, delta = 0.5)
>
> plot(time, main = "Serie temporale", col="blue", type="o",
+ ylab="Osservazioni semestrali")
```

producono il grafico in Figura 2.5 in cui la frequenza delle osservazioni è semestrale. Se la prima osservazione corrisponde al secondo semestre dell'anno 2010

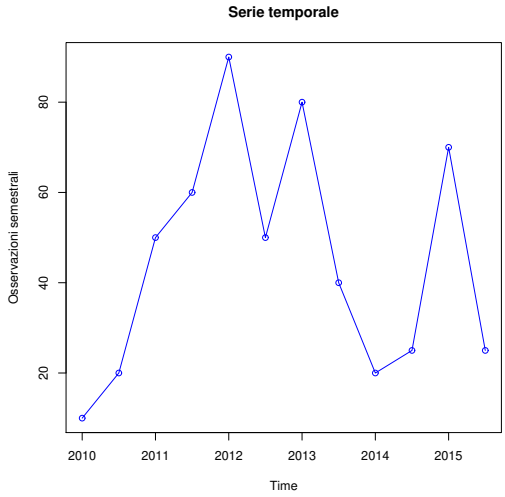

Figura 2.5: Osservazioni semestrali a partire dal 2010.

si può procedere nel seguente modo:

```
> osservazionisemestrali <- c(20,50,60,90,50, 80,40,20,25,70,25)
> time <- ts(osservazionisemestrali,start = c(2010,2), frequency = 2)
>
> time # visualizza la serie temporale
Time Series:
Start = c(2010, 2)
End = c(2015, 2)
Frequency = 2
 [1] 20 50 60 90 50 80 40 20 25 70 25
 >
> plot(time, main = "Serie temporale", col="blue", type="o",
+ ylab="Osservazioni semestrali")
```

che produce il il grafico in Figura 2.6 in cui la frequenza delle osservazioni è semestrale.

Possiamo anche riferirci ad osservazioni mensili. Ad esempio, consideriamo delle osservazioni mensili che partono dal quarto mese del 2010. Il codice seguente

```
> osservazionimensili <- c(10,20,50,60,90,50,80,40,20,25,70,25,
+ 15,25,55,65,95,55)
> time<-ts(osservazionimensili,start=c(2010,4),frequency=12)
>
> time # visualizza la serie temporale
     Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
2010                 10  20  50  60  90  50  80  40  20
2011  25  70  25  15  25  55  65  95  55
```

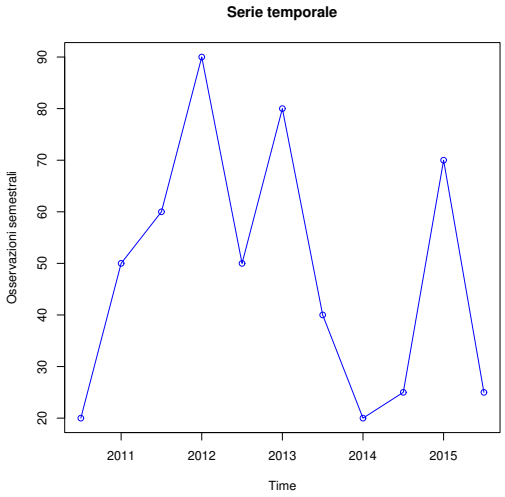

Figura 2.6: Osservazioni semestrali a partire dal secondo semestre del 2010.

È possibile inoltre confrontare sullo stesso grafico diverse serie temporali rappresentate su differenti colonne di un data frame. Ad esempio, il codice

```
> osservazioni <- data.frame(serie1=c(10, 20,50,60,90,50),
+ serie2=c(80,40,20,25,70,25))
> time <- ts(osservazioni,start = 2010, frequency = 1)
>
> time
Time Series:
Start = 2010
End = 2015
Frequency = 1
     serie1 serie2
2010     10     80
2011     20     40
2012     50     20
2013     60     25
2014     90     70
2015     50     25
>
> plot(time, main = "Confronto annuale per due gruppi",
+ col="magenta", type="o")
```

produce il grafico in Figura 2.7 in cui la frequenza delle osservazioni delle due serie temporali è annuale.

2.4 Grafici per le colonne di matrici o data frame

Supponiamo ora di disporre di una matrice di dati numerici. A partire da essa è possibile creare dei vettori contenenti gli elementi delle singole colonne.

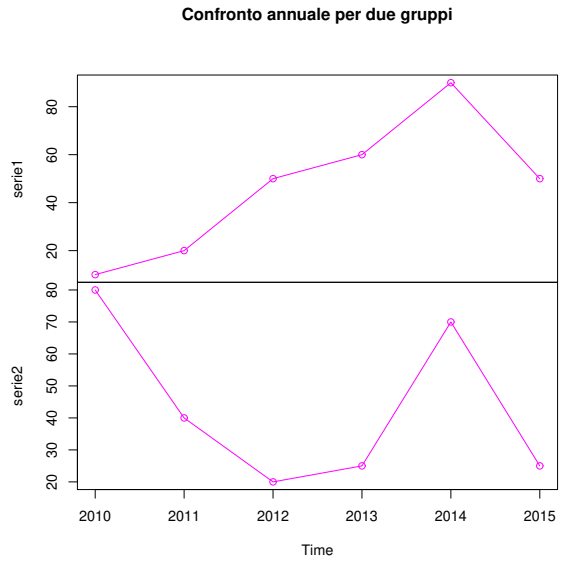

Figura 2.7: Confronto delle osservazioni annuali di due gruppi a partire dal 2010.

Utilizzando poi la funzione barplot() è possibile creare dei grafici a barre, come mostrato nell'esempio seguente in cui per 5 individui vengono forniti il peso, l'altezza e la presenza o assenza di una specifica caratteristica.

```
> m<-cbind(c(60,65,55,60,50), c(1.75,1.65,1.70,1.60,1.80),
+ c(1,0,0,1,1))
> rownames(m)<-c("I1","I2","I3","I4","I5")
> colnames(m)<-c("Peso","Altezza","Presenza")
> m
   Peso Altezza Presenza
I1   60    1.75        1
I2   65    1.65        0
I3   55    1.70        0
I4   60    1.60        1
I5   50    1.80        1
>
> b1<-m[,1] # peso dei 5 individui
> b1
I1 I2 I3 I4 I5
60 65 55 60 50
>
> b2<-m[,2] # altezza dei 5 individui
> b2
  I1   I2   I3   I4   I5
1.75 1.65 1.70 1.60 1.80
>
> b3<-m[,3] # presenza/assenza di una  caratteristica dei 5 individui
> b3
```

```
I1 I2 I3 I4 I5
 1  0  0  1  1
>
> barplot(b1,ylab="Peso", col=1:5)
> barplot(b2,ylab="Altezza", col=1:5)
```

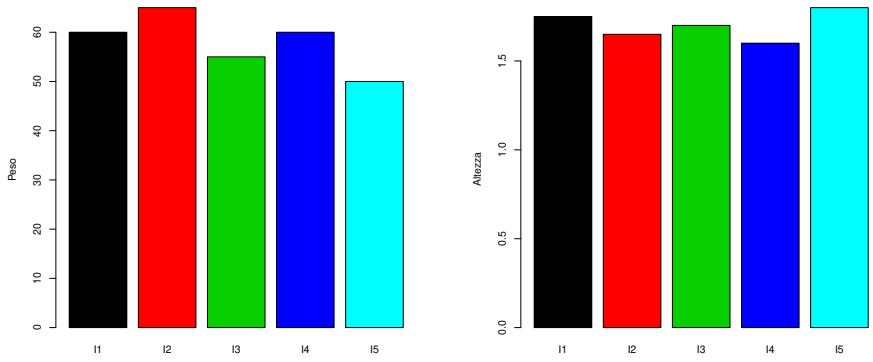

Figura 2.8: Rappresentazione delle prime due colonne della matrice m.

In Figura 2.8 sono mostrati i grafici a barre relativi alle prime due colonne della matrice m, relativi al peso e l'altezza dei 5 individui.

Le unità di misura in una matrice dei dati sono importanti e spesso occorre procedere ad una *standardizzazione dei dati* per avere dei numeri puri, ossia privi di unità di misura. Tale standardizzazione può avvenire in diversi modi. Un semplice modo consiste nel dividere gli elementi di ogni colonna per la somma dei valori della colonna presa in considerazione. Ad esempio, relativamente all'esempio precedente

```
> peso<-c(60,65,55,60,50)
> altezza<-c(1.75,1.65,1.70,1.60,1.80)
> presenza<- c(1,0,0,1,1)
>
> mrel<-cbind(peso/sum(peso),altezza/sum(altezza),
+ presenza/sum(presenza))
> colnames(mrel)<-c("Peso","Altezza","Presenza")
> rownames(mrel)<-c("I1","I2","I3","I4","I5")
>
> mrel
        Peso   Altezza  Presenza
I1 0.2068966 0.2058824 0.3333333
I2 0.2241379 0.1941176 0.0000000
I3 0.1896552 0.2000000 0.0000000
I4 0.2068966 0.1882353 0.3333333
I5 0.1724138 0.2117647 0.3333333
>
> b1<-mrel[,1]
> b2<-mrel[,2]
```

```
barplot(b1,ylab="Peso relativo", col=rainbow(5))
barplot(b2,ylab="Altezza Relativa", col=rainbow(5))
```

Si nota che la somma degli elementi di ogni colonna della matrice mrel è unitaria. In Figura 2.9 sono mostrati i grafici a barre relativi alle prime due colonne della

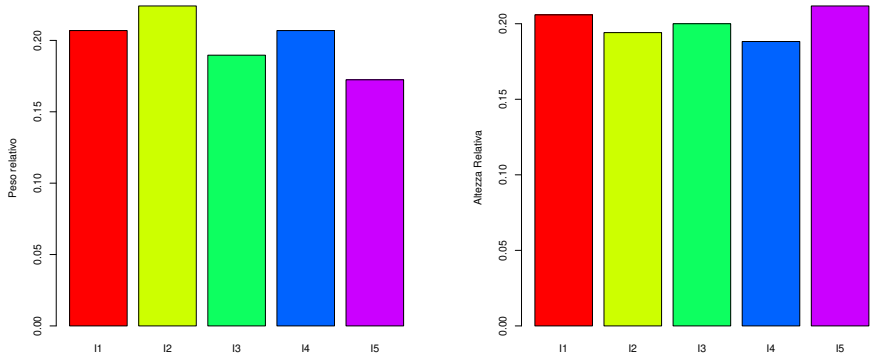

Figura 2.9: Rappresentazione delle prime due colonne della matrice mrel.

matrice mrel.

Se invece si è interessati a grafici a barre per le *percentuali* basta moltiplicare per 100 tutti gli elementi della matrice mrel. In questo caso, la somma degli elementi di ogni colonna della matrice delle percentuali è uguale a 100.

Gli stessi grafici di Figura 2.9 si possono ottenere utilizzando un data frame invece di una matrice:

```
> df<-data.frame(peso=c(60,65,55,60,50),
+ altezza=c(1.75,1.65,1.70,1.60,1.80),  presenza=c(1,0,0,1,1))
> rownames(df)<-c("I1","I2","I3","I4","I5")
>
> dfrel<-data.frame(peso=df$peso/sum(df$peso),
+ altezza=df$altezza/sum(df$altezza),
+ presenza=df$presenza/sum(df$presenza))
>
> barplot(dfrel$peso,ylab="Peso relativo",
+ names.arg=c("I1","I2","I3","I4","I5"),col=rainbow(5))
>
> barplot(dfrel$altezza,xlab="Altezza relativa",
+ names.arg=c("I1","I2","I3","I4","I5"),col=rainbow(5))
```

Il parametro names.arg definisce un vettore di nomi che appaiono sotto ogni barra nel grafico a barre. Questo parametro è anche utilizzato per colorare le barre nel grafico.

2.5 Grafici per coppie di variabili: scatterplot

Consideriamo un campione $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$ costituito da n coppie di osservazioni di tipo quantitativo, ossia coppie di tipo numerico. Le relazioni tra coppie di dati quantitativi possono essere rappresentate graficamente mediante *diagrammi di dispersione* (*scatterplot*) in cui ogni coppia di osservazioni viene rappresentata sotto forma di un punto in un piano euclideo. Dopo aver scelto la variabile da porre sulle *ascisse* (*variabile indipendente*) e la variabile da porre sulle *ordinate* (*variabile dipendente*), si disegnano dei punti in corrispondenza delle coppie. Il risultato finale è una *nuvola di punti* che può essere ottenuto con la funzione plot(x, y). Il grafico che si ottiene mira ad evidenziare se *le coppie di punti presentano qualche forma di regolarità*. Inoltre, il grafico di dispersione tende ad evidenziare se *esiste una relazione tra le variabili* e di che tipo è tale relazione (lineare, quadratica, ...).

Supponiamo, ad esempio, di voler analizzare i dati in Tabella 2.1 che riguardano l'età, l'altezza e il peso di un campione di 30 individui.

Tabella 2.1: Tabella con l'età, l'altezza e il peso di un campione di 30 individui

Età	44	55	36	72	29	24	27	50	49	22
Altezza	150	165	170	180	167	175	158	162	190	155
Peso	55	50	48	46	60	65	70	53	51	42
Età	34	29	48	43	54	23	36	60	65	85
Altezza	158	169	175	178	160	162	166	168	171	160
Peso	54	51	44	50	60	65	68	70	47	45
Età	85	91	18	19	20	70	37	23	90	78
Altezza	156	154	165	190	165	166	167	162	171	162
Peso	45	44	70	71	65	67	58	57	45	47

Le seguenti linee di codice

```
> df<-data.frame(
+ eta=c(44,55,36,72,29,24,27,50,49,22,34,29,48,
+ 43,54,23,36,60,65,85,85,91,18,19,20,70,37,23,90,78),
+ altezza=c(150,165,170,180,167,175,158,162,190,155,
+ 158,169,175,178,160,162,166,168,171,160,156,
+ 154,165,190,165,166,167,162,171,162),
+ peso=c(55,50,48,46,60,65,70,53,51,42,54,51,44,50,
+ 60,65,68,70,47,45,45,44,70,71,65,67,58,57,45,47))
> df # visualizza il data frame
   eta altezza peso
1   44     150   55
2   55     165   50
3   36     170   48
4   72     180   46
5   29     167   60
6   24     175   65
7   27     158   70
8   50     162   53
9   49     190   51
10  22     155   42
```

```
11   34      158    54
12   29      169    51
13   48      175    44
14   43      178    50
15   54      160    60
16   23      162    65
17   36      166    68
18   60      168    70
19   65      171    47
20   85      160    45
21   85      156    45
22   91      154    44
23   18      165    70
24   19      190    71
25   20      165    65
26   70      166    67
27   37      167    58
28   23      162    57
29   90      171    45
30   78      162    47
```

definiscono un data frame contenente l'età, il peso e l'altezza dei 30 individui. Realizziamo lo scatterplot considerando l'età come variabile indipendente ed l'altezza come variabile dipendente, disegnando 30 punti in uno spazio euclideo con la seguente linea di codice:

```
> plot(df$eta,df$altezza, main="Altezza in funzione dell'eta' ",
+ xlab="Eta'  ",ylab="Altezza (cm)", xlim=c(0,100),col="red")
```

che produce il grafico alla sinistra di Figura 2.10, mentre la linea di codice

```
> plot(df$eta,df$peso, main="Peso in funzione dell'eta' ",
+ xlab="Eta' ",ylab="Peso (kg)", xlim=c(0,100),col="blue")
```

produce invece il grafico alla destra di Figura 2.10.

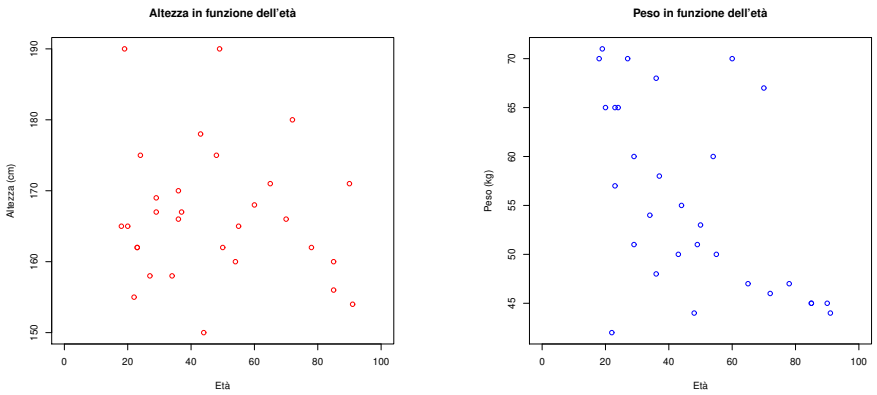

Figura 2.10: Rappresentazione grafica dell'altezza (alla sinistra) e del peso (alla destra) in funzione dell'età.

La funzione pairs() è in grado di visualizzare in un'unica finestra grafica una pluralità di grafici per punti ottenuti mettendo in relazione tutte le coppie di variabili quantitative definite all'interno di un data frame (o di una matrice). Riferendosi all'esempio precedente, la seguente linea di codice

```
> pairs(df,main="Scatterplot per le coppie di variabili")
```

produce il grafico visualizzato in Figura 2.11. Le diverse immagini illustrano le nuvole di punti che si ottengono prendendo in considerazione tutte le differenti coppie di variabili.

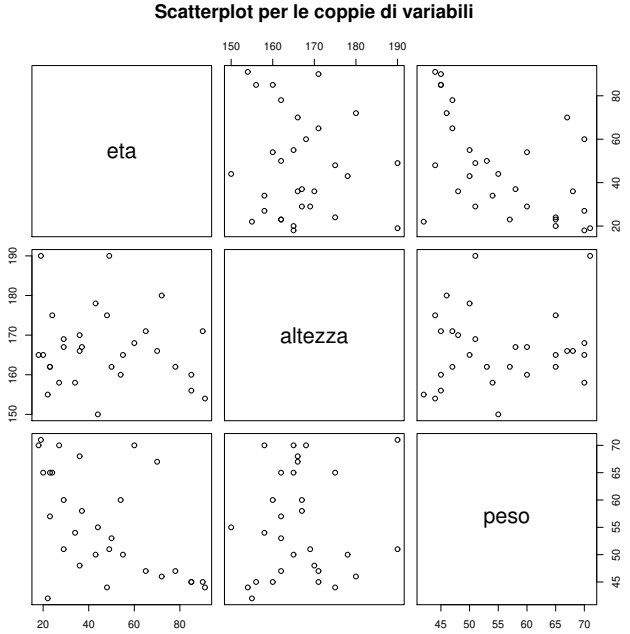

Figura 2.11: Rappresentazione grafica delle relazioni esistenti tra età, altezza e peso ottenute con la funzione pairs().

2.6 Distribuzioni di frequenza

Per quanto riguarda la distribuzione di frequenza, consideriamo una variabile X e indichiamo con $z_1, z_2, \ldots, z_k$ le modalità distinte da essa assunte. Se la variabile X è qualitativa le modalità indicano delle qualità distinte degli individui, mentre se la variabile X è quantitativa le modalità sono dei numeri reali distinti. Consideriamo poi un campione $(x_1, x_2, \ldots, x_n)$ costituito da n *osservazioni* di X. Se indichiamo con n_i il numero di volte in cui ciascuna modalità z_i è presente nel campione, ossia la *frequenza assoluta* con cui essa appare nel campione,

l'insieme $\{(z_i, n_i), i = 1, 2, \ldots, k\}$ si chiama *distribuzione di frequenza*. Se non esistono dati mancanti, la somma delle frequenze assolute è sempre uguale alla numerosità del campione, ossia $n = n_1 + n_2 + \ldots + n_k$. È possibile calcolare anche le *frequenze relative*

$$f_i = \frac{n_i}{n} \qquad (i = 1, 2, \ldots, k).$$

Se non esistono dati mancanti la somma delle frequenze relative è sempre unitaria, ossia $f_1 + f_2 + \ldots + f_k = 1$.

Per una *variabile qualitativa*, in R la costruzione di una distribuzione di frequenza viene effettuata utilizzando la funzione table(). Ad esempio, consideriamo un campione di 36 elementi che costituiscono delle osservazioni di una variabile qualitativa uomo con quattro modalità (bambino, giovane, adulto, anziano):

```
> uomo<-c(rep("bambino",8),rep("giovane",3),rep("adulto",4),
+ rep("giovane",9),rep("anziano",6),rep("bambino",2),
+ rep("adulto",4))
>
> table(uomo) # calcola le frequenze assolute
uomo
 adulto anziano bambino giovane
      8       6      10      12
```

Nell'esempio è stato utilizzato il comando rep() per non ripetere più volte le varie modalità nel vettore. La funzione table() riporterà le modalità della variabile qualitativa che presentano un valore di frequenza assoluta diverso da zero. Si nota che table(uomo) ordina le *modalità* della variabile qualitativa uomo in ordine alfabetico.

Se si vuole ottenere la distribuzione delle frequenze relative basta utilizzare il comando

```
> table(uomo)/length(uomo) # calcola le frequenze relative
uomo
   adulto   anziano   bambino   giovane
0.2222222 0.1666667 0.2777778 0.3333333
```

Se si prende in esame una *variabile ordinabile*, bisogna considerarla come un fattore e ordinare i livelli di questo fattore prima di costruire la distribuzione di frequenza. Riferendosi all'esempio precedente si ha:

```
> persona<-ordered(uomo,levels=c("bambino","giovane","adulto","
    anziano"))
> table(persona)
persona
bambino giovane  adulto anziano
     10      12       8       6
```

La *frequenza assoluta cumulata* è definita come segue:

$$N_i = n_1 + n_2 + \ldots + n_i \qquad (i = 1, 2, \ldots, k),$$

mentre la *frequenza relativa cumulata* è

$$F_i = f_1 + f_2 + \ldots + f_i \qquad (i = 1, 2, \ldots, k).$$

Per calcolare le *frequenze assolute cumulate* si deve utilizzare la funzione cumsum() che permette di calcolare le somme cumulate degli elementi di un vettore. Riferendosi all'esempio precedente le seguenti linee di codice

```
> cumsum(table(persona))
bambino giovane  adulto anziano
     10      22      30      36
>
> cumsum(table(persona)/length(persona))
  bambino   giovane    adulto   anziano
0.2777778 0.6111111 0.8333333 1.0000000
```

permettono di calcolare le frequenze assolute cumulate e le frequenze relative cumulate.

Le frequenze assolute e relative possono essere calcolate anche per *variabili quantitative* sempre che il numero di modalità distinte sia ben definito (ad esempio, i voti dal 18 al 30). Spesso si preferisce raccogliere le informazioni riguardanti *variabili quantitative* o numeriche in classi e calcolare le *frequenze assolute o relative per classi*, ossia le frequenze con cui gli elementi del vettore cadono nelle diverse classi. Per fare questo si utilizza la funzione cut() che permette di raggruppare i dati relativi ad un vettore in intervalli elencando nel parametro breaks gli estremi degli intervalli che sono aperti a sinistra e chiusi a destra. Se desideriamo ottenere intervalli chiusi a sinistra e aperti a destra occorre specificare in cut() l'opzione aggiuntiva right = FALSE. Nell'esempio seguente si calcola la frequenza assoluta con cui un assegnato insieme di voti è suddiviso nelle quattro classi: $(17, 21]$, $(21, 24]$, $(24, 27]$, $(27, 31]$ per 49 studenti:

```
> votiStudenti<-c(rep(18,6),rep(21,8),rep(25,10),rep(28,6),
+ rep(30,7),rep(25,2),rep(24,6),rep(25,4))
>
> table(cut(votiStudenti,breaks=c(17,21,24,27,31)))

(17,21] (21,24] (24,27] (27,31]
     14       6      16      13
```

2.7 Grafici a barre, a bastoncini e diagrammi a torta

Consideriamo una *variabile qualitativa* X e indichiamo con $z_1, z_2, \ldots, z_k$ le modalità distinte da essa assunte. Consideriamo poi un campione $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ costituito da n osservazioni di X. Disponiamo sull'asse orizzontale ed in modo equispaziato le modalità assunte da X e sull'asse verticale riportiamo le frequenze assolute o le frequenze relative. Tracciamo dei rettangoli centrati sulle modalità z_i tutti della stessa base e altezza pari alle frequenze (assolute o relative), ottenendo un grafico (o diagramma) a barre. Grafici di questo tipo sono di solito utilizzati per visualizzare i valori di una qualche quantità (ad esempio,

la frequenza) per diverse modalità o categorie. In R si ottiene un *grafico a barre* utilizzando barplot(table(x)). Le seguenti linee di codice

```
> uomo<-c(rep("bambino",8),rep("giovane",3),rep("adulto",4),
+ rep("giovane",9),rep("anziano",6),rep("bambino",2),rep("adulto"
    ,4))
>
> barplot(table(uomo),col=1:4)
```

producono il grafico a barre illustrato in Figura 2.12. Per colorare i rettangoli con differenti colori basta aggiungere in barplot() il parametro col = 1 : 4, essendo 4 le modalità considerate. Si nota che le modalità della variabile uomo sono state ordinate nella Figura 2.12 in ordine alfabetico.

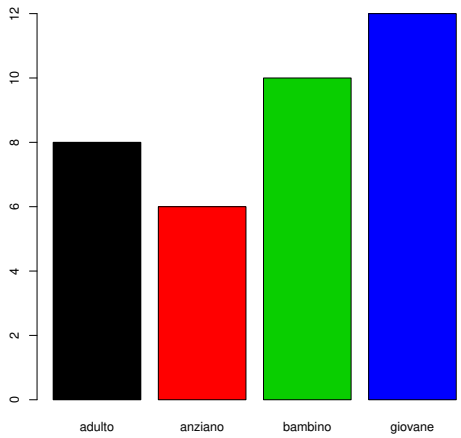

Figura 2.12: Frequenza assoluta della variabile qualitativa "uomo" tramite un grafico a barre.

Affinché le modalità siano ordinate in modo differente occorre trasformare il vettore uomo in un fattore uomo1. Le seguenti linee di codice

```
> uomo1<-ordered(uomo,levels=c("bambino","giovane","adulto","
    anziano"))
>
> barplot(table(uomo1),col=1:4)
```

producono il grafico a barre illustrato in Figura 2.13; si nota che le modalità della variabile uomo sono state ordinate come specificato in levels.

Per la variabile qualitativa X è possibile anche costruire un *grafico a bastoncini* utilizzando il comando plot(table(x)). Relativamente, all'esempio precedente la linea di codice

```
> plot(table(uomo),col=1:4)
```

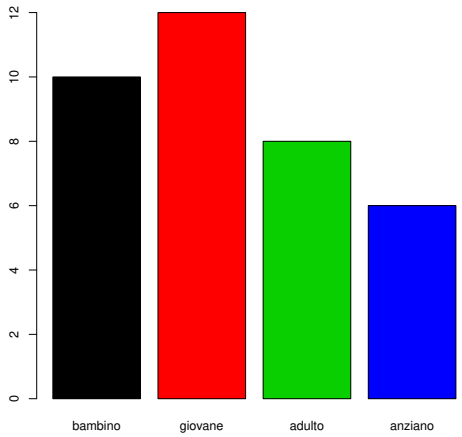

Figura 2.13: Frequenza assoluta della variabile qualitativa "uomo" tramite un grafico a barre ordinando le modalità.

produce il grafico a bastoncini illustrato in Figura 2.14. Si nota che le modalità della variabile uomo sono state ordinate nella Figura 2.14 in ordine alfabetico. Inoltre le linea di codice

```
> uomo1<-ordered(uomo,levels=c("bambino","giovane","adulto","
    anziano"))
> plot(table(uomo1),col=1:4)
```

producono il grafico a bastoncini illustrato in Figura 2.15; si nota che le modalità della variabile uomo sono state ordinate come specificato in levels.

Se si desidera un grafico a bastoncini che riporti le frequenze relative piuttosto che le frequenze assolute si può utilizzare il comando plot(table(x)/length(x)). Relativamente, all'esempio precedente la linea di codice

```
> uomo1<-ordered(uomo,levels=c("bambino","giovane","adulto","
    anziano"))
>
> plot(table(uomo1)/length(uomo1),col=1:4)
```

produce il grafico a bastoncini illustrato in Figura 2.16.

Si possono anche realizzare grafici a barre per variabili qualitative ordinabili utilizzando la funzione plot() invece della funzione barplot(). Ad esempio, se si considera il titolo finale di studio posseduto da un campione di 75 persone, le seguenti linee di codice

```
> titolo<-c(rep("nessuno",5),rep("elementare",15),rep("media",25),
+ rep("diploma",20),rep("laurea",10))
>
> titoloFinale<-ordered(titolo,levels=c("nessuno","elementare",
```

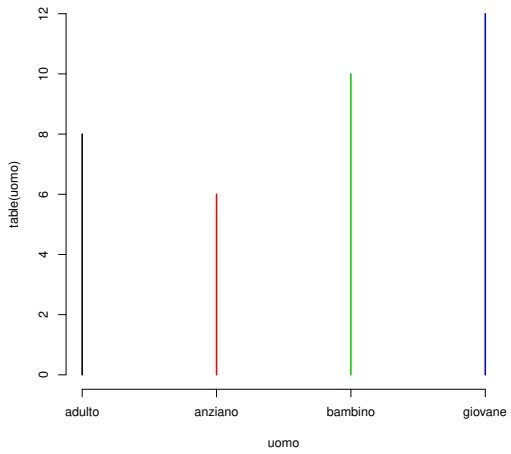

Figura 2.14: Frequenza assoluta della variabile qualitativa “uomo” tramite un grafico a bastoncini.

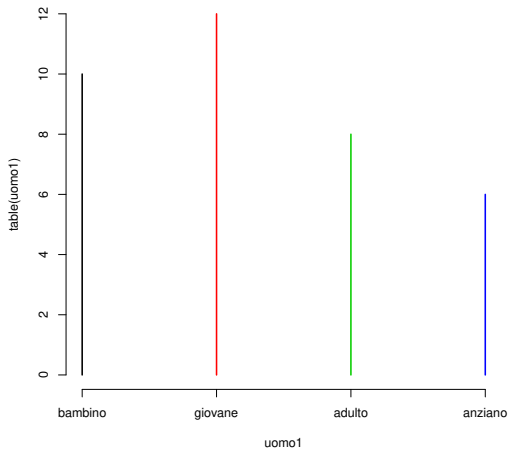

Figura 2.15: Frequenza assoluta della variabile qualitativa “uomo” tramite un grafico a bastoncini ordinando le modalità.

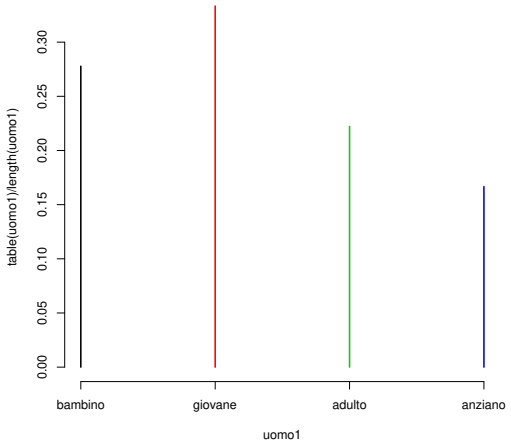

Figura 2.16: Frequenza relativa della variabile qualitativa "uomo" tramite un grafico a bastoncini ordinando le modalità.

```
+ "media","diploma","laurea"))
>
> plot(titoloFinale,xlab="Titolo finale di studi posseduti",col
    =1:5)
```

producono il grafico a barre visualizzato in Figura 2.17.

Un altro tipo di rappresentazione si ottiene mediante i *diagrammi a torta* che permettono di attribuire ciascuna modalità della variabile qualitativa in esame ad un settore circolare di un cerchio, la cui ampiezza è proporzionale alle frequenze. I diagrammi a torta sono quindi particolarmente utili quando i dati non sono numerici ma categorici. Un grafico a torta si costruisce tracciando un cerchio e suddividendolo in tanti settori circolari (fette o spicchi) quante sono le modalità distinte di dati; ogni settore ha un angolo al centro proporzionale alla frequenza (relativa o assoluta) della modalità corrispondente.

Il sistema R sceglie il tipo di diagramma a torta con i diversi settori colorati differentemente utilizzando il comando pie(table(x)), dove x è un vettore o un fattore. Riferendosi all'esempio precedente, il seguente codice

```
> titolo<-c(rep("nessuno",5),rep("elementare",15),rep("media",25),
+ rep("diploma",20),rep("laurea",10))
> pie(table(titolo))
```

produce il diagramma a torta mostrato in Figura 2.18.

Si può anche scegliere un *tratteggio particolare* da utilizzare nei diagrammi a torta per tratteggiare differentemente i diversi settori utilizzando i comandi density = e angle = . Riferendosi al precedente esempio il seguente codice

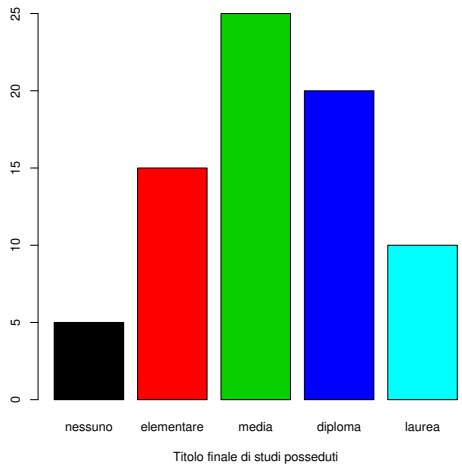

Figura 2.17: Frequenza della variabile qualitativa “titolo” utilizzando un grafico a barre ordinando le modalità.

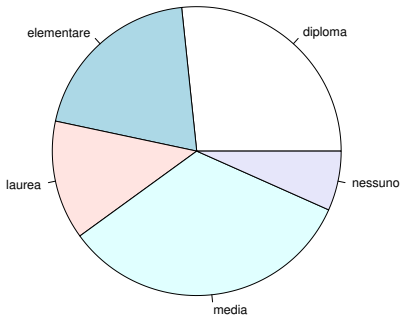

Figura 2.18: Frequenza della variabile qualitativa “titolo” utilizzando un diagramma a torta.

```
> pie(table(titolo),density=10,angle=15+10*(1:5), col=1:5)
```

produce il diagramma a torta mostrato in Figura 2.19.

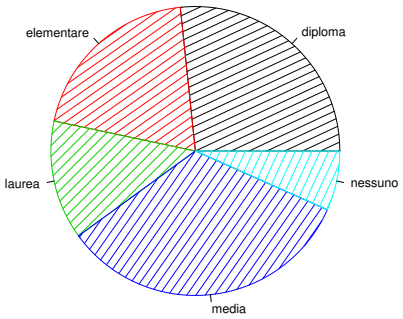

Figura 2.19: Frequenza della variabile qualitativa "titolo" utilizzando un particolare diagramma a torta.

Si nota che i cinque settori sono stati tratteggiati differentemente utilizzando delle linee ugualmente distanziate e inclinate opportunamente con angolazioni di 25, 35, 45, 55, 65 gradi.

Costruiamo ora un diagramma a torta per il titolo di studio del campione di 75 individui utilizzando i valori percentuali.

```
> freqRel<-table(titolo)/length(titolo)
> freqRel # visualizza le frequenze relative
titolo
   diploma elementare     laurea      media    nessuno
0.26666667 0.20000000 0.13333333 0.33333333 0.06666667
>
> percentuali<-freqRel*100
> percentuali # visualizza le percentuali
titolo
   diploma elementare     laurea      media    nessuno
 26.666667  20.000000  13.333333  33.333333   6.666667
> sum(percentuali)
[1] 100
>
> perc<-round(percentuali)
> labelP<-c("diploma","elementare","laurea","media","nessuno")
> labelP<-paste(labelP,perc)
> labelP<-paste(labelP,"%",sep="")
> pie(percentuali,label=labelP, col=rainbow(length(labelP)),
+ main="Valori percentuali")
```

che produce il grafico in Figura 2.20.

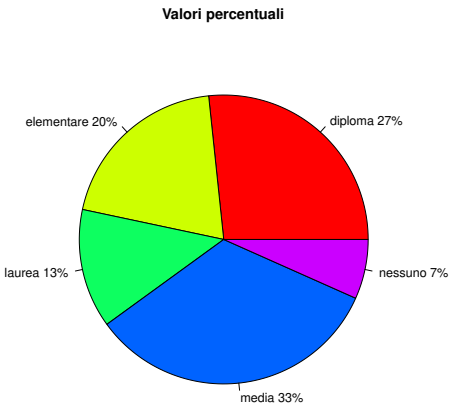

Figura 2.20: Valori percentuali della variabile qualitativa "titolo" utilizzando un diagramma a torta.

Con la funzione round() si arrotondano i numeri. La funzione label() definisce le etichette da inserire nel grafico a torta. La funzione paste() serve a concatenare stringhe e la funzione sep = serve per definire il separatore tra le stringhe (spazio vuoto). La funzione rainbow() è un modo alternativo per selezionare i colori.

Per rappresentare invece correttamente la distribuzione di frequenza di una *variabile quantitativa* X occorre utilizzare il comando plot(table(x)) che produce un grafico a bastoncini in cui sull'asse orizzontale sono riportati i valori e sull'asse verticale le frequenze assolute dei valori distinti assunti nel vettore. Riferendosi ai voti di 30 studenti universitari si ha:

```
> voti<-c(18,19,20,30,29,28,21,22,23,27,26,25,24,25,
+ 26,24,23,22,27,28,21,24,25,25,27,19,21,28,29,28)
> plot(table(voti),ylab="Frequenza dei voti di 30 studenti
    universitari", col=1:13)
```

produce il grafico in Figura 2.21.

Per variabili quantitative si può anche considerare una rappresentazione tramite diagrammi a torta attraverso il comando pie(table(x)), dove x è un vettore o un fattore, anche se tale grafico non si rivela spesso utile. Ad esempio, la seguente linea di codice

```
> pie(table(voti), col = rainbow(13), radius = 0.9, xlab="Voti di
    30 studenti universitari")
```

producono il grafico mostrato in Figura 2.22.

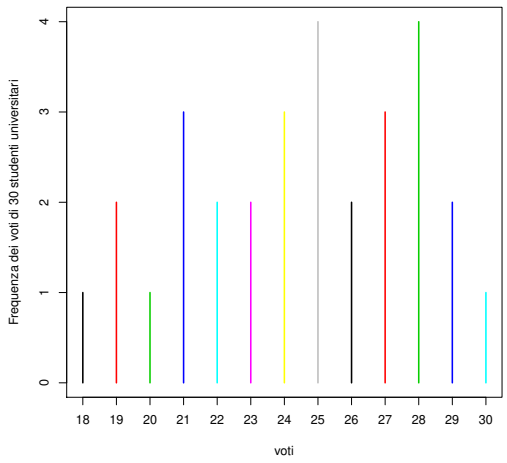

Figura 2.21: Distribuzione di frequenza del vettore numerico "voti".

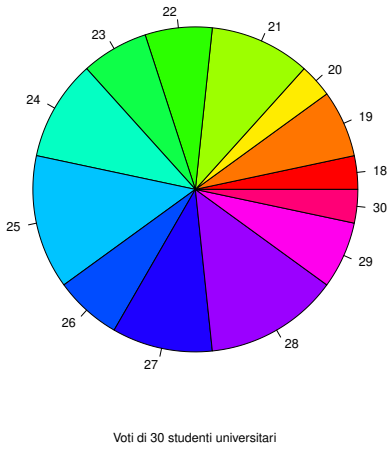

Figura 2.22: Frequenze del vettore "voti" tramite un diagramma a torta.

2.8 Tabelle di contingenza

Per vedere se esistono legami tra due variabili X e Y *rilevati congiuntamente* sugli stessi individui occorre costruire una tabella a doppia entrata detta *tabella di contingenza*. A tal fine sia X una variabile e indichiamo con $z_1, z_2, \ldots, z_h$ le modalità distinte da essa assunte e sia Y un'altra variabile e indichiamo con $w_1, w_2, \ldots, w_k$ le modalità distinte da essa assunte. Consideriamo un campione $\big((x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\big)$ costituito da n *osservazioni* di (X, Y). L'elemento n_{ij} in posizione (i, j) della tabella di contingenza, detto *frequenza congiunta*, rappresenta il numero di volte in cui una particolare coppia di modalità (z_i, w_j) si presenta nel campione. Dalla tabella di contingenza è possibile ottenere la *distribuzione di frequenza marginale di X*:

$$n_{i.} = \sum_{j=1}^{k} n_{i,j} \qquad (i = 1, 2, \ldots, h)$$

e la *distribuzione di frequenza marginale di Y*:

$$n_{.j} = \sum_{i=1}^{h} n_{i,j} \qquad (j = 1, 2, \ldots, k)$$

Tabella 2.2: Tabella di contingenza

X \ Y	w_1	w_2	$\ldots$	w_j	$\ldots$	w_k	
z_1	n_{11}	n_{12}	$\ldots$	n_{1j}	$\ldots$	n_{1k}	$n_{1.}$
z_2	n_{21}	n_{22}	$\ldots$	n_{2j}	$\ldots$	n_{2k}	$n_{2.}$
$\vdots$	$\ddots$	$\ddots$	$\ddots$	$\ddots$	$\ddots$	$\ddots$	$\vdots$
z_i	n_{i1}	n_{i2}	$\ldots$	n_{ij}	$\ldots$	n_{ik}	$n_{i.}$
$\vdots$	$\ddots$	$\ddots$	$\ddots$	$\ddots$	$\ddots$	$\ddots$	$\vdots$
z_h	n_{h1}	n_{h2}	$\ldots$	n_{hj}	$\ldots$	n_{hk}	$n_{h.}$
	$n_{.1}$	$n_{.2}$	$\ldots$	$n_{.j}$	$\ldots$	$n_{.k}$	n

In Tabella 2.2 sono illustrate la distribuzione di frequenza congiunta e le due distribuzioni di frequenza marginali delle variabili X e Y. In R per determinare le distribuzioni di frequenza bivariate il comando da utilizzare è sempre table(). Ad esempio, per un campione di 15 individui, consideriamo una tabella di contingenza supponendo che X individua il colore dei capelli (biondi, castani, neri, rossi) e Y il tipo di capelli (lisci, ondulati, ricci):

```
> colore<-c("biondi","castani","neri","rossi","biondi","castani",
+ "neri","castani","biondi","biondi","castani","castani",
+ "castani","neri","rossi")
>
> tipo<-c("lisci","lisci","lisci","lisci","ondulati","ondulati",
```

```
+ "ondulati","ondulati","ondulati","ondulati","ondulati",
+ "ricci","ricci","lisci","ricci")
>
> table(colore,tipo)
         tipo
colore    lisci ondulati ricci
  biondi      1         3     0
  castani     1         3     2
  neri        2         1     0
  rossi       1         0     1
```

Dopo aver creato la tabella della distribuzione di frequenza congiunta nomeTable, per estrarre la riga i-esima basta utilizzare il comando nomeTable[i,] e per estrarre la colonna j-esima basta utilizzare il comando nomeTable[, j].

Riferendosi all'esempio precedente estraiamo la seconda riga e la terza colonna della tabella di contingenza

```
> capelli<-table(colore,tipo)
>
> capelli[2,] # seconda riga della tabella
   lisci ondulati    ricci
       1        3        2
>
> capelli[,3] # terza colonna della tabella
 biondi castani    neri   rossi
      0       2       0       1
```

I comandi margin.table(nomeTable, 1) e margin.table(nomeTable, 2) permettono rispettivamente di ottenere la *distribuzione di frequenza marginale di* X e la *distribuzione di frequenza marginale di* Y. Riferendosi all'esempio precedente calcoliamo le due distribuzioni di frequenza marginali:

```
> capelli<-table(colore,tipo)
>
> margin.table(capelli,1) # distribuzione marginale del colore dei
    capelli
colore
 biondi castani    neri   rossi
      4       6       3       2
>
> margin.table(capelli,2) # distribuzione marginale del tipo dei
    capelli
tipo
   lisci ondulati    ricci
       5        7        3
```

È anche possibile considerare le *frequenze relative* associate ad una tabella di contingenza. Se si denota con

$$f_{ij} = \frac{n_{ij}}{n} \qquad (i = 1, 2, \ldots, h; j = 1, 2, \ldots, k)$$

le *frequenze relative congiunte* e con

$$f_{i,\cdot} = \sum_{j=1}^{k} f_{ij} = \sum_{j=1}^{k} \frac{n_{ij}}{n} = \frac{1}{n} \sum_{j=1}^{k} n_{ij} = \frac{n_{i\cdot}}{n}$$

$$f_{\cdot,j} = \sum_{i=1}^{h} f_{ij} = \sum_{i=1}^{h} \frac{n_{ij}}{n} = \frac{1}{n} \sum_{i=1}^{h} n_{ij} = \frac{n_{\cdot\, j}}{n}$$

rispettivamente le *frequenze relative marginali di X e di Y*, è possibile costruire la Tabella 2.3 delle frequenze relative.

Tabella 2.3: Tabella delle frequenze relative

X \ Y	w_1	w_2	...	w_j	...	w_k	
z_1	f_{11}	f_{12}	...	f_{1j}	...	f_{1k}	$f_{1\cdot}$
z_2	f_{21}	f_{22}	...	f_{2j}	...	f_{2k}	$f_{2\cdot}$
$\vdots$	$\ddots$	$\ddots$	$\ddots$	$\ddots$	$\ddots$	$\ddots$	$\vdots$
z_i	f_{i1}	f_{i2}	...	f_{ij}	...	f_{ik}	$f_{i\cdot}$
$\vdots$	$\ddots$	$\ddots$	$\ddots$	$\ddots$	$\ddots$	$\ddots$	$\vdots$
z_h	f_{h1}	f_{h2}	...	f_{hj}	...	f_{hk}	$f_{h\cdot}$
	$f_{\cdot 1}$	$f_{\cdot 2}$	...	$f_{\cdot j}$	...	$f_{\cdot k}$	1

In R per calcolare la *distribuzione delle frequenze relative congiunte* si può utilizzare il comando prop.table(nomeTabella) e da questa ricavare le distribuzioni delle frequenze relative marginali.

```
> prop.table(capelli) # frequenze relative congiunte
         tipo
colore          lisci   ondulati      ricci
  biondi   0.06666667 0.20000000 0.00000000
  castani  0.06666667 0.20000000 0.13333333
  neri     0.13333333 0.06666667 0.00000000
  rossi    0.06666667 0.00000000 0.06666667
>
> freqCapelli <- prop.table(capelli)
>
> margin.table(freqCapelli, 1) # distribuzione delle frequenze
    relative marginali del colore dei capelli
colore
   biondi   castani      neri     rossi
0.2666667 0.4000000 0.2000000 0.1333333
>
> margin.table(freqCapelli, 2) # distribuzione delle frequenze
    relative marginali del tipo di capelli
tipo
    lisci  ondulati     ricci
0.3333333 0.4666667 0.2000000
```

Conoscendo la distribuzione delle frequenze relative congiunte e la distribuzione delle frequenze relative marginali è possibile calcolare la *distribuzione delle frequenze relative di Y condizionata dalle modalità assunte da X*, così definita:

$$f(j|i) = \frac{f_{ij}}{f_{i\cdot}} = \frac{n_{ij}}{n_{i\cdot}} \qquad (i = 1, 2, \ldots, h; j = 1, 2, \ldots, k)$$

e la *distribuzione delle frequenze relative di X condizionata dalle modalità assunte da Y*, così definita:

$$f(i|j) = \frac{f_{ij}}{f_{\cdot j}} = \frac{n_{ij}}{n_{\cdot j}} \qquad (i = 1, 2, \ldots, h; j = 1, 2, \ldots, k)$$

Una importante proprietà delle frequenze relative condizionate è:

$$\sum_{j=1}^{k} f(j|i) = 1 \ (i = 1, 2, \ldots, h), \qquad \sum_{i=1}^{h} f(i|j) = 1 \ (j = 1, 2, \ldots, k).$$

Il comando prop.table(nomeTabella, 1) permette invece di ottenere la *distribuzione delle frequenze relative di Y condizionata dalle modalità assunte da X*, mentre il comando prop.table(nomeTabella, 2) permette di ricavare la *distribuzione delle frequenze relative di X condizionata dalle modalità assunte da Y*. Riferendosi all'esempio precedente si ha:

```
> prop.table(capelli, 1) # distribuzione delle frequenze relative
    condizionate f(j|i)
         tipo
colore        lisci  ondulati     ricci
  biondi   0.2500000 0.7500000 0.0000000
  castani  0.1666667 0.5000000 0.3333333
  neri     0.6666667 0.3333333 0.0000000
  rossi    0.5000000 0.0000000 0.5000000
>
> prop.table(capelli, 2) # distribuzione delle frequenze relative
    condizionate f(i|j)
         tipo
colore        lisci  ondulati     ricci
  biondi   0.2000000 0.4285714 0.0000000
  castani  0.2000000 0.4285714 0.6666667
  neri     0.4000000 0.1428571 0.0000000
  rossi    0.2000000 0.0000000 0.3333333
```

Nel primo caso la somma degli elementi di ogni riga è unitaria; invece, nel secondo caso la somma degli elementi di ogni colonna è unitaria.

Si dice che le variabili X e Y sono *indipendenti* se la frequenza relativa congiunta si fattorizza come il prodotto delle due frequenze relative marginali, ossia se risulta:

$$f_{ij} = f_{i\cdot}\, f_{\cdot j} \qquad (i = 1, 2, \ldots, h;\ j = 1, 2, \ldots, k).$$

Questa circostanza si verifica molto raramente sui dati reali.

2.9 Grafici per tabelle di contingenza

Sia X una variabile di *tipo qualitativo* e indichiamo con $z_1, z_2, \ldots, z_h$ le modalità distinte da essa assunte e sia Y un'altra variabile e indichiamo con $w_1, w_2, \ldots, w_k$ le modalità distinte da essa assunte. Considerato un campione

$\big((x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\big)$ costituito da n osservazioni di (X, Y), costruiamo la *tabella di contingenza* contenente nella posizione (i, j) la *frequenza congiunta* n_{ij}, che rappresenta il numero di volte in cui una particolare coppia di valori (z_i, w_j) si presenta nel campione.

Le tabelle di contingenza possono essere rappresentate graficamente in vari modi.

→ Grafici per le frequenze assolute congiunte

Una tabella di contingenza può essere rappresentata mediante un grafico con rettangoli costituito da $h \times k$ rettangoli. L'area totale dei rettangoli è proporzionale al numero di casi. Il rettangolo in posizione (i, j) rappresenta l'elemento nella stessa posizione nella tabella di contingenza e la sua area rappresenta il peso relativo della coppia di modalità di X e Y all'interno della distribuzione congiunta. In R, se si associa table(X, Y) alla variabile Z, la rappresentazione tramite rettangoli di una tabella di contingenza si può realizzare con la funzione plot(Z). Ad esempio, riconsiderando l'esempio del colore e del tipo di capelli di un campione di 15 individui, le seguenti linee di codice

```
> colore<-c("biondi","castani","neri","rossi","biondi","castani",
+ "neri","castani","biondi","biondi","castani","castani",
+ "castani","neri","rossi")
>
> tipo<-c("lisci","lisci","lisci","lisci","ondulati","ondulati",
+ "ondulati","ondulati","ondulati","ondulati","ondulati",
+ "ricci","ricci","lisci","ricci")
>
> table(colore,tipo)
          tipo
colore     lisci ondulati ricci
  biondi       1         3     0
  castani      1         3     2
  neri         2         1     0
  rossi        1         0     1
> capelli<-table(colore,tipo)
>
> plot(capelli,col="orange")
```

producono il grafico illustrato in Figura 2.23. Si nota che sono presenti soltanto nove rettangoli pieni e tre rettangoli vuoti corrispondenti alle coppie la cui frequenza è nulla.

È preferibile rappresentare la tabella di contingenza mediante un *grafico a barre sovrapposte*. Il numero di barre è pari al numero delle modalità delle colonne della tabella di contingenza. Inoltre, all'interno di ciascuna barra sono rappresentate una sopra l'altra in altezza le frequenze di ciascuna modalità delle righe della tabella di contingenza. Sull'asse delle ordinate sono indicate le frequenze (assolute o relative). In R tale grafico può essere realizzato tramite la funzione barplot(Z). Riconsiderando l'esempio precedente, le linee di codice

```
> capelli<-table(colore,tipo)
> barplot(capelli,main="Frequenza assoluta congiunta",
+ legend=c("biondi","castani","neri","rossi"),
+ col=c("yellow","brown","black","red"))
```

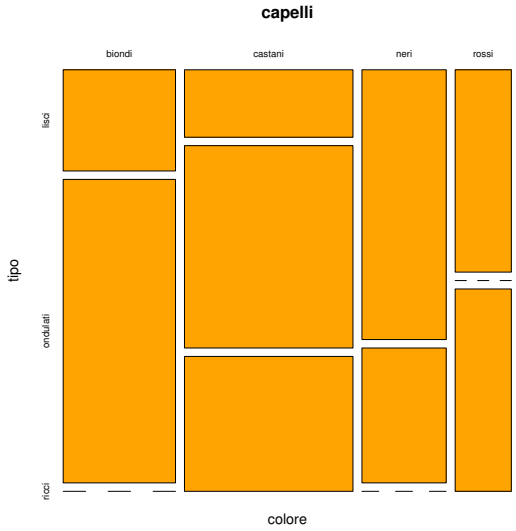

Figura 2.23: Grafico con rettangoli della distribuzione di frequenza assoluta congiunta della tabella di contingenza sul colore e tipo di capelli.

che utilizza la funzione barplot(), produce il grafico illustrato alla sinistra di Figura 2.24 in cui abbiamo inserito un titolo e una legenda. Inoltre, le seguenti linee di codice,

```
> capelliNew<-table(tipo,colore)
> barplot(capelliNew,main="Frequenza assoluta congiunta",
+ legend=c("lisci","ondulati","ricci"),
+ col=c("red","green","blue"))
```

in cui sono state invertite le righe e le colonne della tabella di contingenza, producono il grafico illustrato alla destra di Figura 2.24. Si nota che sull'asse delle ordinate di entrambi i grafici a barre sovrapposte di Figura 2.24 sono indicate le frequenze assolute congiunte.

→ **Grafici per le frequenze assolute marginali**

Per ottenere il grafico della *distribuzione marginale* relativa al colore dei capelli basta utilizzare il comando

```
> plot(margin.table(capelli,1),
+ main="Frequenza assoluta marginale del colore dei capelli",
+ col=1:4)
```

che produce il *grafico a bastoncini* illustrato alla sinistra di Figura 2.25 oppure il comando

```
> barplot(margin.table(capelli,1),
+ main="Frequenza assoluta marginale del colore dei capelli",
```

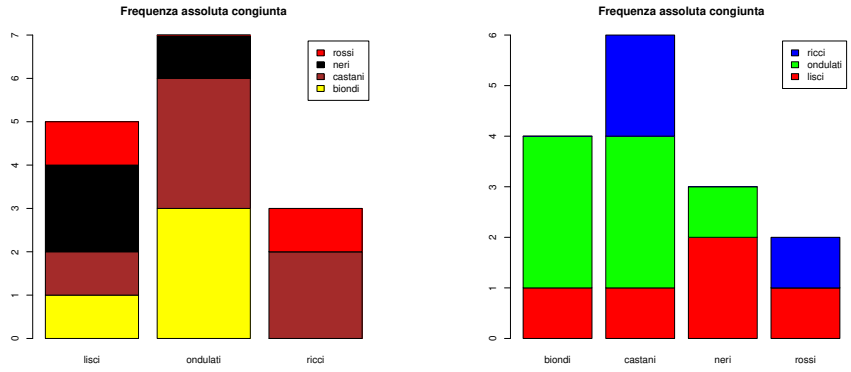

Figura 2.24: Grafici a barre sovrapposte della distribuzione di frequenza assoluta congiunta della tabella di contingenza sul colore e tipo di capelli.

```
+ col=1:4)
```

che produce il *grafico a barre* illustrato alla destra di Figura 2.25.

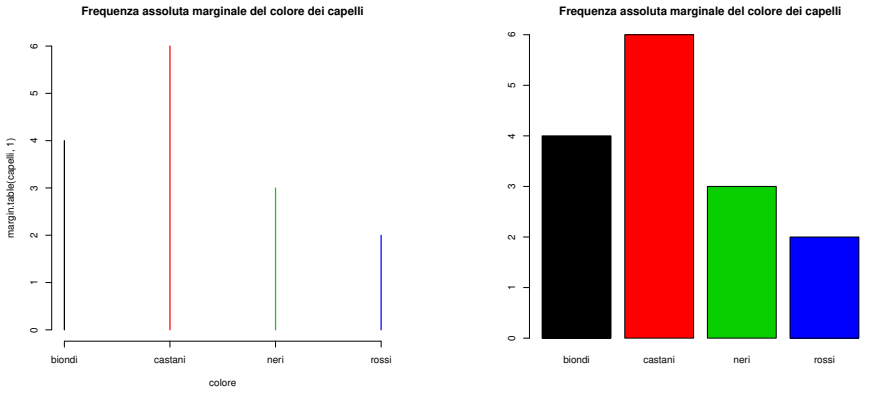

Figura 2.25: Distribuzione di frequenza marginale sul colore dei capelli della tabella di contingenza.

Analogamente, per ottenere il grafico della *distribuzione marginale* relativa al tipo dei capelli basta utilizzare il comando

```
> plot(margin.table(capelli,2),
+ main="Frequenza assoluta marginale del tipo dei capelli",
+ col=1:3)
```

che produce il *grafico a bastoncini* illustrato alla sinistra di Figura 2.26 oppure il comando

```
> barplot(margin.table(capelli,2),
+ main="Frequenza assoluta marginale del tipo di capelli",
+ col=1:3)
```

che produce il *grafico a barre* illustrato alla destra di Figura 2.26.

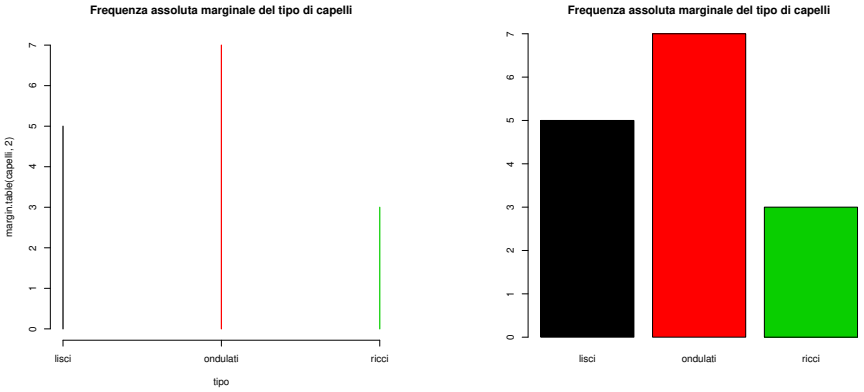

Figura 2.26: Distribuzione di frequenza marginale sul tipo di capelli della tabella di contingenza.

→ **Grafici per le frequenze relative congiunte**

Consideriamo ora le distribuzioni le *frequenze relative*. La rappresentazione grafica della distribuzione delle *frequenze relative congiunte* conduce a *grafici a barre sovrapposte* simili a quelli illustrati in Figura 2.24. Infatti, utilizzando le linee di codice:

```
> capelli<-table(colore,tipo)
> freqCapelli<-prop.table(capelli)
> barplot(freqCapelli,main="Frequenza relativa congiunta",
+ legend=c("biondi","castani","neri","rossi"),
+ col=c("yellow","brown","black","red"))
```

si giunge al grafico a barre sovrapposte alla sinistra di Figura 2.27, mentre utilizzando le seguenti linee di codice:

```
> capelliNew<-table(tipo,colore)
> freqCapelliNew<-prop.table(capelliNew)
> barplot(freqCapelliNew,main="Frequenza relativa congiunta",
+ legend=c("lisci","ondulati","ricci"),
+ col=c("red","green","blue"))
```

si giunge al grafico a barre sovrapposte alla sinistra di Figura 2.27. I due grafici sono ovviamente equivalenti. Si nota che sull'asse delle ordinate di entrambi i grafici a barre sovrapposte sono indicate le frequenze relative congiunte.

→ **Grafici per le frequenze relative marginali**

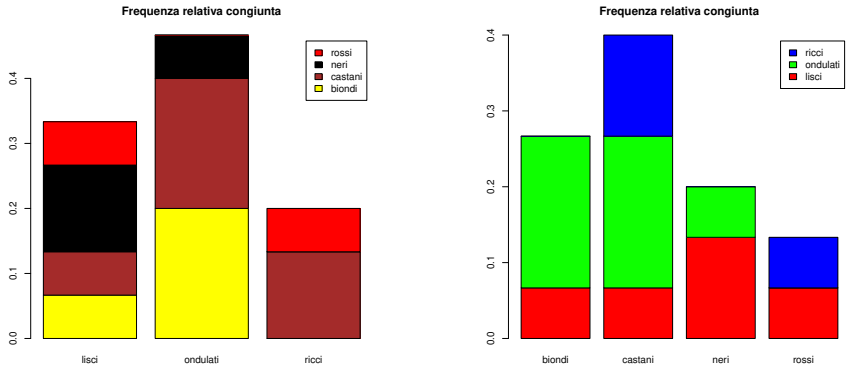

Figura 2.27: Grafici a barre sovrapposte della distribuzione di frequenza relativa congiunta della tabella di contingenza sul colore e tipo di capelli.

È anche possibile rappresentare graficamente le *frequenze relative marginali*. Utilizzando nuovamente la funzione barplot(), si ottiene la distribuzione delle frequenze relative marginali del colore illustrata sulla destra di Figura 2.28 tramite le linee di codice:

```
> capelli<-table(colore,tipo)
> freqCapelli<-prop.table(capelli)
> barplot(margin.table(freqCapelli,1),
+ main="Frequenza relativa marginale del colore dei capelli",
+ col=1:4)
```

e la distribuzione delle frequenze relative marginali del tipo di capelli sulla sinistra di Figura 2.28 tramite le linee di codice:

```
> capelli<-table(colore,tipo)
> freqCapelli<-prop.table(capelli)
> barplot(margin.table(freqCapelli,2),
+ main="Frequenza relativa marginale del tipo di capelli",
+ col=1:3)
```

→ Grafici per le frequenze relative condizionate

È anche possibile rappresentare graficamente le *frequenze relative condizionate* utilizzando nuovamente la funzione barplot(). Infatti, sulla sinistra di Figura 2.29 è rappresentata la distribuzione delle frequenze relative condizionate del colore fissato il tipo utilizzando un grafico a barre appaiate ottenuto tramite le linee di codice

```
> capelli<-table(colore,tipo)
> barplot(prop.table(capelli,2),beside=TRUE,
+ main="Frequenza relativa condizionata f(colore|tipo)",
+ legend=c("biondi","castani","neri","rossi"),
+ col=c("yellow","brown","black","red"))
```

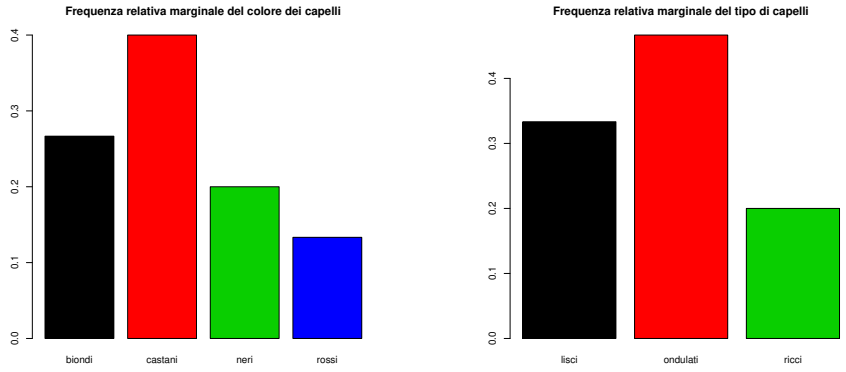

Figura 2.28: Grafico della distribuzione di frequenza relativa marginale del colore (alla sinistra) e del tipo (alla destra) dei capelli della tabella di contingenza.

mentre sulla destra di Figura 2.29 è rappresentata la distribuzione delle frequenze relative condizionate del tipo fissato il colore mediante un grafico a barre sovrapposte ottenuto tramite le linee di codice

```
> capelliNew<-table(tipo,colore)
> barplot(prop.table(capelliNew,2),beside=TRUE,
+ main="Frequenza relativa condizionata f(tipo|colore)",
+ legend=c("lisci","ondulati","ricci"),
+ col=c("red","green","blue"))
```

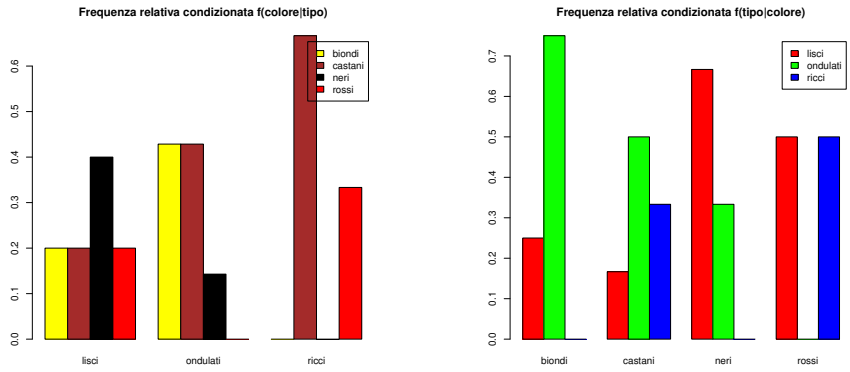

Figura 2.29: Rappresentazioni grafiche delle distribuzioni di frequenza relative condizionate della tabella di contingenza.

Per ottenere la Figura 2.29 abbiamo utilizzato l'opzione beside = TRUE che

permette di ottenere al posto di un grafico a barre sovrapposte *un grafico a barre appaiate*, ognuna delle quali indica la frequenza relativa condizionata dell'ordinata nota l'ascissa. Si nota che la somma delle frequenze relative condizionate relative alle ordinate è unitaria per ogni fissata modalità delle ascisse.

Capitolo 3

Rappresentazioni grafiche dei dati

3.1 Introduzione

Ogni volta che si richiede l'esecuzione di un comando grafico si attiva automaticamente una *finestra grafica* (device) su cui vengono indirizzati i risultati dei diversi comandi grafici eseguiti dalla linea di comando. In R, per default, ogni nuovo grafico creato cancellerà quello precedentemente realizzato occupando la stessa periferica attiva. Le dimensioni di ogni grafico prodotto possono essere modificate con il mouse trascinando i lati della relativa finestra rendendo così una figura più schiacciata o più allungata.

È possibile richiedere che i grafici prodotti in una sessione di R vengano inseriti in opportuni file che utilizzano un particolare formato grafico nella directory corrente. La lista dei formati grafici possibili può essere visualizzata con il comando ?device. I file creati restano attivi finché non si chiude la sessione di R oppure possono essere chiusi anticipatamente attraverso la funzione dev.off().

Per procedere a salvare un'immagine presente nella finestra grafica è necessario selezionare la finestra grafica in maniera tale da attivare alcuni menù e comandi grafici tra i quali è possibile scegliere se inviare l'immagine in stampa oppure se salvare l'immagine nel formato desiderato.

Le funzioni grafiche possono essere divise in tre gruppi:

- *funzioni ad alto livello* che creano un nuovo grafico nella finestra grafica e agiscono in modo automatico sulla base dei soli parametri indicati negli argomenti;
- *funzioni a basso livello* che aggiungono particolari ad un grafico già esistente;
- *funzioni per grafici interattivi* che consentono di aggiungere o estrarre interattivamente informazioni da un grafico già esistente.

Il sistema grafico di R mette a disposizione dell'utente varie funzioni grafiche in grado di produrre differenti grafici in base alla tipologia dei dati forniti. La funzione polimorfica di alto livello più utilizzata è plot(). Nella Tabella 3.1 sono elencate le varie varianti della funzione plot().

Tabella 3.1: Le funzioni plot()

plot(x)	Se x è un *vettore*, il grafico illustra l'andamento dei valori assunti dal vettore rispetto ai rispettivi indici; se x è un vettore di numeri complessi, il grafico illustra l'andamento delle parti immaginarie rispetto alle parti reali.
plot(f)	Se f è di tipo *fattore*, viene prodotto un grafico a barre che illustra l'andamento delle frequenze dei valori contenuti nella variabile.
plot(x, y)	Se x e y sono *vettori*, il grafico produce uno scatterplot (nuvola di punti) di y rispetto a x.
plot(f, y)	Se f è un *fattore* e y un *vettore* numerico, la funzione produce un boxplot di y per ogni livello di f.
plot(m)	Se m è una matrice a due colonne produce uno scatterplot di una colonna rispetto all'altra.
plot(df)	Se df è un *data frame*, viene prodotto un grafico distribuzionale delle variabili del data frame.
plot(~expr)	Se expr è una lista di nomi di oggetti separati da +, viene prodotto un grafico distribuzionale degli oggetti indicati.
plot(y~expr)	Se y è un oggetto e expr è una lista di nomi di oggetti separati da +, viene prodotto un grafico di y rispetto a ciascun oggetto indicato in expr.

Tabella 3.2: Altre funzioni per rappresentare dati multivariati

pairs(x)	Nel caso x sia una *matrice* o un *data frame*, viene prodotta una matrice di scatterplot accoppiati delle variabili definite dalle colonne di x, ossia la funzione disegna tutti i possibili grafici bidimensionali ottenibili combinando tra loro le colonne presenti in x.
coplot(a~b|d)	In questo caso a e b sono due vettori e d è un vettore oppure un fattore. Se d è un fattore, la funzione produce un numero di scatterplot di a rispetto a b per ogni fissato livello del fattore d. Se d è un vettore numerico, allora i suoi valori sono suddivisi in un numero di intervalli e per ogni intervallo si ha un grafico di a rispetto a b per i valori di d nell'intervallo.

Le funzioni pairs(x) e coplot(a~b|d), considerate in Tabella 3.2 sono molto utili per rappresentare dati multivariati. Infatti la funzione pairs(x) genera una pluralità di grafici costituiti da tutte le "nuvole di punti" che corrispondono alla combinazione delle variabili presenti nella struttura. Utilizzando la funzione coplot() è possibile invece generare grafici bivariati, suddivisi anche in base ai valori di altre variabili che costituiscono i fattori condizionanti.

Altre funzioni molto utili per ottenere differenti grafici in base alla tipologia dei dati forniti sono illustrate in Tabella 3.3

Tabella 3.3: Altre funzioni frequentemente utilizzate per i grafici

barplot(x)	Produce un grafico a barre dei valori presenti nel vettore x sulla base delle frequenze.
boxplot(x)	Produce un boxplot dei valori presenti nel vettore x sulla base delle frequenze.
boxplot(split(x, f))	Se x è un vettore e f un fattore, la funzione realizza nella stessa immagine dei boxplot relativi alla variabile x per ciascuna delle modalità indicate nel fattore f.
pie(x)	Fornisce un grafico a torta dei valori presenti in x sulla base delle frequenze.
hist(x)	Produce l'istogramma sulla base delle frequenze calcolate su x. Il numero di classi viene scelto per default.
hist(x, nclass = n)	Produce l'istogramma sulla base delle frequenze calcolate su x e viene indicato il numero di classi.
hist(x, breaks = ...)	Produce l'istogramma sulla base delle frequenze calcolate su x e sono indicati gli estremi degli intervalli.
qqplot(x, y)	Confronta le distribuzioni empiriche di x e di y, ossia produce un grafico dei quantili dei dati del vettore x rispetto ai quantili dei dati del vettore y.
qqnorm(x)	Produce un grafico quantile-quantile dei dati del vettore x rispetto ad una corrispondente distribuzione normale standard.
qqline(x)	Funzione di basso livello che aggiunge una linea che passa attraverso il primo e il terzo quartile.

In ciascuna delle funzioni elencate nelle Tabelle 3.1, 3.2 e 3.3 si possono utilizzare dei *parametri aggiuntivi* che permettono di gestire e di controllare il processo grafico. Alcuni di tali parametri sono elencati in Tabella 3.4

Il sistema grafico di R mette a disposizione dell'utente anche un'altra serie di *funzioni a basso livello* usate per aggiungere caratteristiche a grafici precedentemente realizzati. Alcune volte inoltre si desidera ottenere delle immagini che non possono essere prodotte utilizzando le funzioni grafiche di alto livello. L'utente può allora progettare nuovi tipi di immagini utilizzando soltanto le funzioni grafiche di basso livello. Le funzioni che realizzano i comandi a basso livello sono anche usate dai progettisti di applicazioni R per realizzare i comandi grafici ad alto livello. Alcuni comandi grafici di basso livello sono elencati nella Tabella 3.5.

Tabella 3.4: Parametri aggiuntivi per la gestione e il controllo grafico

area =	Se si utilizza FALSE non vengono disegnati gli assi e il contorno del grafico
col =	Colore usato per evidenziare le linee, i punti e i rettangoli; si può usare anche un vettore di colori per evidenziare colori multipli.
lty =	Tipo di linea richiesto (0 = tratto bianco, 1 = linea continua, 2 = linea tratteggiata, 3 = linea punteggiata, 4 = linea punto-tratto, 5 = linea con tratto lungo, 6 = linea con doppio tratto).
lwd =	Spessore delle linee.
log = "x"	Indica di porre in scala logaritmica la variabile posta sull'asse delle ascisse
log = "y"	Indica di porre in scala logaritmica la variabile posta sull'asse delle ordinate
log = "xy"	Indica di porre in scala logaritmica entrambe le variabili poste sull'asse delle ascisse e delle ordinate
type = "x"	Il carattere "x" indica il tipo di grafico da visualizzare; con "p" si ha una nuvola di punti individuali, con "l" delle linee connesse, con "b" punti connessi da linee, con "o" punti connessi da linee che passano sopra i punti, con "h" linee verticali dai punti dell'asse delle ascisse, con "s" o "S" una funzione a gradini, con "n" nessun grafico.
xlab = "str"	Stringa da associare all'asse delle ascisse.
ylab = "str"	Stringa da associare all'asse delle ordinate.
main = "str"	Titolo della figura che appare in testa al grafico.
sub = "str"	Sottotitolo della figura a caratteri più piccoli che appare sotto l'asse delle ascisse.
xlim = c(n_1, n_2)	Limite inferiore n_1 e superiore n_2 dell'asse delle ascisse.
ylim = c(n_1, n_2)	Limite inferiore n_1 e superiore n_2 dell'asse delle ordinate.
border =	Permette di indicare il colore da utilizzare per colorare il contorno di figure come rettangoli o poligoni.
bg =	Indica quale colore di sfondo utilizzare nella finestra grafica.
cex =	Il valore associato controlla la grandezza del testo e dei simboli generati. Il valore di default è cex = 1.
mfcol =	Utilizzando un vettore c(nr, nc), dove nr è il numero di righe e nc il numero di colonne, gestisce la forma di suddivisione della finestra grafica. I disegni sono inseriti per colonna.
mfrow =	Gestisce la forma di suddivisione della finestra grafica e i disegni sono inseriti per riga.
pch =	Controlla il tipo di carattere da utilizzare attraverso un valore numerico compreso tra 0 e 20 oppure attraverso alcuni caratteri particolari quali " + ","x"," ∗ ".
font =	Controlla lo stile del testo: normale (1), corsivo (2), grassetto (3), grassetto corsivo (4).
add =	Se add = TRUE il grafico corrente viene sovrapposto ad uno precedentemente disegnato. Per default add = FALSE.
new =	Se new = TRUE il grafico successivo viene sovrapposto a quello precedente. Per default new = FALSE.
las =	fornisce lo stile delle labels dell'asse delle coordinate: 0 per labels parallele all'asse, 2 per labels perpendicolari all'asse.

Tabella 3.5: Alcuni comandi grafici a basso livello

points(x, y)	Aggiunge punti in un grafico nella posizione indicata dalle coordinate presenti nell'argomento.
lines(v)	Aggiunge linee di collegamento al grafico corrente che collegano i punti del vettore.
abline(a, b)	Aggiunge la linea di inclinazione b e intercetta a ad una curva.
abline(h = y)	Aggiunge la linea orizzontale che passa per l'ordinata y.
abline(v = x)	Aggiunge la linea verticale che passa per l'ascissa x.
abline(lm.obj)	Aggiunge la linea di regressione stimata.
segments(x0, y0, x1, y1)	Traccia un segmento tra i punti di coordinate $x0, y0)$ e $(x1, y1)$.
polygon(x, y, ...)	Disegna un poligono che collega tutti i punti le cui ascisse e ordinate sono indicate nei due vettori x e y di uguale ampiezza, in cui il primo e l'ultimo valore devono coincidere per ottenere una curva chiusa.
rect(x, y, z, w, ...)	Disegna un rettangolo tra i due punti di coordinate (x, y) in basso a sinistra e (z, w) in alto a destra.
curve(expr, from, to, add, ...)	Disegna una curva sulla base dell'espressione indicata in expr, nell'intervallo [from, to]. È possibile utllizzare il parametro add posto a TRUE (di default è FALSE) per aggiungere la curva ad un grafico precedentemente realizzato.
arrows(x,y,z,w,angle,code,...)	Aggiunge una freccia che congiunge i due punti (x, y) e (z, w). Se code = 1 la punta della freccia è (z, w), mentre se code = 2 la punta della freccia è posizionata in (x, y). Il parametro angle definisce l'inclinazione della freccia.

Il sistema grafico di R consente l'inserimento di testi sia lungo i margini, sia all'interno del grafico. Le scritte possono contenere risultati ottenuti da precedenti elaborazioni e anche formule complesse. In Tabella 3.6 sono elencati alcuni comandi a basso livello per l'inserimento di testi. In particolare, la funzione mtext() è simile ad axis(), ma con un utilizzo più ampio; infatti tale funzione può essere utilizzata oltre che per inserire i nomi sugli assi, per disegnare etichette in una specifica posizione ed anche per aggiungere un titolo. Anche la funzione text() consente di inserire del testo all'interno di un grafico, ma a differenza delle funzioni mtext() e axis() richiede le coordinate in cui il testo deve essere inserito, mentre in labels deve essere inserito il testo da visualizzare nel grafico. Un'altra funzione particolarmente utile per aggiungere una legenda all'interno di un grafico è legend() con la quale le componenti dei vettori vengono affiancate nel momento della creazione del testo.

Tabella 3.6: Comandi grafici a basso livello per l'inserimento di testi

text(x, y, labels = ...)	Aggiunge il testo indicato nel parametro labels nella posizione indicata dalle coordinate (x, y).
title(main, sub)	Aggiunge il titolo (sopra) e anche un sottotitolo (sotto) al grafico.
axis(side, ...)	Aggiunge un asse nel lato indicato dall'intero side: 1 (basso), 2 (sinistra), 3 (alto), 4 (destra); il vettore, se presente, indica le posizioni dei marcatori sull'asse.
axes	Se FALSE non disegna gli assi e il contorno del grafico (che possono essere disegnati successivamente con il comando axis).
mtext(testo, side, line, ...)	Aggiunge il testo indicato sul margine scelto con il parametro side (1,2,3,4); il parametro line indica lo scostamento (valore intero partendo da 0) dalla posizione in cui sono tracciati gli assi.
legend(x, y, legend, ...)	Aggiunge una legenda al disegno nel punto (x, y). Occorre fornire un vettore v della stessa lunghezza di legend e impostare i parametri di tale vettore.
paste(x, y)	Consente di concatenare vettori dopo averli convertiti in stringhe di caratteri.
format(x, digits =)	Permette di formattare i numeri utilizzando un numero specificato di cifre.

Il sistema offre anche la possibilità di aggiungere facilmente ad un grafico sia simboli matematici sia delle formule inserendo come parametro all'interno della funzione expression() un'espressione che utilizzi una sintassi simile a quella di LATEX. Quando si genera il grafico, l'espressione viene tradotta nella corrispondente espressione matematica rispettando lo stile utilizzato nei documenti LATEX. La funzione expression() può essere fornita in ingresso alle funzioni text(), mtext(), axis(), title(). Per avere un'idea degli operatori e funzioni necessarie ad ottenere questo risultato è sufficiente consultare l'help tramite il

comando ?plotmath. È anche possibile visionare alcuni esempi con il comando demo(plotmath)

Per quanto riguarda i colori è possibile utilizzare le funzioni a basso livello indicate in Tabella 3.7. Per default il sistema R mette a disposizione 8 colori (black, red, green, blue, cyan, magenta, yellow, gray). Nel caso si desideri utilizzare uno dei 657 colori definiti da colors(), è possibile passare al parametro col il corrispondente nome nella posizione indicata nel vettore dei colori prodotto dalla funzione colors(). La funzione rainbow() (arcobaleno) implementa una tavolozza di colori ed è utilizzata in molti pacchetti software; è spesso criticata per i suoi numerosi problemi percettivi, poiché i colori, molto appariscenti e sbilanciati, possono condurre a varie illusioni ottiche. Pertanto, si consiglia di utilizzare la tavolozza fornita da hcl.colors(). È possibile visionare alcuni esempi di colori con il comando demo(colors).

Per i colori è anche possibile utilizzare il formato RGB. Infatti, la funzione rgb() ritorna la rappresentazione in esadecimale dei colori attraverso i tre colori primari: rosso (red), verde (green) e blu (blue). Ad esempio,

```
> rgb(255, 0, 0, maxColorValue = 255)
[1] "#FF0000"
> rgb(0, 255, 0, maxColorValue = 255)
[1] "#00FF00"
```

forniscono il colore rosso e il colore verde, rispettivamente. Pertanto, nel codice di una figura si può inserire col = "#FF0000" per il rosso e col = "#00FF00" per il verde.

Per alcune funzioni, come ad esempio rect() e polygon(), è possibile colorare opportunamente il contorno o l'interno utilizzando un appropriato colore, oppure un opportuno tratteggio. Il colore all'interno del poligono o del rettangolo può essere scelto mediante il parametro col =, mentre il parametro border = fissa il contorno. Per colorare l'interno di figure chiuse è possibile utilizzare oltre a un colore uniformemente distribuito, anche un insieme di linee parallele colorate (col =), ugualmente distanziate (density =) e opportunamente inclinate (angle =).

Tabella 3.7: Comandi grafici a basso livello per i colori

colors()	Ritorna all'interno di un vettore i nomi di 657 colori predefiniti che possono essere usati dal parametro col nelle diverse funzioni grafiche.
palette(val)	consente di vedere o cambiare la palette di 8 colori ("black", "red", "green3", "blue", "cyan", "magenta", "yellow", "gray") che potranno essere direttamente usati dal parametro col disponibile nelle funzioni grafiche.

Alcune funzioni per la grafica 3D sono riportate in Tabella 3.8. Ognuna di queste funzioni è dotata di diversi parametri che permettono diverse modalità di visualizzazione.

Le principali funzioni per interagire con un grafico sono riportate in Tabella 3.9.

Tabella 3.8: Funzioni per la grafica 3D

image()	Permette di visualizzare grafici 3D, e anche 2D, utilizzando diversi toni di colore per le altezze.
persp()	Permette di visualizzare superfici.
contour()	Rappresenta una superfice 3D tramite curve di livello.

Tabella 3.9: Funzioni per interagire con un grafico

locator(n, type)	Aspetta che l'utente selezioni le posizioni del grafico usando il tasto sinistro del mouse e continua finché non si selezionano n punti. L'argomento n permette di disegnare i punti selezionati e ammette uno tra i seguenti caratteri "b", "p", "l", "o". L'elaborazione può essere interrotta digitando il tasto destro del mouse e scegliendo il comando "Stop".
identify(x, y, labels)	Permette che l'utente selezioni alcuni punti definiti da (x, y) usando il tasto sinistro del mouse, disegnando vicino le corrispondenti label.

Per poter configurare le caratteristiche di un grafico si utilizza la funzione par() passando come argomento uno o più parametri. Se si desidera, ad esempio, suddividere la finestra grafica in nr × nc parti in modo da poter rappresentare contemporaneamente nc grafici su ognuna delle nr righe, il comando da utilizzare è il seguente

$$> \mathsf{par}(\mathsf{mfrow} = \mathsf{c}(\mathsf{nr}, \mathsf{nc}))$$

se l'immissione delle immagini avviene per riga, mentre è

$$> \mathsf{par}(\mathsf{mfcol} = \mathsf{c}(\mathsf{nr}, \mathsf{nc}))$$

se l'immissione delle immagini avviene per colonna. Ad esempio, il comando par(mfcol = c(3, 2)) partiziona la finestra grafica in modo da formare una matrice regolare di rettangoli avente tre righe e due colonne, nei cui riquadri verranno collocate per colonna le immagini successive prodotte. È tuttavia possibile creare suddivisioni della finestra grafica con riquadri non regolari sulla base di particolari esigenze dell'utente. Il comando par() permette di controllare varie caratteristiche tra cui il tipo di assi da utilizzare, i font, il colore delle etichette degli assi, il titolo del grafico, l'area su cui visualizzare il grafico, ... Per avere un elenco completo dei parametri grafici che è possibile configurare con il comando par() è consigliabile consultare l'help in linea con il comando ?par. Uno dei parametri più interessanti disponibili nella funzione par() è new = TRUE che consente di sovrapporre tra loro le immagini create da comandi grafici di alto livello ed evita la cancellazione del precedente grafico.

3.2 Istogrammi

Gli istogrammi, che si utilizzano per *variabili quantitative*, sono una particolare rappresentazione grafica di una distribuzione di frequenza in classi. Consideriamo un campione $(x_1, x_2, \ldots, x_n)$ costituito da n osservazioni, che suddividiamo in classi; ogni osservazione deve cadere in una ed una sola classe (o intervallo). Gli istogrammi sono quindi una particolare rappresentazione grafica ottenuta mediante rettangoli adiacenti aventi per basi segmenti i cui estremi corrispondono agli estremi delle classi. L'asse delle ascisse di un istogramma è quindi sempre dotato di un'unità di misura. Fissate le basi, le altezze debbono essere tali che l'area di ogni rettangolo risultante sia uguale alla frequenza (relativa o assoluta) della classe stessa.

Se si utilizzano le *frequenze assolute delle classi*, l'area di ogni rettangolo è uguale alla frequenza assoluta della classe e l'area totale dei rettangoli è uguale all'ampiezza del campione. Quindi, l'area del rettangolo i-esimo è uguale a $n_i = b_i \times h_i$, dove n_i è il numero di valori che cadono nella classe i-esima (frequenza assoluta della classe i-esima), b_i è la base e h_i è l'altezza della classe i-esima. Se si sceglie $b_i = 1$ per ogni classe, le altezze h_i delle classi dell'istogramma corrispondono alle frequenze assolute n_i delle classi.

La funzione che realizza in R un istogramma è hist(). È possibile lasciare scegliere ad R il numero di classi da utilizzare, oppure suggerire ad R il numero di classi da utilizzare mediante il parametro breaks. Per realizzare un istogramma in base alle frequenze assolute, occorre impostare nella funzione hist() il parametro freq = TRUE (di default).

Ad esempio, riferendoci al vettore "voti" dei 30 studenti precedentemente considerato, le seguenti linee di codice

```
> voti<-c(18,19,20,30,29,28,21,22,23,27,26,25,24,25,
+ 26,24,23,22,27,28,21,24,25,25,27,19,21,28,29,28)
> table(voti)
voti
18 19 20 21 22 23 24 25 26 27 28 29 30
 1  2  1  3  2  2  3  4  2  3  4  2  1
>
> hist(voti,freq=TRUE,main="Istogramma dei voti",
+ ylab="Frequenza assoluta delle classi")
```

producono l'istogramma rappresentato in Figura 3.1 in cui il numero di classi è stato scelto automaticamente da R. Abbiamo utilizzato il parametro main per definire il titolo del grafico. Si nota che l'area totale è uguale all'ampiezza del campione. Scegliendo basi unitarie, si nota che l'area totale è uguale all'ampiezza 30 del campione.

La funzione hist() è in grado di generare oltre al grafico, anche una serie di informazioni sulla sua natura che possono essere salvate in un variabile h di tipo list. Queste informazioni possono essere visualizzate utilizzando la funzione str(h) applicata alla variabile generata dalla funzione hist(). Riferendosi all'esempio precedente, si ottiene

```
> h<-hist(voti,freq=TRUE,main="Istogramma dei voti",
```

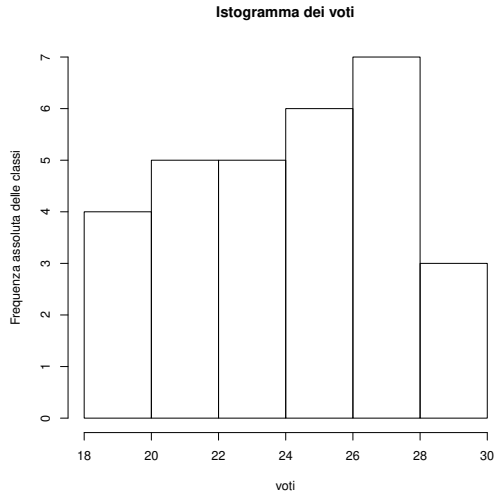

Figura 3.1: Istogramma relativo al vettore numerico "voti" in base alle frequenze assolute delle classi.

```
+ ylab="Frequenza assoluta delle classi")
> str(h)
List of 6
 $ breaks  : num [1:7] 18 20 22 24 26 28 30
 $ counts  : int [1:6] 4 5 5 6 7 3
 $ density : num [1:6] 0.0667 0.0833 0.0833 0.1 0.1167 ...
 $ mids    : num [1:6] 19 21 23 25 27 29
 $ xname   : chr "voti"
 $ equidist: logi TRUE
 - attr(*, "class")= chr "histogram"
```

Come si vede la funzione str(h) fornisce i punti di suddivisione in classi (breaks), le frequenze assolute delle classi (counts), la densità delle classi (density) e i punti centrali delle classi (mids) come mostra il seguente codice:

```
> h$breaks
[1] 18 20 22 24 26 28 30
>
> h$counts
[1] 4 5 5 6 7 3
>
> h$density
[1] 0.06666667 0.08333333 0.08333333 0.10000000 0.11666667
    0.05000000
>
> h$mids
[1] 19 21 23 25 27 29
```

Operando sui risultati contenuti nella variabile h generata dalla funzione hist(), è possibile calcolare alcune statistiche utilizzando le funzioni h$breaks, h$counts,

h$density e h$mids.

In Figura 3.1, la suddivisione in classi scelta automaticamente da R è la seguente: $[18, 20]$, $(20, 22]$, $(22, 24]$, $(24, 26]$, $(26, 28]$, $(28, 30]$. Infatti, dei 30 voti, 4 cadono nella prima classe, 5 nella seconda classe, 6 nella quarta classe, 7 nella quinta classe e 3 nella sesta classe (frequenza assoluta delle classi).

Le *frequenze relative* associate alle sei classi possono essere ottenute moltiplicando gli elementi del vettore h$density per 2 (ampiezza effettiva di ogni classe) e sono:

```
> f<-2*h$density
> f
[1] 0.1333333 0.1666667 0.1666667 0.2000000 0.2333333 0.1000000
> sum(f)
[1] 1
```

ossia $f_1 = 0.133$, $f_2 = 0.167$, $f_3 = 0.167$, $f_4 = 0.200$, $f_5 = 0.233$ e $f_6 = 0.100$. Si nota che la somma delle frequenze relative associate alle classi è unitaria.

Supponiamo ora di fissare le classi dell'istogramma con breaks, considerando le seguenti linee di codice

```
> classi<- 17+1*(0:13)
> hist(voti,freq=TRUE,main="Istogramma dei voti", breaks=classi,
+ ylab="Frequenza assoluta delle classi")
```

che produce invece l'istogramma visualizzato in Figura 3.2. Gli intervalli unitari sono aperti a sinistra e chiusi a destra, ossia del tipo $(k, k + 1]$ dove $k = 17, 18, \ldots, 29$. Si nota nuovamente che l'area totale è uguale all'ampiezza del campione. Occorre sottolineare che troppe classi portano ad un istogramma

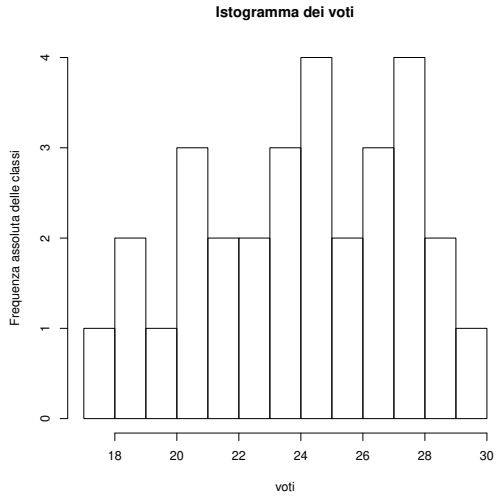

Figura 3.2: Istogramma relativo al vettore numerico "voti" fissando il numero di classi e in base alle frequenze assolute delle classi.

simile ad un grafico a barre e troppo poche classi rendono l'istogramma troppo piatto. Inoltre, se si desidera l'istogramma delle frequenze assolute occorre scegliere classi di uguale ampiezza.

Se si utilizzano le frequenze relative delle classi l'area di ogni rettangolo è uguale alla frequenza relativa della classe stessa e l'area totale è uguale all'unità. Quindi, l'area del rettangolo i-esimo è uguale a $f_i = n_i/n = b_i \times h_i$, dove f_i è la frequenza relativa dei valori che della classe i-esima, b_i è l'ampiezza e h_i l'altezza della classe i-esima. In questo caso l'altezza di ogni rettangolo esprime la *densità di frequenza* associata alla classe considerata.

Per realizzare un istogramma in base alle frequenze relative, occorre impostare nella funzione hist() il parametro freq = FALSE.

Riferendoci al campione di voti di 30 studenti, la seguente linea di codice:

```
> hist(voti,freq=FALSE, main="Istogramma dei voti",
+ ylab="Densita' di frequenza delle classi",col=rainbow(6))
```

produce l'istogramma rappresentato in Figura 3.3 in base alle densità di frequenza delle classi. In tal caso il numero di classi è stato scelto automaticamente da R e l'area totale è unitaria.

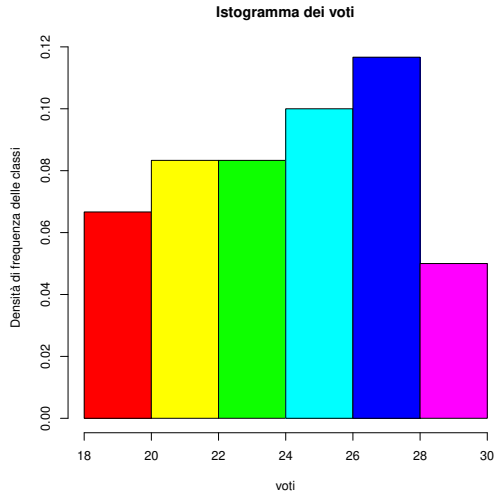

Figura 3.3: Istogramma relativo al vettore numerico "voti" fissando il numero di classi in base alle densità di frequenza delle classi.

Osservando il grafico in Figura 3.3, si nota che h$density fornisce le altezze dei rettangoli associate alle 6 classi.

3.3 Kernel density plot

Gli istogrammi sono un importante strumento per la rappresentazione di una distribuzione di frequenza in classi per variabili quantitative univariate. La scelta degli intervalli delle classi è cruciale per l'aspetto finale del grafico dell'istogramma. Un modo più elegante per affrontare questo problema consiste nello smussare l'istogramma utilizzando grafici che utilizzano la *stima di densità basata su kernel*. Con tale metodo, invece di raccogliere le osservazioni in barre, come negli istogrammi, si traccia una curva continua determinata da un fattore K, detto *kernel*, e da un parametro h, detto *ampiezza della banda* (bandwidth).

Sia $(x_1, x_2, \ldots, x_n)$ un campione costituito da n osservazioni di una variabile quantitativa la cui densità di frequenza $f(x)$ non è nota in ogni punto x. Siamo interessati a stimare la forma (shape) di questa funzione $f(x)$ in base al campione di osservazioni. Un grafico della densità basata sul kernel può essere realizzato utilizzando la seguente funzione stimata in base al campione

$$\widehat{f}_h(x) = \frac{1}{n\,h} \sum_{i=1}^{n} K\Big(\frac{x - x_i}{h}\Big), \qquad x \in \mathbb{R},$$

dove n è l'ampiezza del campione, $K(x)$ è il kernel, ossia una funzione densità di probabilità (non negativa) con media nulla e $h > 0$ è un parametro di smoothing, detto *bandwidth*. La formula precedente assicura che la funzione densità stimata sia integrata ad uno. Infatti, si ha

$$\int_{-\infty}^{+\infty} \widehat{f}_h(x)\, dx = \frac{1}{n\,h} \sum_{i=1}^{n} \int_{-\infty}^{+\infty} K\Big(\frac{x - x_i}{h}\Big)\, dx = \frac{1}{n} \sum_{i=1}^{n} \int_{-\infty}^{+\infty} K(y)\, dy = 1.$$

Mostriamo che il valore medio utilizzando la densità stimata coincide con la media campionaria. Infatti, si ha

$$\begin{aligned}\int_{-\infty}^{+\infty} x\, \widehat{f}_h(x)\, dx &= \frac{1}{n\,h} \sum_{i=1}^{n} \int_{-\infty}^{+\infty} x\, K\Big(\frac{x - x_i}{h}\Big)\, dx = \frac{1}{n} \sum_{i=1}^{n} \int_{-\infty}^{+\infty} (x_i + h\,y)\, K(y)\, dy \\ &= \frac{1}{n} \sum_{i=1}^{n} \Big[x_i \int_{-\infty}^{+\infty} K(y)\, dy + h \int_{-\infty}^{+\infty} y\, K(y)\, dy \Big] = \frac{1}{n} \sum_{i=1}^{n} x_i,\end{aligned}$$

essendo per ipotesi $K(x)$ una funzione densità di probabilità con media nulla.

La scelta del kernel $K(x)$ può influenzare l'aspetto generale del grafico. Inoltre, la scelta del parametro h è uno degli aspetti più delicati del metodo: un valore troppo vicino a zero rende la stima irregolare e con varianza troppo elevata; invece, un valore troppo elevato comporta problemi di distorsione (un campione distorto fornisce, in generale, una stima falsata delle caratteristiche della popolazione oggetto dell'indagine statistica).

R mette a disposizione vari tipi di kernel: "gaussian", "rectangular", "triangular", "epanechnikov", "biweight","cosine"and "optcosine". Il kernel che si basa sulla densità normale standard è chiamato "gaussian"" ed è l'impostazione

di default in R; è possibile produrre un grafico della densità basata sul kernel gaussiano combinando soltanto ii comandi plot e density. La scelta del kernel dipende dal campione, ma la scelta di default è spesso quella da preferire. Ad esempio per un kernel rettangolare si procede nel seguente modo:

```
# Kernel density estimation
d <- density(x, kernel = "rectangular")
# Kernel density plot
plot(d, lwd = 2, main = "Rectangular kernel")
```

dove x è il vettore dei dati e il parametro lwd definisce lo spessore delle curve.

I differenti kernel sono i seguenti:

gaussian: $K(x) = \frac{1}{\sqrt{2\,\pi}}\,\exp\{-\frac{x^2}{2}\}, \quad x \in \mathbb{R} \qquad \sigma^2 = 1$

rectangular (uniform): $K(x) = \begin{cases} 1/2, & -1 \le x \le 1, \\ 0, & \text{altrimenti}, \end{cases} \qquad \sigma^2 = \frac{1}{3}$

triangular: $K(x) = \begin{cases} 1 - |x|, & -1 \le x \le 1, \\ 0, & \text{altrimenti.} \end{cases} \qquad \sigma^2 = 1/6$

epanechnikov (parabolic): $K(x) = \begin{cases} \frac{3}{4}(1 - x^2), & -1 \le x \le 1, \\ 0, & \text{altrimenti.} \end{cases} \qquad \sigma^2 = \frac{1}{5}$

biweigh (quartic): $K(x) = \begin{cases} \frac{15}{16}(1 - x^2)^2, & -1 \le x \le 1, \\ 0, & \text{altrimenti.} \end{cases} \qquad \sigma^2 = \frac{1}{7}$

cosine: $K(x) = \frac{\pi}{4}\cos\left(\frac{\pi}{2}x\right), \quad -1 \le x \le 1, \qquad \sigma^2 = 1 - \frac{8}{\pi^2}$

optcosine: $K(x) = \frac{1}{2}[1 + \cos(\pi\,x)], \quad -1 \le x \le 1, \qquad \sigma^2 = \frac{1}{3} - \frac{2}{\pi^2}$

tutte caratterizzate da valore medio nullo e varianza $\sigma^2 = \int_{-\infty}^{+\infty} x^2\,K(x)\,dx$ (coincidente con il momento del secondo ordine). Ad esempio, consideriamo un vettore di altezze di 30 individui e rappresentiamo la stima della densità per i vari kernel. Il codice seguente

```
> altezze<- c(1.81,1.50,1.90,1.70,1.55,1.90,1.95,1.55, 1.60,1.62,
+ 1.63,1.80,1.91,1.70,1.58,1.72,1.64,1.67,1.58,1.69,
+ 1.61,1.74,1.67,1.56,1.60,1.71,1.65,1.63,1.80,1.90)
> length(altezze)
[1] 30
> par(mfrow=c(2,2))
> hist(altezze,freq=FALSE, main="Istogramma delle altezze",
+ col= hcl.colors(5))
>
> d1<-density(altezze,kernel = "gaussian")
> plot(d1, lwd = 2, main = "Gaussian kernel",col="red")
>
> d2<-density(altezze,kernel = "rectangular")
> plot(d2, lwd = 2, main = "Rectangular kernel",col="blue")
>
> d3<-density(altezze,kernel = "triangular")
> plot(d3, lwd = 2, main = "Triangular kernel",col="magenta")
```

permette di ottenere la Figura 3.4 in cui si confronta l'istogramma delle frequenze relative con le stime delle densità basate su kernel gaussiano, rettangolare e triangolare. Nella Figura 3.5 si confrontano invece le stime delle densità ottenute

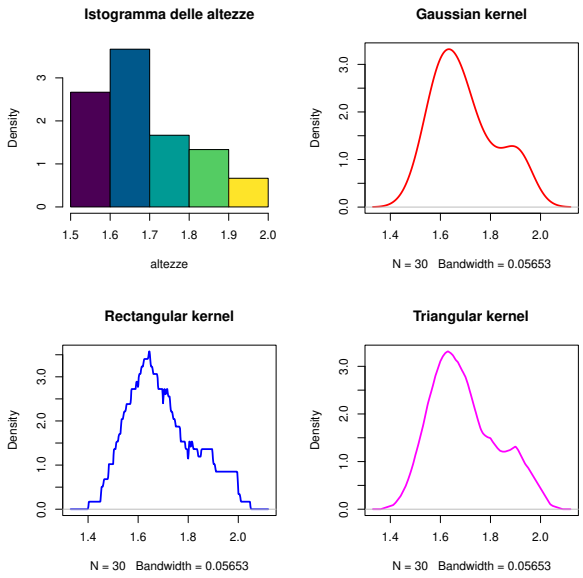

Figura 3.4: Confronto tra l'istogramma e le stime delle densità basate su kernel gaussiano, rettangolare e triangolare.

con i restanti kernel per lo stesso insieme di dati.

3.4 Boxplot

Consideriamo un campione $(x_1, x_2, \ldots, x_n)$ dei valori assunti da una *variabile quantitativa* X. Procediamo ad ordinare i valori del campione in ordine crescente. Si chiama *primo quartile*, e si indica con Q_1, il valore per il quale il 25% dei dati sono alla sua sinistra e il restante 75% alla sua destra. Analogamente si chiama *terzo quartile*, e si indica con Q_3, il valore per il quale il 75% dei dati sono alla sua sinistra e il restante 25% alla sua destra. Il *secondo quartile* Q_2, ossia il valore per il quale 50% dei dati sono alla sua sinistra e il restante 50% è alla sua destra è detto *mediana*. Q_0 e Q_4 forniscono il *minimo* e il *massimo* dei valori del campione. In R i quartili si calcolano tramite la funzione quantile(nomeVettore) e la funzione summary(nomeVettore) permette di determinare i valori precisi del minimo, del massimo, della media, della mediana, del primo e del terzo quantile. Ad esempio, riferendoci al vettore voti di 30 studenti risulta:

```
> voti<-c(18,19,20,30,29,28,21,22,23,27,26,25,24,25,
+ 26,24,23,22,27,28,21,24,25,25,27,19,21,28,29,28)
> quantile(voti)
```

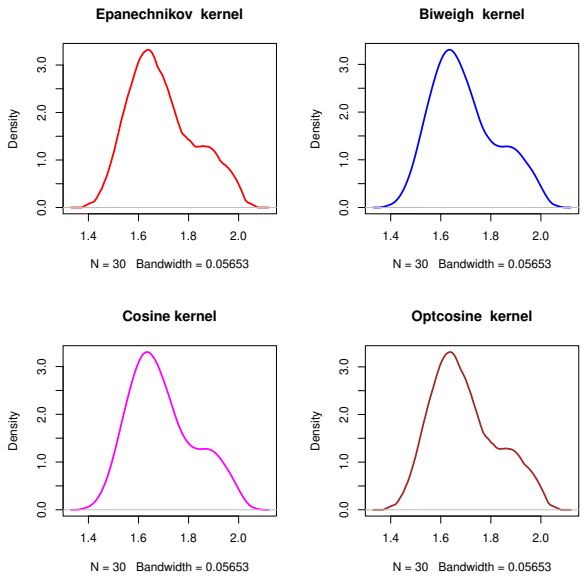

Figura 3.5: Confronto tra le stime delle densità basate sugli altri kernel.

```
  0%  25%  50%  75% 100%
  18   22   25   27   30
```

da cui si deduce che $Q_0 = 18$, $Q_1 = 22$, $Q_2 = 25$, $Q_3 = 27$, $Q_4 = 30$. Inoltre, si ha

```
> summary(voti)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  18.00   22.00   25.00   24.47   27.00   30.00
```

Il *boxplot*, detto anche *scatola con baffi*, è il disegno di una scatola i cui estremi sono Q_1 e Q_3, tagliata da una linea orizzontale in corrispondenza di Q_2, ossia della mediana. In basso e in alto sono presenti altre due linee tratteggiate, dette i *baffi*. L'estremo del *baffo inferiore* corrisponde al *valore più piccolo tra le osservazioni che risulta maggiore o uguale di* $Q_1 - 1.5 \cdot (Q_3 - Q_1)$, mentre l'estremo del *baffo superiore* corrisponde al *valore più grande delle osservazioni che risulta minore o uguale* a $Q_3 + 1.5 \cdot (Q_3 - Q_1)$. La distanza tra il primo e il terzo quartile è detta *intervallo interquartile* o *scarto interquartile*.

Se tutti i dati del campione rientrano nell'intervallo

$$\big(Q_1 - 1.5 \cdot (Q_3 - Q_1),\ Q_3 + 1.5 \cdot (Q_3 - Q_1)\big)$$

gli estremi dei baffi sono posti in corrispondenza del minimo valore e del massimo valore del campione. Gli eventuali valori al di fuori dell'intervallo $\big(Q_1 - 1.5 \cdot (Q_3 - Q_1),\ Q_3 + 1.5 \cdot (Q_3 - Q_1)\big)$ sono visualizzati nel grafico sotto forma

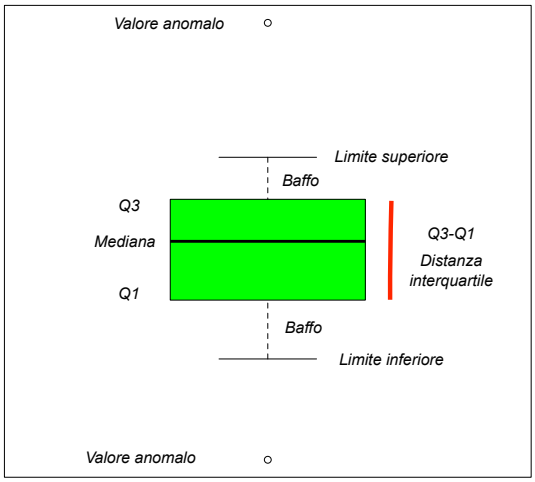

Figura 3.6: Boxplot.

di punti, detti *valori anomali* o *outlier*. Questi valori infatti costituiscono una "anomalia" rispetto alla maggior parte dei valori osservati e pertanto è necessario identificarli per poterne analizzare le caratteristiche e le eventuali cause che li hanno determinati. In Figura 3.6 è visualizzato un boxplot con eventuali valori anomali.

Il boxplot viene utilizzato per illustrare alcune caratteristiche di una distribuzione di frequenza: la *centralità*, la *forma*, la *dispersione* e la *presenza di eventuali valori anomali*, detti "outlier". La *centralità* è espressa dalla *mediana*. La *forma simmetrica o asimmetrica* può essere dedotta esaminando le distanze del primo e del terzo quartile dalla linea mediana. I baffi, superiore e inferiore, forniscono informazioni sulla dispersione e sulla forma della distribuzione ed anche sulle code della distribuzione. Infatti, la *dispersione* è deducibile esaminando le distanze dell'estremo del baffo superiore da Q_3 e dell'estremo del baffo inferiore da Q_1.

In R un boxplot è ottenuto tramite la funzione boxplot(nomeVettore). La figura può essere disegnata sia in senso verticale che orizzontale sulla base del parametro logico horizontal =, che ovviamente è FALSE (per default) nel caso verticale. Ad esempio, se desideriamo costruire un boxplot a partire dalla variabile quantitativa voti possiamo utilizzare il comando

```
> boxplot(voti,xlab="voti",main="Boxplot dei voti di 30 studenti",
    col="green")
```

che conduce al grafico illustrato di Figura 3.7. Si nota che gli estremi della scatola sono $Q_1 = 22$ e $Q_3 = 27$; essa è tagliata da una linea orizzontale in corrispondenza di $Q_2 = 25$. Il baffo inferiore corrisponde al valore più piccolo tra le osservazioni che risulta maggiore o uguale di $Q_1 - 1.5 \cdot (Q_3 - Q_1) = 14.5$, ossia 18, mentre il baffo superiore corrisponde al valore più grande delle osservazioni

che risulta minore o uguale a $Q_3 + 1.5 \cdot (Q_3 - Q_1) = 34.5$, ossia 30. Poiché tutti i voti cadono nell'intervallo $[14.5, 34.5]$ i baffi sono posti in corrispondenza del minimo voto 18 e del massimo voto 30.

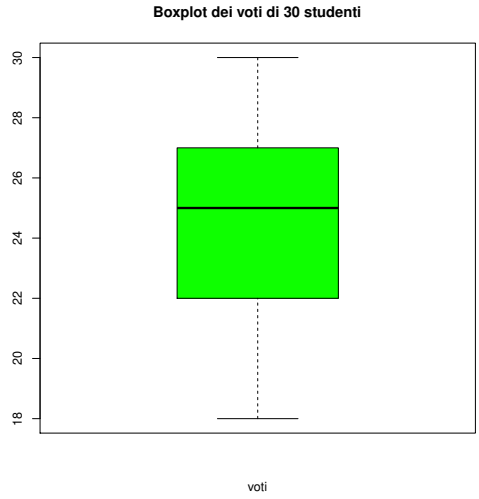

Figura 3.7: Boxplot relativo al vettore numerico "voti".

Se invece, aggiungiamo al vettore voti due valori anomali, ad esempio i valori 10 e 38 le seguenti linee di codice

```
> voti1<-c(voti,10,38)
>
> quantile(voti1)
   0%   25%   50%   75%  100%
10.00 21.75 25.00 27.25 38.00
>
> boxplot(voti1,xlab="voti1",main="Boxplot dei voti con outlier",
    col="green")
```

producono il grafico illustrato di Figura 3.8. Si nota che gli estremi della scatola sono $Q_1 = 21.75$ e $Q_3 = 27.25$; essa è tagliata da una linea orizzontale in corrispondenza di $Q_2 = 25$. Il baffo inferiore corrisponde al valore più piccolo tra le osservazioni che risulta maggiore o uguale di $Q_1 - 1.5 \cdot (Q_3 - Q_1) = 13.5$, ossia 18, mentre il baffo superiore corrisponde al valore più grande delle osservazioni che risulta minore o uguale a $Q_3 + 1.5 \cdot (Q_3 - Q_1) = 35.5$, ossia 30. Quindi i baffi sono stati posti in corrispondenza del minimo voto 18 e del massimo voto 30. Gli eventuali valori al di fuori dell'intervallo $[13.5, 35.5]$ sono 10 e 38 e sono visualizzati nel grafico sotto forma di punti, che costituiscono i due valori anomali.

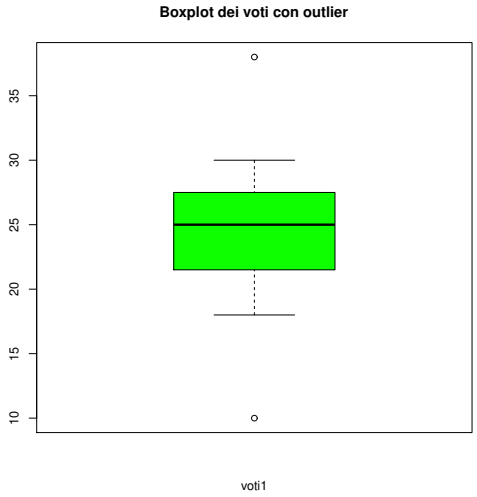

Figura 3.8: Boxplot del vettore "voti"con due valori anomali a 10 e 38.

3.5 Rappresentazioni grafiche per confrontare variabili

Per *confrontare differenti variabili* che descrivono insiemi di dati numerici di uno stesso *fenomeno quantitativo*, è possibile costruire un grafico che contiene i diversi boxplot delle distribuzioni associate alle diverse variabili. Supponiamo, ad esempio, di voler analizzare i voti riportati ad un certo esame universitario da studenti (uomini e donne) appartenenti a due differenti classi: "Classe A" e "Classe B". La classe A è costituita da 30 studenti, mentre nella classe B sono presenti 24 studenti. Nella Tabella 3.10 sono riportati i voti degli studenti delle due classi e il loro sesso. Si nota che i primi 14 studenti elencati della Classe A sono uomini e i restanti 16 studenti sono donne, mentre nella Classe B i primi 13 studenti elencati sono uomini e i restanti 11 studenti sono donne.

Tabella 3.10: Tabella dei voti riportati ad un esame nelle due classi

Classe A	18	19	20	30	29	28	21	22	23	27	26	25
Sesso	U	U	U	U	U	U	U	U	U	U	U	U
Classe A	24	25	26	24	23	22	27	28	21	24	25	25
Sesso	U	U	D	D	D	D	D	D	D	D	D	D
Classe A	27	19	21	28	29	28						
Sesso	D	D	D	D	D	D						
Classe B	19	19	20	27	19	28	21	22	23	26	27	25
Sesso	U	U	U	U	U	U	U	U	U	U	U	U
Classe B	24	25	26	24	23	22	28	29	21	24	21	19
Sesso	U	D	D	D	D	D	D	D	D	D	D	D

Rappresentiamo ora in uno stesso grafico i voti delle due classi in funzione dei loro indici, indicandoli con votiClasseA e votiClasseB associando simboli diversi, ossia con + per la Classe A e x per la Classe B. Inseriamo inoltre una legenda i cui caratteri hanno una grandezza opportuna. A tal fine le seguenti linee di codice

```
> votiClasseA<-c(18, 19, 20, 30, 29, 28, 21, 22, 23, 27,
+ 26, 25, 24, 25, 26, 24, 23, 22, 27, 28,
+ 21, 24, 25, 25, 27, 19, 21, 28, 29, 28)
> votiClasseB<-c(19, 19, 20, 27, 19, 28, 21, 22, 23, 26,
+ 27, 25, 24, 25, 26, 24, 23, 22, 28, 29,
+ 21, 24, 21, 19)
> plot(votiClasseA,pch="+",ylim=c(17,31),ylab="Voti delle due
    classi",col="blue")
> points(votiClasseB,pch="x",col="red")
> legend(25,31,c("Classe A","Classe B"),pch=c("+","x"),
+ col=c("blue","red"),bg="gray",cex=0.6)
```

producono il grafico illustrato in Figura 3.9.

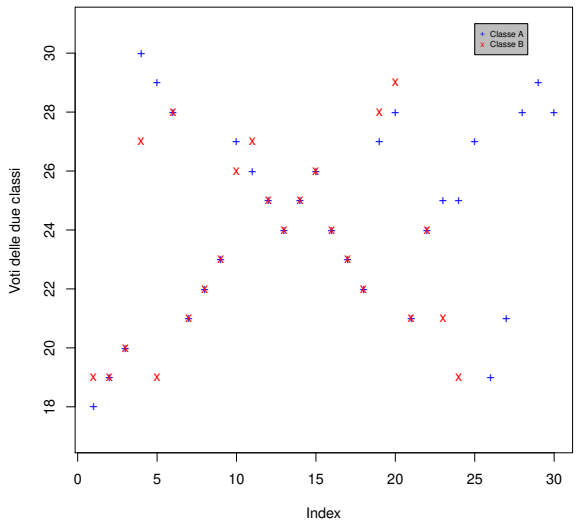

Figura 3.9: Rappresentazione dei voti riportati nelle due classi in corrispondenza del loro indice.

La funzione points() permette di aggiungere punti al grafico precedentemente rappresentato, il parametro pch individua il tipo di carattere da utilizzare, il parametro bg indica quale colore di sfondo utilizzare e il parametro cex indica la grandezza del testo e dei simboli generati.

Realizziamo ora un *grafico che permetta di confrontare i boxplot* delle distribuzioni dei voti delle due classi di studenti. A tal fine, la seguente linea di codice

```
> boxplot(votiClasseA,votiClasseB,
+ names=c("Classe A","Classe B"), col=c("green","orange"))
```

produce il grafico illustrato in Figura 3.10 che consiste dei due boxplot per la Classe A e per la Classe B.

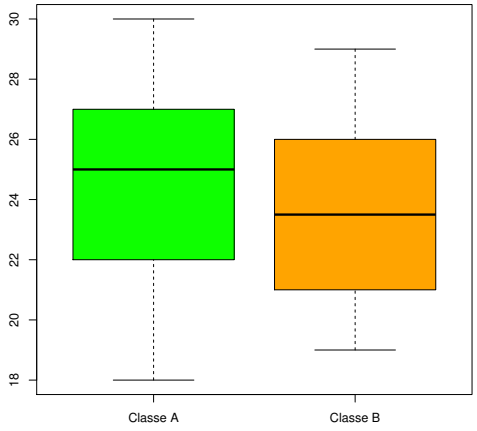

Figura 3.10: Confronto dei due boxplot relativi ai voti di un esame riportati in due differenti classi.

Se utilizziamo la funzione summary() possiamo visualizzare le principali misure statistiche:

```
> summary(votiClasseA)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  18.00   22.00   25.00   24.47   27.00   30.00
>
> summary(votiClasseB)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  19.00   21.00   23.50   23.42   26.00   29.00
```

che si possono visualizzare nel grafico di Figura 3.10.

È anche possibile visualizzare la statistica dei voti degli studenti uomini e donne della Classe A e della Classe B creando *due boxplot per la stessa variabile quantitativa* (voti) *categorizzata secondo i livelli di un fattore* (sesso degli studenti). Infatti, il codice seguente

```
> sessoA<-c(rep("Uomini",14),rep("Donne",16))
> sessoClasseA<-factor(sessoA)
> plot(sessoClasseA,votiClasseA,main="Classe A",
```

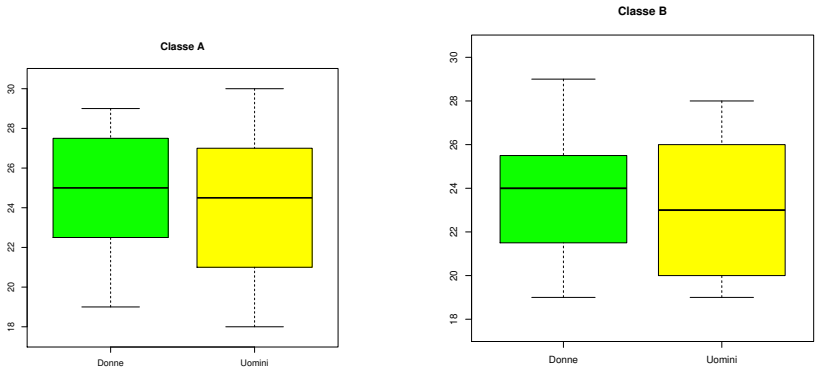

Figura 3.11: Confronto tra i boxplot relativi ai voti di un esame riportati dagli studenti uomini e donne della classe A (alla sinistra) e della classe B (alla destra).

```
+ ylim=c(17.5,30.5),col=c("green","yellow"))
```

permette di ottenere il grafico con due boxplot alla sinistra di Figura 3.11. Analogamente le linee di codice seguenti

```
> sessoB<-c(rep("Uomini",13),rep("Donne",11))
> sessoClasseB<-factor(sessoB)
> plot(sessoClasseB,votiClasseB,main="Classe B",
+ ylim=c(17.5,30.5),col=c("green","yellow"))
```

permettono di ottenere il grafico con due boxplot alla destra di Figura 3.11. Notiamo infine che abbiamo utilizzato per semplicità il comando rep() poiché nella tabella sono elencati prima gli uomini e poi le donne.

3.6 Boxplot ad intaglio

I *boxplot ad intaglio* (notched boxplots) sono una rappresentazione grafica dei boxplot che permette di visualizzare anche l'intervallo di confidenza (intervallo di fiducia) del 95% per la mediana. Se si sceglie come grado di fiducia nella stima della mediana $1-\alpha = 0.95$, si dimostra che per campioni numerosi l'intervallo di confidenza approssimato per la mediana è [1]

$$(M_1, M_2) = \Big(M - 1.57\,\frac{IQR}{\sqrt{n}}, M + 1.57\frac{IQR}{\sqrt{n}}\Big), \tag{3.1}$$

dove

- M è la mediana;

[1] McGill R., Tuket J.W., Larsen W.A. Variations of Box Plots. The American Statistician, **32**, 11–16, 1978.

- M_1 è l'estremo inferiore dell'intervallo di confidenza per la mediana;

- M_2 è l'estremo superiore dell'intervallo di confidenza per la mediana;

- $IQR = Q_3 - Q_1$ è lo scarto interquartile;

- n il numero di osservazioni nel campione.

L'intervallo di confidenza approssimato in (3.1) è basato sulla distribuzione normale ed è accurato per grandi campioni. Il numero 1.57 è dovuto ad un rapporto $(1.25 \cdot 1.7)/1.35$ che appare nell'approssimazione.

In R un boxplot ad intaglio è ottenuto tramite la funzione boxplot(nomeVettore) inserendo l'opzione notch = TRUE. Ad esempio, relativamente al vettore voti di 30 studenti universitari, il codice seguente produce il grafico di Figura 3.12.

```
> voti<-c(18,19,20,30,29,28,21,22,23,27,26,25,24,25,
+ 26,24,23,22,27,28,21,24,25,25,27,19,21,28,29,28)
>
> quantile(voti)
  0%  25%  50%  75% 100%
  18   22   25   27   30
  >
> boxplot(voti,notch=TRUE, xlab="voti",main="Boxplot ad intaglio
    dei voti di 30 studenti", col="green")
```

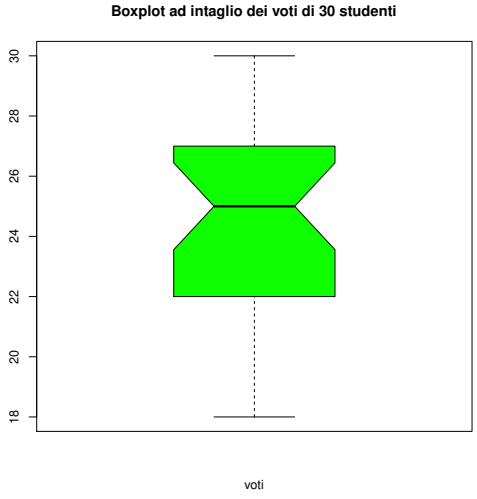

Figura 3.12: Boxplot ad intaglio relativo ai voti di un esame.

Si nota che

```
> IQR<-quantile(voti,0.75)-quantile(voti,0.25)
> M1<-quantile(voti,0.5)-1.57*IQR/sqrt(length(voti))
> M2<-quantile(voti,0.5)+1.57*IQR/sqrt(length(voti))
```

```
> c(M1,M2)
     50%      50%
23.56679 26.43321
```

Lo scarto interquartile è $IQR = 27 - 22 = 5$, la mediana è $M = 25$, l'ampiezza del campione è 30; quindi, con un grado di fiducia del 95% l'intervallo di confidenza approssimato per la mediana è $(23.6, 26.4)$.

Determiniamo ora l'intervallo di confidenza approssimato per la mediana del vettore voti1. il codice seguente produce il grafico di Figura 3.13.

```
> voti1<-c(voti,10,38)
>
> quantile(voti1)
   0%   25%   50%   75%  100%
10.00 21.75 25.00 27.25 38.00
>
> boxplot(voti1,notch=TRUE, xlab="voti",main="Boxplot ad intaglio
    dei voti di 30 studenti", col="green")
```

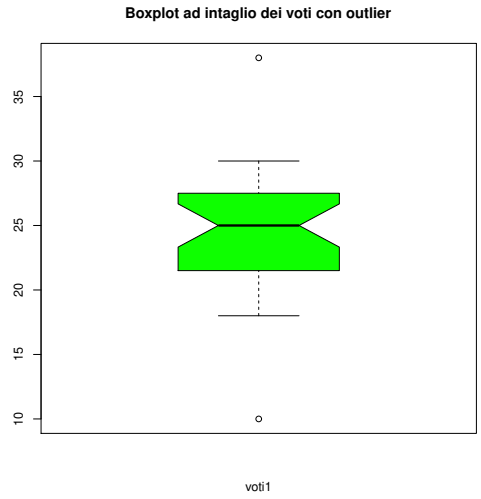

Figura 3.13: Boxplot ad intaglio relativo ai voti di un esame con valori anomali.

Si nota che

```
> IQR<-quantile(voti1,0.75)-quantile(voti1,0.25)
> M1<-quantile(voti1,0.5)-1.57*IQR/sqrt(length(voti1))
> M2<-quantile(voti1,0.5)+1.57*IQR/sqrt(length(voti1))
> c(M1,M2)
     50%      50%
23.47353 26.52647
```

Lo scarto interquartile è $IQR = 27.25 - 21.75 = 5.5$, la mediana è $M = 25$, l'ampiezza del campione è 32; quindi, con un grado di fiducia del 95% l'intervallo di confidenza approssimato per la mediana è $(23.5, 26.5)$.

I boxplot ad intaglio sono spesso utilizzati per confrontare gruppi. Infatti, se si effettua un test statistico sulla differenza tra le mediane dei due gruppi con un livello di significatività del 5%, per grandi campioni le mediane dei due gruppi differiscono statisticamente se gli intervalli di confidenza dei due boxplot non si sovrappongono.

Ad esempio, relativamente ai voti riportati da due classi di studenti, il codice seguente produce il grafico di Figura 3.14.

```
> votiClasseA<-c(18, 19, 20, 30, 29, 28, 21, 22, 23, 27,
+ 26, 25, 24, 25, 26, 24, 23, 22, 27, 28,
+ 21, 24, 25, 25, 27, 19, 21, 28, 29, 28)
>
> votiClasseB<-c(19, 19, 20, 27, 19, 28, 21, 22, 23, 26,
+ 27, 25, 24, 25, 26, 24, 23, 22, 28, 29,
+ 21, 24, 21, 19)
>
> summary(votiClasseA)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  18.00   22.00   25.00   24.47   27.00   30.00
 >
> summary(votiClasseB)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  19.00   21.00   23.50   23.42   26.00   29.00
 >
> boxplot(votiClasseA,votiClasseB,notch=TRUE,
+ names=c("Classe A","Classe B"), col=c("green","orange"))
```

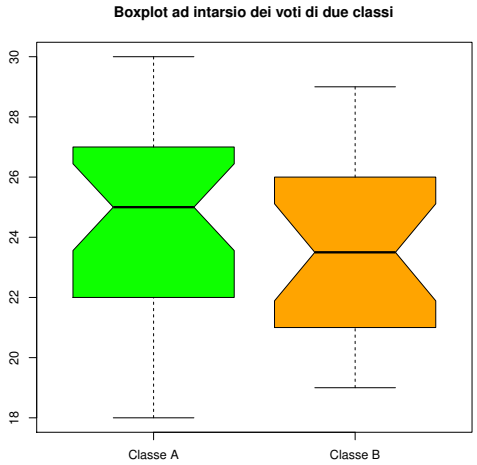

Figura 3.14: Boxplot ad intaglio relativo ai voti di un esame con valori anomali.

Per la classe A l'intervallo di confidenza approssimato per la mediana dei voti è $(23.6, 26.4)$. Calcoliamo anche l'intervallo di confidenza approssimato per la mediana della classe B

```
> IQR<-quantile(votiClasseB,0.75)-quantile(votiClasseB,0.25)
> M1<-quantile(votiClasseB,0.5)-1.57*IQR/sqrt(length(votiClasseB))
>
> M2<-quantile(votiClasseB,0.5)+1.57*IQR/sqrt(length(votiClasseB))
> c(M1,M2)
     50%      50%
21.89763 25.10237
```

Per la classe B l'intervallo di confidenza approssimato per la mediana dei voti è quindi $(21.9, 25.1)$. Poiché i due intervalli si sovrappongono, con un livello di significatività del 5% non si può affermare che le mediane dei voti delle due classi sono differenti. Infatti, la differenza tra le mediane varia nell'intervallo $(23.56679 - 25.10237, 26.43321 - 21.89763)$, ossia $(-1.53558, 4.53558)$.

Come vedremo nel prossimo capitolo, R mette a disposizione 9 differenti algoritmi per il calcolo dei quantili e l'intervallo di confidenza approssimato per le mediane potrebbe leggermente variare a seconda dell'algoritmo utilizzato. In generale, i quantili sono ottenuti scrivendo $\mathsf{quantile}(\mathsf{v}, \mathsf{probs} =, \mathsf{type} = \mathsf{j})$, dove v è un vettore numerico, probs è il vettore delle probabilità e $j = 1, 2, \ldots, 9$ denota il tipo di algoritmo selezionato.

3.7 Diagramma di Pareto

Nel 1897 l'economista italiano Vilfredo Pareto (1848–1923), studiando la distribuzione dei redditi, mostrò che in una data regione solo pochi individui possedevano la maggior parte della ricchezza. Questa osservazione ispirò la cosiddetta "*legge 80/20*", una legge empirica che è anche nota con il nome di *principio di Pareto*, sintetizzabile nell'affermazione: *la maggior parte degli effetti è dovuta ad un numero ristretto di cause (considerando grandi numeri).*

Secondo la legge 80/20 (i valori 80% e 20% sono ottenuti mediante osservazioni empiriche di numerosi fenomeni e sono solo indicativi), *l'80% dei risultati dipende dal 20% delle cause*. Questo principio può avere diverse applicazioni pratiche in diversi settori, ad esempio:

- *Economia*: l'80% delle ricchezze è in mano al 20% della popolazione (ma ovviamente i valori reali variano a seconda dei paesi e dei periodi);

- *Qualità*: il 20% dei tipi di guasti possibili in un processo produttivo genera l'80% delle non conformità totali;

- *Informatica*: l'80% del tempo di esecuzione è impiegato soltanto dal 20% delle istruzioni di un programma;

- *Telefonate*: l'80% delle chiamate viene effettuato da e verso il 20% dei contatti in memoria;

- *Trasporto stradale*: sul 20% delle strade avviene l'80% di tutti gli spostamenti;

- *Internet*: l'80% del traffico dati è generato dal 20% dei siti web;

- *Consumo*: il 20% dei prodotti fatturano l'80% dei guadagni.

Sfruttando queste osservazioni, è possibile analizzare un insieme di dati in modo da determinare le poche variabili (fra le tante prese in esame) che influenzano in modo significativo i risultati finali di un determinato fenomeno (analisi di Pareto). Uno strumento utile a tale scopo è il cosiddetto *diagramma di Pareto* che permette di analizzare fenomeni qualitativi. Il *diagramma di Pareto* consiste di un diagramma a barre verticali con le modalità ordinate in ordine decrescente rispetto alla loro frequenza relativa; inoltre le frequenze relative sono visualizzate anche nella loro forma cumulata, mediante una sequenza di segmenti crescenti.

L'analisi basata sul diagramma di Pareto consiste nelle seguenti fasi:

1. decidere come classificare i dati;
2. rilevare i dati ed ordinarli;
3. disegnare il diagramma;
4. costruire la linea cumulativa;
5. aggiungere le informazioni di base.

Ad esempio, si supponga di dover valutare la qualità di un prodotto, come ad esempio un computer portatile, e di aver osservato, per un determinato intervallo temporale, la *numerosità dei difetti dei componenti*. La prima fase consiste nella scelta del metodo con cui classificare i dati da raccogliere e supponiamo, ad esempio, di classificare i prodotti difettosi per tipo di componente difettoso. Nella seconda fase i dati raccolti per un determinato arco temporale sono riportati sinteticamente in una tabella: tipo di componente difettoso e loro numero (vedi Tabella 3.11).

Tabella 3.11: Tabella contenente la numerosità dei difetti (frequenza assoluta) dei componenti di un computer portatile.

Elenco	Numero pezzi difettosi
Audio	9
Dvd	4
Monitor	3
Tastiera	30
Touchpad	15
Webcam	2
WiFi	6
Altro	1
Totale	70

Prima di costruire il diagramma di Pareto risulta utile, per una visione immediata, riordinare le voci in una nuova tabella in base alla rilevanza del parametro in esame; essendo stata scelta nell'esempio la quantità, si elenca prima il difetto più numeroso, poi il successivo e così via. I dati così ordinati (vedi Tabella 3.12)

costituiscono la base per la costruzione del diagramma di Pareto. Ai fini della rappresentazione nel diagramma di Pareto, poiché in generale i tipi di difetti sono numerosi, è utile raggruppare i difetti minori sotto la voce altro o altre cause. Tale voce, in generale, dovrebbe risultare sempre all'ultimo posto, quindi numericamente non rilevante, altrimenti vuol dire che in essa sono confluite tipologie di difetti numerosi, che invece meritano una valutazione specifica.

Tabella 3.12: Tabella contenente il numero di pezzi difettosi (frequenza assoluta), la frequenza relativa e la percentuale cumulata

Elenco	Frequenza	Frequenza relativa	Percentuale cumulata
Tastiera	30	0.42857143	43%
Touchpad	15	0.21428571	64%
Audio	9	0.12857143	77%
WiFi	6	0.08571429	86%
Dvd	4	0.05714286	91%
Monitor	3	0.04285714	96%
Webcam	2	0.02857143	99%
Altro	1	0.01428571	100%
Totale	70		

La fase 3 consiste nella rappresentazione grafica dei dati rilevati costruendo un grafico a barre. Sull'asse verticale viene riportato la frequenza relativa dei difetti rilevati e sull'asse orizzontale i tipi di difetti. Nel riportare sull'asse orizzontale i tipi di difetti all'estrema sinistra verrà indicato il difetto rilevato più frequentemente poi gli altri nell'ordine decrescente della tabella. Nell'Esempio considerato si vede che il difetto più frequente è stato rilevato 30 volte per la tastiera; quindi si traccerà una colonna alta fino al livello 30/70=0.42857143. Il secondo difetto più frequente è stato rilevato 15 volte: a lato della colonna precedentemente disegnata ne tracceremo un'altra che arrivi al livello 15/70=0.21428571 e che sia larga come la prima. Con lo stesso criterio si tracceranno le altre colonne, sempre rispettando l'ordine decrescente delle frequenze relative.

La fase 4 consiste nel tracciare la cosiddetta linea dei valori cumulati o linea cumulativa Questa linea risulta utile quando si vogliono individuare le percentuali cumulate di più colonne. La linea dei valori percentuali cumulati è rappresentata da una spezzata. Il primo segmento congiunge il punto che indica la percentuale dei difetti del primo tipo con quello ad altezza pari alla somma delle percentuali dei difetti del primo e secondo tipo. Si ripete il procedimento per i segmenti successivi; l'ultimo segmento della linea cumulativa terminerà nel punto più alto della scala percentuale, corrispondente al 100% dei difetti.

Nel linguaggio R non è presente una funzione in grado di generare un diagramma di Pareto. Tuttavia, riferendoci all'esempio considerato, è possibile creare facilmente questo tipo di grafico con il seguente codice:

```
> computer <- c(rep("Audio", 9), rep("Dvd", 4), rep("Monitor", 3),
+ rep("Tastiera", 30), rep("Touchpad", 15), rep("Webcam", 2),
+ rep("WiFi", 6), rep("Altro", 1))
```

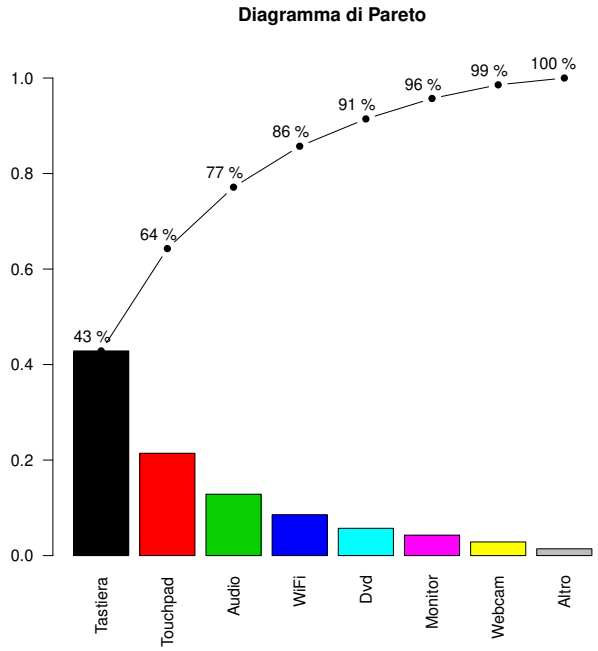

Figura 3.15: Diagramma di Pareto ottenuto osservando la numerosità dei componenti difettosi in un fissato intervallo temporale.

```
> tab <- table(computer)
> ord <- sort(tab,decreasing=TRUE)
> propOrd <- prop.table(ord)
> x <- barplot(propOrd, ylim = c(0, 1.05), main = "Diagramma di
    Pareto", col=1:8,las=2)
> lines(x, cumsum(propOrd), type = "b", pch = 16)
> text(x - 0.2, cumsum(propOrd) + 0.03, paste(format(cumsum(propOrd
    ) * 100,digits = 2), "%"))
```

che permette di ottenere il grafico illustrato in Figura 3.15. Abbiamo utilizzato la funzione barplot(propOrd) che produce un grafico a barre dei valori presenti nel vettore propOrd in base alle frequenze relative; ylim fornisce il limite superiore dell'asse delle ordinate; main il titolo della figura che appare in testa al grafico. Si utilizzano poi le funzioni a basso livello lines(x, cumsum(propOrd)) con il comando type = "b" che consente di tracciare punti connessi da linee e pch = 16 che controlla il tipo di carattere da utilizzare. Inoltre, text() permette di aggiungere del testo nella posizione indicata da (x − 0.2, cumsum(propOrd) + 0.03), paste() permette di concatenare vettori dopo averli convertiti in caratteri e format() permette di formattare i numeri utilizzando un numero appropriato di cifre.

Osservando il diagramma di Pareto è possibile determinare su quali tipologie di difetti occorre intervenire prioritariamente. Nel caso specifico, circa l'80% del

numero difetti è dovuto soltanto alle prime tre tipologie. Dovendo quindi stabilire una strategia di ottimizzazione, conviene migliorare la tastiera, il touchpad e il sistema audio che appresentano l'80% della difettosità totale.

Il grafico ha la capacità di mostrare immediatamente quali modalità si presentano con maggiore frequenza e può aiutare a stabilire quali di esse principalmente influenzano un determinato fenomeno. Una caratteristica importante dell'analisi di Pareto è legata alla sua versatilità e facilità di applicazione in ogni campo in cui occorre analizzare aspetti di qualità, di efficienza, di sicurezza, di affidabilità, di costi, ecc. Il diagramma di Pareto permette anche di confrontare con facilità due rappresentazioni dello stesso fenomeno relative a tempi o a condizioni differenti e quindi di evidenziare i risultati di eventuali azioni di miglioramento intraprese. È quindi un utile strumento di analisi nei processi decisionali, nella gestione della qualità, nelle strategie di marketing per nuovi prodotti ed in numerosi altri settori.

3.8 Grafici di funzioni

Con il linguaggio R si possono rappresentare graficamente molte funzioni matematiche, utilizzando la funzione curve(expr, from, to) che disegna una curva sulla base dell'espressione indicata in expr, nell'intervallo [from, to]. Inoltre, per aggiungere una curva ad un grafico precedentemente realizzato basta utilizzare il parametro add posto a TRUE (di default è FALSE). Ad esempio, supponiamo di voler tracciare il grafico della funzione

$$y = f(x) = \frac{3\,x^2 + 7\,x + 7}{x^2 + x + 1}. \tag{3.2}$$

che ha un asintoto orizzontale in $y = 3$, è decrescente nell'intervallo $(-\infty, -2)$, è crescente in $(-2, 0)$ ed è decrescente in $(0, +\infty)$. Le seguenti linee di codice

```
> curve( (3*x^2+7*x+7)/(x^2+x+1),from=-15,
+ to=15, col="red")
> abline(h=3,lty=2,col="blue")
```

producono il grafico illustrato in Figura 3.16. La funzione abline() permette di aggiungere al grafico della funzione (3.2) una linea orizzontale che ha ordinata y = 3 rappresentante l'asintoto orizzontale. L'argomento lty = 2 specifica che tale linea deve essere tratteggiata.

È possibile disegnare più grafici nella stessa finestra grafica utilizzando le funzioni par(mfrow = c(nr, nc)) e par(mfcol = c(nr, nc)) e specificando il numero nr di grafici da mettere in riga e il numero di grafici nc da mettere in colonna. Ad esempio le linee di codice seguenti

```
> par(mfrow = c(2, 2))
> curve(cos(x), from = 0, to = 6 * pi)
> abline(h = 0, col="red")
> title("Funzione coseno")
> curve(sin(x), from = 0, to = 6 * pi)
> abline(h = 0,col="red")
```

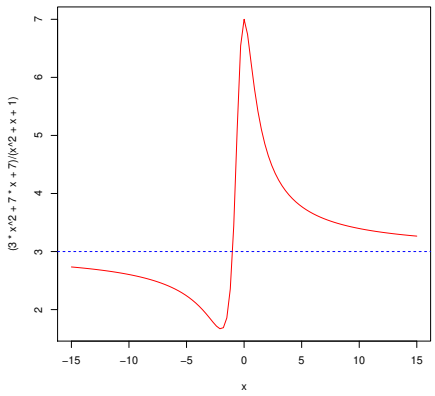

Figura 3.16: Rappresentazione della funzione definita in (3.2).

```
> title("Funzione seno")
> curve(x * cos(x) - sin(x), from = 0, to = 6 * pi)
> abline(h = 0,col="red")
> curve(+x, add = TRUE, lty = 2,col="blue")
> curve(-x, add = TRUE, lty =2,col="blue")
> title("Funzione con \noscillazioni")
> curve(sin(x)/x, from = 0, to = 6 *pi)
> abline(h =0,col="red")
> curve(1/x, add =TRUE, lty =3, from =1, to =6 * pi,col="black")
> curve(-1/x, add = TRUE, lty =3, from = 1, to = 6 * pi,col="black"
    )
> title("Funzione con \noscillazioni smorzate")
```

creano il grafico di Figura 3.17 composto da quattro grafici, due per ogni riga. Il parametro mfrow specifica che l'immissione dei grafici avviene per riga.

Si nota che nel terzo grafico e nel quarto grafico è stato utilizzato il parametro add = TRUE che permette di rappresentare nuove funzioni sul grafico corrente. In particolare, nel terzo grafico relativo alla funzione $x\cos x - \sin x$ sono state rappresentate le curve tratteggiate $y = -x$ e $y = x$; invece nel quarto grafico relativo alla funzione $\sin x/x$ sono state rappresentate le curve tratteggiate $y = -1/x$ e $y = 1/x$.

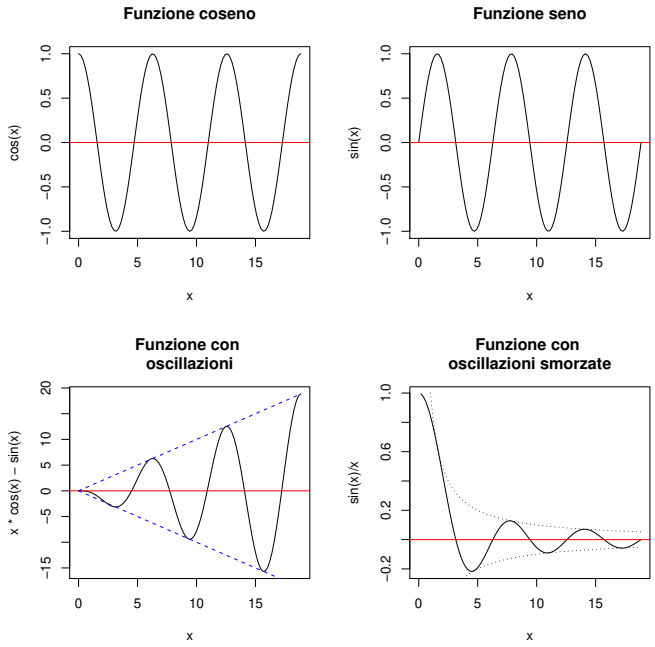

Figura 3.17: Rappresentazione delle funzioni $\cos(x)$, $\sin(x)$, $x\cos(x) - \sin(x)$ e $\sin(x)/x$.

Capitolo 4

Statistica descrittiva univariata

4.1 Introduzione

La *statistica descrittiva* è costituita da un insieme di metodi di natura logica e matematica atti a raccogliere, elaborare, analizzare ed interpretare dati allo scopo di descrivere fenomeni collettivi e di estendere la descrizione di certi fenomeni osservati ad altri fenomeni dello stesso tipo non ancora osservati. Questi fenomeni possono riguardare particolari aspetti del mondo reale di natura economica, industriale, sociale oppure descrivere comportamenti o situazioni riguardanti singoli individui o insiemi di individui.

La statistica descrittiva è utilizzata per analizzare il comportamento dei fenomeni oggetto di studio. Ogni fenomeno può essere descritto tramite opportune categorie di dati di *tipo qualitativo* oppure di *tipo quantitativo discreti o continui*. I dati sono utilizzati per ricavare *misure di sintesi* che consentano di comprendere il comportamento del fenomeno in esame. Sulla base dell'analisi dei dati è spesso possibile formulare opportune ipotesi statistiche da sottoporre successivamente a procedimenti di verifica mediante gli strumenti tipici dell'inferenza statistica.

Prima di iniziare una qualsiasi *elaborazione dei dati* è necessario avere informazioni generali sul fenomeno riguardanti:

- la natura del fenomeno in esame;
- il numero di osservazioni disponibili (ampiezza del campione);
- il numero di variabili utilizzate per rappresentare i diversi aspetti del fenomeno in esame (numero di caratteristiche);
- il tipo di informazione disponibile per ciascuna variabile (qualitativa o quantitativa);
- lo scopo che l'analisi esplorativa dei dati si propone di raggiungere.

L'indagine statistica è sempre effettuata su un insieme di entità (individui, oggetti,...) in cui si manifesta il fenomeno che si studia. Questo insieme è detto

popolazione o *universo* e può essere costituito da un numero finito oppure infinito di unità; nel primo caso si parla di *popolazione finita* e nel secondo caso di *popolazione illimitata*. La conoscenza delle caratteristiche di una *popolazione finita* può essere ottenuta osservando la totalità delle entità della popolazione oppure un sottoinsieme di questa, detto *campione* estratto dalla popolazione. Una popolazione illimitata può invece essere studiata solo tramite un *campione* estratto dalla popolazione. Poiché il campione deve contenere informazioni sulla popolazione complessiva, deve essere rappresentativo di quella popolazione. In generale, per ottenere campioni rappresentativi di una popolazione occorre scegliere gli elementi in modo completamente casuale poiché ogni criterio di selezione non casuale rischia di produrre campioni sbilanciati verso particolari valori.

In questo capitolo ci occupiamo della statistica descrittiva univariata e nel capitolo successivo della statistica descrittiva bivariata. Nel Paragrafo 4.2 definiamo la *funzione di distribuzione empirica discreta* e la *funzione di distribuzione empirica continua*. Nei Paragrafi 4.3 – 4.6 discutiamo come si possono ottenere informazioni di sintesi su un campione sperimentale introducendo alcune misure statistiche, ossia gli indici di posizione centrali e non centrali e gli indici di dispersione. Invece, nel Paragrafo 4.8 definiamo degli ulteriori indici statistici che consentono di ricavare informazioni sulla forma di una distribuzione di frequenza. Nel Paragrafo 4.9 consideriamo la media ponderata

4.2 Funzione di distribuzione empirica

Per i *fenomeni quantitativi* è spesso utile definire la funzione di distribuzione empirica. Nel seguito descriviamo la *funzione di distribuzione empirica discreta* e la *funzione di distribuzione empirica continua*.

Se, ad esempio, siamo interessati al numero di figli di un campione di 30 famiglie utilizziamo la funzione di distribuzione empirica discreta. Analogamente, se disponiamo di un campione con i risultati, ossia $1, 2, 3, 4, 5, 6$, di 100 lanci di un dado utilizziamo la funzione di distribuzione empirica discreta. Per fenomeni quantitativi continui i dati debbono essere organizzati in classi ed occorre considerare una funzione di distribuzione empirica continua.

4.2.1 Funzione di distribuzione empirica discreta

Nel caso discreto questa funzione è definita a partire dalle frequenze relative cumulative. Consideriamo una variabile quantitativa X e indichiamo con $z_1, z_2, \ldots, z_k$ i valori distinti da essa assunti e assumiamo che essi siano ordinati in ordine crescente, ossia $z_1 < z_2 < \ldots < z_k$. Consideriamo poi un campione $(x_1, x_2, \ldots, x_n)$ costituito da n osservazioni di X. Denotiamo con n_i il numero di volte in cui ciascun valore z_i è presente nel campione, ossia la frequenza assoluta con cui esso appare nel campione, e con $f_i = n_i/n$ le frequenze relative. Le *frequenze relative cumulative* sono così definite:

$$F_i = f_1 + f_2 + \ldots + f_i = \frac{n_1 + n_2 + \ldots + n_i}{n} \qquad (i = 1, 2, \ldots, k), \tag{4.1}$$

dove la generica F_i rappresenta la *proporzione dei dati del campione minori o uguali di* z_i.

Se supponiamo che i k valori distinti assunti dalla variabile quantitativa X siano ordinati in ordine crescente, ossia $z_1 < z_2 < \ldots < z_k$, allora la *funzione di distribuzione empirica* $F(x)$ è così definita:

$$F(x) = \frac{\#\{x_i \leq x, i = 1, 2, \ldots, n\}}{n} = \begin{cases} 0, & x < z_1 \\ F_1, & z_1 \leq x < z_2 \\ \ldots & \\ F_i, & z_i \leq x < z_{i+1} \\ \ldots & \\ 1, & x \geq z_k \end{cases} \tag{4.2}$$

dove # indica la cardinalità dell'insieme. La funzione di distribuzione empirica $F(x)$ è definita per ogni x reale ed è una *funzione a gradini* in cui ogni gradino indica quale proporzione di dati presenta un valore minore o uguale di quello indicato sull'asse delle ascisse.

La funzione di distribuzione empirica $F(x)$ gode delle seguenti proprietà:

- è una funzione non decrescente;

- la funzione assume il valore a sinistra in corrispondenza ad ogni punto di salto;

- la funzione vale 0 per ogni valore minore dell'osservazione minima e vale 1 per ogni valore maggiore o uguale dell'osservazione massima.

Il linguaggio R dispone della classe stepfun che implementa una serie di metodi per trattare funzioni a gradino. In particolare, la funzione ecdf() (*empirical cumulative distribution function*) permette di disegnare il grafico della funzione di distribuzione empirica per variabili quantitative discrete.

Ad esempio, consideriamo i voti conseguiti da 30 studenti in un fissato esame universitario, ossia:

$$18, 19, 20, 30, 29, 28, 21, 22, 23, 27, 26, 25, 24, 25, 26,$$
$$24, 23, 22, 27, 28, 21, 24, 25, 25, 27, 19, 21, 28, 29, 28$$

Nella Tabella 4.1 sono indicate le frequenze assolute, le frequenze relative, le frequenze relative cumulative e la funzione di distribuzione empirica discreta.

Le seguenti linee di codice

```
> voti<-c(18,19,20,30,29,28,21,22,23,27,26,25,24,25,
+ 26,24,23,22,27,28,21,24,25,25,27,19,21,28,29,28)
>
> round(cumsum(table(voti)/length(voti)),3)
   18    19    20    21    22    23    24    25    26    27    28
           29    30
0.033 0.100 0.133 0.233 0.300 0.367 0.467 0.600 0.667 0.767 0.900
    0.967 1.000
```

Tabella 4.1: Frequenze assolute, frequenze relative, frequenze relative cumulative e funzione di distribuzione empirica discreta del voti dei 30 studenti.

i	z_i	n_i	f_i	F_i
1	18	1	1/30	1/30
2	19	2	2/30	3/30
3	20	1	1/30	4/30
4	21	3	3/30	7/30
5	22	2	2/30	9/30
6	23	2	2/30	11/30
7	24	3	3/30	14/30
8	25	4	4/30	18/30
9	26	2	2/30	20/30
10	27	3	3/30	23/30
11	28	4	4/30	27/30
12	29	2	2/30	29/30
13	30	1	1/30	30/30

$$F(x) = \begin{cases} 0, & x < 18 \\ 1/30, & 18 \leq x < 19 \\ 3/30, & 19 \leq x < 20 \\ 4/30, & 20 \leq x < 21 \\ 7/30, & 21 \leq x < 22 \\ 9/30, & 22 \leq x < 23 \\ 11/30, & 23 \leq x < 24 \\ 14/30, & 24 \leq x < 25 \\ 18/30, & 25 \leq x < 26 \\ 20/30, & 26 \leq x < 27 \\ 23/30, & 27 \leq x < 28 \\ 27/30, & 28 \leq x < 29 \\ 29/30, & 29 \leq x < 30 \\ 1, & x \geq 30 \end{cases}$$

permettono di ottenere le frequenze relative cumulative relative al vettore voti, arrotondate alla terza cifra decimale. La seguente linea di codice

```
> plot(ecdf(voti),main="Funzione di distribuzione empirica
+ discreta",verticals=FALSE,col="red")
```

produce il grafico illustrato in Figura 4.1. Se si utilizza l'opzione verticals = TRUE i segmenti sono uniti da segmenti verticali.

Se inoltre desideriamo conoscere il valore della funzione di distribuzione empirica discreta nel punto $x = 27.5$ occorre scrivere:

```
> ecdf(voti)(27.5)
[1] 0.7666667
```

che, osservando la Tabella 4.1, corrisponde a 23/30.

4.2.2 Funzione di distribuzione empirica continua

Per *fenomeni quantitativi continui* occorre considerare la funzione di distribuzione empirica continua, ossia una funzione di distribuzione empirica strutturata in classi. Supponiamo di organizzare i dati numerici in k distinte classi $C_1 = [z_0, z_1), C_2 = [z_1, z_2) \ldots, C_k = [z_{k-1}, z_k]$, con $z_0 < z_1 < \ldots < z_{k-1} < z_k$, dove z_0 corrisponde al *minimo delle osservazioni* e z_k al *massimo delle osservazioni.*

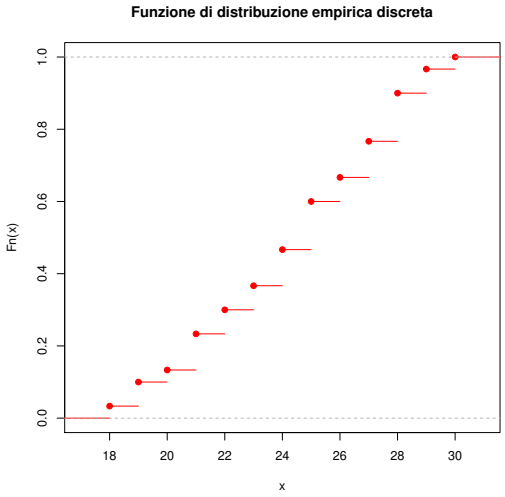

Figura 4.1: Funzione di distribuzione empirica discreta del vettore voti.

La funzione di distribuzione empirica continua è così definita:

$$F(x) = \begin{cases} 0, & x < z_0 \\ \cdots\cdots & \\ F_{i-1}, & x = z_{i-1} \\ \dfrac{F_i - F_{i-1}}{z_i - z_{i-1}}\, x + \dfrac{z_i\, F_{i-1} - z_{i-1}\, F_i}{z_i - z_{i-1}}, & z_{i-1} < x < z_i \\ F_i, & x = z_i \\ \cdots\cdots & \\ 1, & x \geq z_k, \end{cases} \tag{4.3}$$

dove $F_0 = 0$ e F_i denota la frequenza relativa cumulativa della classe C_i $(i = 1, 2, \ldots, k)$. Si nota che $F(x) = 0$ per $x < z_0$, $F(x) = 1$ per $x \geq z_k$, mentre se $z_{i-1} < x < z_i$ la funzione di distribuzione empirica continua coincide con il segmento che passa per i punti (z_{i-1}, F_{i-1}) e (z_i, F_i), ossia

$$\frac{y - F_{i-1}}{x - z_{i-1}} = \frac{F_i - F_{i-1}}{z_i - z_{i-1}} \qquad (i = 1, 2, \ldots, k),$$

da cui segue

$$y = F_{i-1} + \frac{F_i - F_{i-1}}{z_i - z_{i-1}}\,(x - z_{i-1}) = \frac{F_i - F_{i-1}}{z_i - z_{i-1}}\, x + \frac{F_{i-1}(z_i - z_{i-1}) - z_{i-1}\,(F_i - F_{i-1})}{z_i - z_{i-1}}$$
$$= \frac{F_i - F_{i-1}}{z_i - z_{i-1}}\, x + \frac{z_i\, F_{i-1} - z_{i-1}\, F_i}{z_i - z_{i-1}}.$$

Ad esempio, riferendoci al vettore voti contenente i voti dei 30 studenti, introduciamo le seguenti classi $C_1 = [18, 21)$, $C_2 = [21, 24)$, $C_3 = [24, 27)$,

$C_4 = [27, 30]$. Nella Tabella 4.2 sono indicate le frequenze assolute, le frequenze relative e le frequenze relative cumulative associate al vettore voti utilizzando le classi $C_1 = [18, 21)$, $C_2 = [21, 24)$, $C_3 = [24, 27)$, $C_4 = [27, 30]$, essendo 18 il minimo e 30 il massimo nel vettore voti considerato. In questo caso, scegliamo $z_0 = 18$, $z_1 = 21$, $z_2 = 24$, $z_3 = 27$, $z_4 = 30$.

Tabella 4.2: Frequenze assolute, relative e relative cumulative delle classi associate al vettore voti di 30 studenti.

i	C_i	n_i	f_i	F_i
1	$[18, 21)$	4	4/30	4/30
2	$[21, 24)$	7	7/30	11/30
3	$[24, 27)$	9	9/30	20/30
4	$[27, 30]$	10	10/30	30/30

La funzione di distribuzione empirica continua associata ai vettore voti di 30 studenti è:

$$F(x) = \begin{cases} 0, & x < 18 \\ \dfrac{4}{90}\,(x - 18), & 18 \le x < 21 \\ \dfrac{1}{90}\,(7\,x - 135), & 21 \le x < 24 \\ \dfrac{1}{90}\,(9\,x - 183), & 24 \le x < 27 \\ \dfrac{1}{90}\,(10\,x - 210), & 27 \le x < 30 \\ 1, & x \ge 30 \end{cases}$$

Le seguenti linee di codice

```
> voti<-c(18,19,20,30,29,28,21,22,23,27,26,25,24,25,
+ 26,24,23,22,27,28,21,24,25,25,27,19,21,28,29,28)
> freqrel<-table(voti)/length(voti)
> round(freqrel,3) # visualizza la frequenza relativa dei voti
voti
   18    19    20    21    22    23    24    25    26    27    28
         29    30
0.033 0.067 0.033 0.100 0.067 0.067 0.100 0.133 0.067 0.100 0.133
    0.067 0.033
> m<-length(freqrel) # visualizza la lunghezza del vettore
    frequenza
> m
[1] 13
> classi<-c(18,21,24,27,30)
> frelclassi<-table(cut(voti,breaks=classi,right=FALSE ))/length(
    voti)
>  frelclassi # visualizza le frequenze relative delle classi
  [18,21)   [21,24)   [24,27)   [27,30)
0.1333333 0.2333333 0.3000000 0.3000000
>
>  Fcum<-cumsum(frelclassi)
```

```
> Fcum
  [18,21)   [21,24)   [24,27)   [27,30)
0.1333333 0.3666667 0.6666667 0.9666667
>
>  Fcum[4]<-Fcum[4]+freqrel[m] # modifica la frequenza relativa
    cumulata dell'ultima classe
>  Fcum # visualizza le frequenze relative cumulative delle classi
    considerate. 30 e' incluso nell'ultima classe.
  [18,21)   [21,24)   [24,27)   [27,30)
0.1333333 0.3666667 0.6666667 1.0000000
```

permettono di visualizzare le frequenze relative cumulative associate alle classi $C_1 = [18, 21)$, $C_2 = [21, 24)$, $C_3 = [24, 27)$, $C_4 = [27, 30]$. Per ottenere tali frequenze si è utilizzata la funzione cut() con l'opzione right = FALSE che consente di costruire classi con intervalli chiusi a sinistra invece che a destra ed anche la funzione cumsum() che permette di ottenere le frequenze cumulative. Ricordiamo che in R la funzione cut() trasforma dati numerici in dati qualitativi mediante la loro collocazione in opportune classi sulla base di quanto specificato nel parametro breaks. La frequenza cumulativa dell'ultima classe è stata modificata poiché l'intervallo in C_4 deve essere chiuso anche a destra.

Vogliamo ora scrivere delle linee di codice in R che consentono di ottenere il grafico della funzione di distribuzione empirica continua introducendo due classi fittizie $C_0 = (14, 18)$ e $C_5 = (30, 34)$ che ci serviranno per tracciare la linea $y = 0$ nell'intervallo $[14, 18)$ e la linea $y = 1$ nell'intervallo $[31, 34)$ nel grafico della funzione di distribuzione empirica continua. Infatti, le seguenti linee di codice

```
> ascisse<-c(14,18,21,24,27,30,34)
> ordinate<-c(0, 0,Fcum[1:4],1)
> plot(ascisse, ordinate, type = "b", axes = FALSE, main = "
    Funzione di distribuzione empirica continua",
col="red",ylim=c(0,1),xlab="x",ylab="F(x)")
> axis(1, ascisse)
> axis(2, format(Fcum, digits = 2))
> box()
```

producono il grafico illustrato in Figura 4.2.

L'utilizzazione della funzione di concatenazione c(0, 0, Fcum[1 : 4], 1) permette di aggiungere due zeri all'inizio del vettore delle frequenze relative cumulative e un 1 alla fine. La funzione plot() permette di rappresentare ordinatamente le coppie di punti (ascisse, ordinate). Nella funzione plot() si è utilizzata l'opzione type = "b" che consente di congiungere i punti successivi del grafico mediante linee continue ai cui estremi sono presenti dei cerchietti, mentre l'opzione axes = FALSE consente di non tracciare gli assi. Infine, mediante l'istruzione axis(1, ascisse) si disegna l'asse orizzontale in basso, mediante l'istruzione axis(2, format(Fcum, digits = 2)) si ottiene l'asse verticale di sinistra con una formattazione opportuna dei numeri. Successivamente mediante l'istruzione box() si racchiude il grafico in un rettangolo.

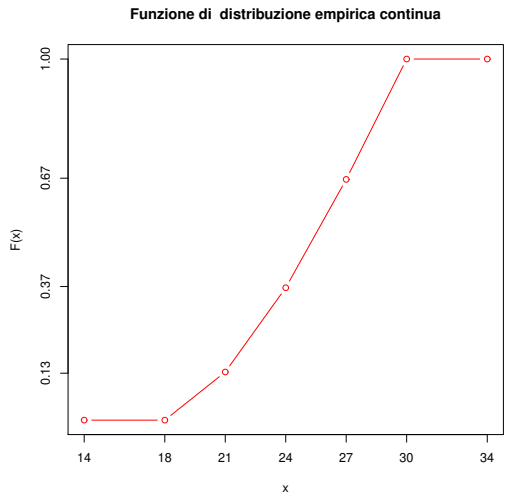

Figura 4.2: Grafico della funzione di distribuzione empirica continua del vettore voti utilizzando le classi $[18, 21)$, $[21, 24)$, $[24, 27)$, $[27, 30]$.

4.3 Indici di sintesi

Alcuni *indici di sintesi*, detti anche *statistiche*, utili a descrivere dei dati numerici, sono *media*, *mediana*, *moda*, *varianza*, *deviazione standard* e *coefficiente di variazione*. La media, la mediana e la moda sono *misure di centralità*, mentre la varianza e la deviazione standard misurano la *dispersione dei dati*.

Ad esempio, consideriamo quattro differenti campioni di numerosità $n = 30$ costituiti da numeri interi i, con $1 \leq i \leq 7$ rappresentati in Tabella 4.3

Tabella 4.3: Quattro differenti campioni di ampiezza 30.

dati1	1	1	2	2	2	3	3	3	3	3	4	4	4	4	4	4	4	4	5	5	5	5	5	5	6	6	6	7	7
dati2	1	1	1	2	2	3	3	3	3	4	4	4	4	4	4	4	4	4	4	4	5	5	5	5	6	6	6	7	7
dati3	1	1	2	2	2	2	2	2	2	2	3	3	3	3	3	3	4	4	4	4	4	5	5	5	5	6	6	6	7
dati4	1	1	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2	3	3	3	3	3	3	4	4	4	4	6	6

Le seguenti linee di codice

```
> dati1<-c(1,1,2,2,2,3,3,3,3,3,3,4,4,4,4,
+ 4,4,4,4,5,5,5,5,5,5,6,6,6,7,7)
>
> dati2<-c(1,1,1,2,2,3,3,3,3,4,4,4,4,4,4,
+ 4,4,4,4,4,4,5,5,5,5,6,6,6,7,7)
>
> dati3<- c(1,1,2,2,2,2,2,2,2,2,2,3,3,3,3,
+ 3,3,4,4,4,4,4,5,5,5,5,6,6,6,7)
>
> dati4<-c(1,1,1,1,1,1,1,1,1,1,2,2,2,2,2,
```

```
+ 2,2,2,3,3,3,3,3,3,4,4,4,4,6,6)
>
> table(dati1)
dati1
1 2 3 4 5 6 7
2 3 6 8 6 3 2
> table(dati2)
dati2
 1  2  3  4  5  6  7
 3  2  4 12  4  3  2
> table(dati3)
dati3
1 2 3 4 5 6 7
2 9 6 5 4 3 1
> table(dati4)
dati4
 1  2  3  4  6
10  8  6  4  2
>
> par(mfrow=c(2,2))
> plot(table(dati1), col="red")
> plot(table(dati2),col="blue")
> plot(table(dati3),col="brown")
> plot(table(dati4),col="magenta")
```

producono il grafico illustrato in Figura 4.3 che rappresenta le distribuzioni di frequenze per quattro differenti campioni ognuno costituito da 30 numeri interi i, con $1 \leq i \leq 7$.

Il primo grafico mostra una distribuzione delle frequenze simmetrica, centrata intorno a 4, valore per il quale la frequenza è massima; il secondo grafico visualizza ancora una distribuzione di frequenze centrata intorno al valore 4 ma non simmetrica, il terzo grafico si riferisce ancora ad una distribuzione delle frequenze non simmetrica, con una coda a destra più lunga rispetto a quella di sinistra; la quarta distribuzione delle frequenze è invece decrescente e non simmetrica, con alcuni valori dispersi.

Le seguenti linee di codice permettono di confrontare i boxplot dei quattro differenti vettori di dati1, dati2, dati3 e dati4.

```
> par(mfrow=c(2,2))
> boxplot(dati1,horizontal=TRUE,col="red",main="Boxplot of dati1")
> boxplot(dati2,horizontal=TRUE,col="blue",main="Boxplot of dati2")
> boxplot(dati3,horizontal=TRUE,col="brown",main="Boxplot of dati3"
    )
> boxplot(dati4,horizontal=TRUE,col="magenta",main="Boxplot of
    dati4")
```

che producono il grafico di Figura 4.4.

Analizziamo i boxplot di Figura 4.4. Si nota che i boxplot di dati1 e di dati2 sono identici anche se i due vettori sono differenti. Invece i boxplot di dati3 e di dati4 rivelano la presenza di asimmetria nei dati. Per il primo e secondo boxplot si ha $Q_1 - 1.5(Q_3 - Q_1) = 0$ e $Q_3 + 1.5(Q_3 - Q_1) = 8$, da cui segue che il baffo inferiore è posto in 1 (minimo) e il baffo superiore in 7 (massimo). Per il terzo boxplot si ha invece $Q_1 - 1.5(Q_3 - Q_1) = -2.125$ e $Q_3 + 1.5(Q_3 - Q_1) = 8.875$,

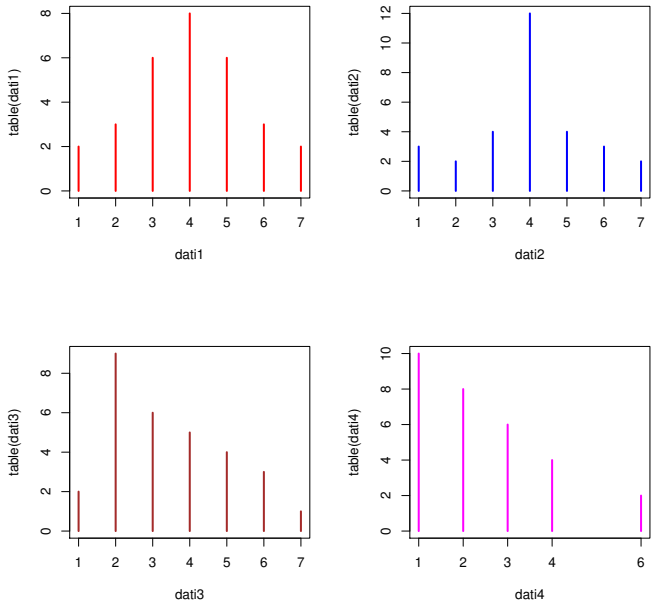

Figura 4.3: Distribuzione delle frequenze assolute di dati1, dati2, dati3 e dati4.

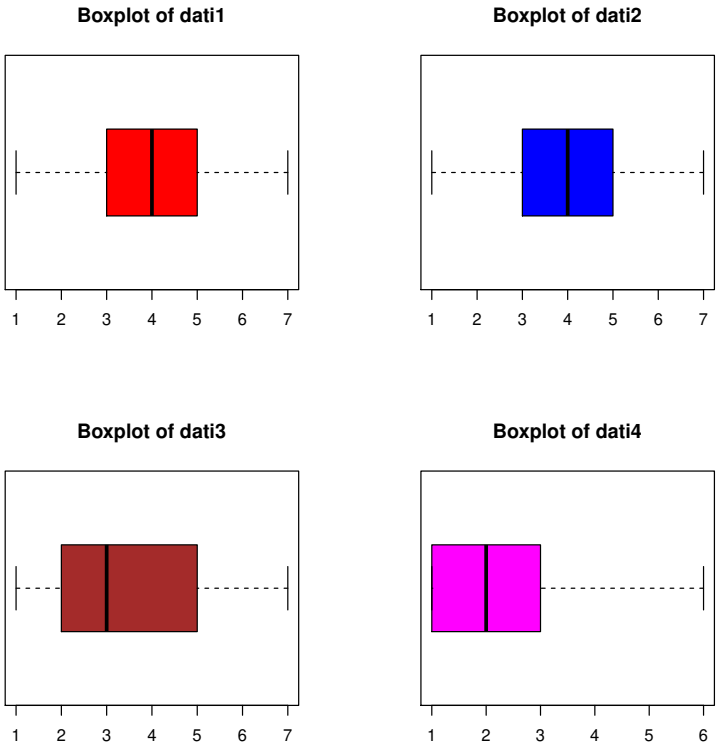

Figura 4.4: Boxplot dei vettori dati1, dati2, dati3 e dati4.

da cui occorre porre il baffo inferiore in 1 (minimo) e il baffo superiore in 7 (massimo). Infine, per il quarto boxplot si ha invece $Q_1 - 1.5(Q_3 - Q_1) = -2$ e $Q_3 + 1.5(Q_3 - Q_1) = 6$, implicando che il baffo inferiore è posto in 1 (minimo e primo quartile) e il baffo superiore in 6 (massimo).

```
> summary(dati1)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
      1       3       4       4       5       7
> summary(dati2)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1.000   3.000   4.000   3.967   5.000   7.000
> summary(dati3)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1.000   2.000   3.000   3.433   4.750   7.000
> summary(dati4)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    1.0     1.0     2.0     2.4     3.0     6.0
```

Gli indici che introdurremo nel seguito servono a misurare quantitativamente alcune delle caratteristiche osservate qualitativamente nei grafici delle distribuzioni di frequenze e nei boxplot, come in Figura 4.3 e Figura 4.4.

4.4 Media, mediana e moda campionarie

Supponiamo di avere un insieme $x_1, x_2, \ldots, x_n$ di n valori numerici (dati statistici quantitativi), detto *campione di ampiezza* o numerosità pari a n. La *media campionaria* è la media aritmetica di questi valori.

Definizione 4.1 *Si definisce media campionaria e si denota con $\overline{x}$, la quantità:*

$$\overline{x} = \frac{1}{n} \sum_{i=1}^{n} x_i \tag{4.4}$$

Si nota che se si considera un nuovo insieme di dati $y_i = a\,x_i + b$ $(i = 1, 2, \ldots, n)$ con $a, b \in \mathbb{R}$, allora la media campionaria di $y_1, y_2, \ldots, y_n$ è legata alla media campionaria dei dati iniziali $x_1, x_2, \ldots, x_n$ dalla stessa *relazione lineare*, ossia

$$\overline{y} = \frac{1}{n} \sum_{i=1}^{n} y_i = \frac{1}{n} \sum_{i=1}^{n} (a\,x_i + b) = a\,\overline{x} + b. \tag{4.5}$$

Se si denotano con $z_1, z_2, \ldots, z_k$ i valori distinti assunti dai dati, con $n_1, n_2, \ldots, n_k$ le frequenze assolute e con $f_1, f_2, \ldots, f_k$ le frequenze relative, allora la (4.4) può essere così riscritta:

$$\overline{x} = \frac{1}{n} \sum_{i=1}^{n} x_i = \frac{1}{n} \sum_{i=1}^{k} z_i\, n_i = \sum_{i=1}^{k} \frac{n_i}{n}\, z_i = \sum_{i=1}^{k} f_i\, z_i, \tag{4.6}$$

che mostra che *la media campionaria è una media pesata dei valori distinti assunti dai dati*. Ogni valore distinto usa come peso la sua frequenza relativa, ovvero la frazione dei dati uguale a tale valore numerico.

Per ogni valore x_i si definisce lo *scarto dalla media campionaria* la quantità

$$s_i = x_i - \overline{x} \qquad (i = 1, 2, \ldots, n),$$

che indica il grado di scostamento del singolo valore x_i dalla media campionaria $\overline{x}$. Si nota immediatamente che *la somma algebrica degli scarti dalla media campionaria è sempre nulla*. Infatti, risulta:

$$\sum_{i=1}^{n} s_i = \sum_{i=1}^{n} (x_i - \overline{x}) = \sum_{i=1}^{n} x_i - n\,\overline{x} = n\,(\overline{x} - \overline{x}) = 0.$$

Una seconda statistica che indica la centralità di un insieme di dati è la *mediana campionaria*.

Definizione 4.2 *Assegnato un insieme di dati di ampiezza n, lo si ordini in ordine crescente (dal valore più piccolo al valore più grande). Se n è dispari, si definisce mediana campionaria il valore che è in posizione $(n+1)/2$, mentre se n è pari la mediana campionaria è invece definita come la media aritmetica dei valori che occupano le posizioni $n/2$ e $n/2+1$.*

Questa definizione della mediana campionaria bipartisce le osservazioni (dopo aver effettuato l'ordinamento) in due gruppi di uguale numerosità, in maniera tale che *lo stesso numero di valori cada sia a sinistra che a destra della mediana stessa*. Ad esempio, la mediana campionaria di un campione ordinato di ampiezza tre è il valore intermedio, mentre per un campione ordinato di ampiezza quattro la mediana campionaria è la media aritmetica dei due valori intermedi.

Facendo riferimento alla Tabella 4.3 si nota che gli $n = 30$ valori sono già ordinati dal minore al maggiore e la mediana campionaria è la media aritmetica dei valori che occupano le posizioni $n/2 = 15$ e $n/2 + 1 = 16$. Si nota che la mediana del primo e del secondo campione è $(4 + 4)/2 = 4$, del terzo campione è $(3 + 3)/2 = 3$ e del quarto campione è $(2 + 2)/2 = 2$.

Media campionaria e mediana campionaria sono entrambe statistiche utili per descrivere misure di centralità dei dati. La media campionaria utilizza tutti i dati ed è influenzata in maniera sensibile da valori eccezionalmente alti o bassi. La mediana campionaria invece dipende solo da uno o da due valori centrali dei dati e non risente dei valori estremi. Inoltre, l'uso della mediana come indice per descrivere le caratteristiche dei dati ha lo svantaggio di dover prima riordinare i dati in ordine crescente, il che non è richiesto per il calcolo della media.

La terza statistica utilizzata per descrivere la centralità di una distribuzione di dati è la *moda campionaria*.

Definizione 4.3 *La moda campionaria di un insieme di dati, se esiste, è la modalità a cui è associata la frequenza (assoluta o relativa) più elevata. Se esistono più modalità con frequenza massima, ciascuna di esse è detto valore modale.*

In altre parole, la moda rappresenta il valore prevalente nell'insieme dei dati, ovvero quello che si presenta con maggiore frequenza (assoluta o relativa). La

moda campionaria è anche utilizzata quando si trattano *dati di tipo qualitativo*, per i quali non è possibile calcolare media e mediana. La moda campionaria può non esistere e non essere unica; quando è unica la distribuzione è detta *unimodale*, quando ci sono due o più mode diverse è detta *bimodale* o *multimodale* (*plurimodale*). In una distribuzione di frequenze *non esiste la moda campionaria* se nessuna modalità ha una frequenza superiore alle altre.

La moda può ritenersi un buon indice di sintesi quando si presenta con una frequenza nettamente maggiore rispetto alle frequenze delle altre modalità. La moda campionaria può non essere utile quando i dati sono numerosi e per la maggior parte diversi tra loro. Inoltre, se tutte le modalità presentano all'incirca la stessa frequenza, la moda non è un indice di sintesi significativo.

La moda campionaria è una statistica molto semplice e intuitiva per riassumere una distribuzione di frequenze. Anche se, come la mediana, non considera il peso delle singole osservazioni, ha alcune proprietà importanti:

→ è possibile identificare la moda per qualsiasi tipo di variabile, quindi anche nelle variabili qualitative ordinabili e non ordinabili;

→ indica sempre un valore realmente osservato nel campione;

→ non è influenzata dai valori estremi;

→ nel caso di distribuzioni di frequenze molto asimmetriche, la moda è il miglior indice per descrivere la tendenza centrale di un campione.

Per variabili quantitative continue raggruppate in classi si può invece definire la *classe modale* (o le classi modali). Per determinare la classe modale (o le classi modali) è opportuno ricorrere all'istogramma, individuando l'intervallo (o gli intervalli) di altezza massima.

Definizione 4.4 *La classe modale, se esiste, è la classe (l'intervallo) con la massima densità di frequenza (ossia la classe con la massima altezza nell'istogramma delle frequenze assolute o relative).*

Media campionaria, mediana campionaria e moda campionaria sono detti *indici di posizione centrali* o *indici di tendenza centrale* poiché descrivono attorno a quali valori è centrato l'insieme dei dati. La mediana è preferibile alla media quando si desidera eliminare gli effetti di valori estremi molto diversi dagli altri dati (valori anomali) poiché la mediana non utilizza tutti i dati, ma solo il valore centrale o i due valori centrali. Tuttavia occorre sottolineare che l'utilizzazione dei soli dati centrali rende la mediana poco sensibile a tutti gli altri valori dei dati e questo può costituire un limite per questo indice.

Il sistema R mette a disposizione le funzioni mean() e median() per calcolare rispettivamente la media e la mediana di un insieme di dati. Non esiste invece in R una funzione per estrarre la moda o la classe modale di una distribuzione di dati poichè è facilmente ricavabile osservando il grafico delle frequenze assolute o l'istogramma delle frequenze relative.

Riferendoci ad esempio ai vettori dati1, dati2, dati3 e dati4, il seguente codice R

```
> mean(dati1)
[1] 4
> median(dati1)
[1] 4
> mean(dati2)
[1] 3.966667
> median(dati2)
[1] 4
> mean(dati3)
[1] 3.433333
> median(dati3)
[1] 3
> mean(dati4)
[1] 2.4
> median(dati4)
[1] 2
```

e la Figura 4.3 della distribuzione delle frequenze assolute mostrano che per i dati del vettore dati1, media campionaria, mediana campionaria e moda campionaria sono uguali a 4; per i dati del vettore dati2, mediana campionaria e moda campionaria sono uguali a 4 mentre la media campionaria è 3.966667; invece per i dati del vettore dati3 la media campionaria è 3.433333, la mediana campionaria è 3 e la moda campionaria è 2; infine per per i dati del vettore dati4 la media campionaria è 2.4, la mediana campionaria è 2 e la moda campionaria è 1.

Per descrivere la forma di una distribuzione si può confrontare la media campionaria e la mediana campionaria. Se queste *due misure sono uguali* la distribuzione di frequenze tende ad essere simmetrica; se la *media campionaria è sensibilmente maggiore della mediana campionaria*, la distribuzione di frequenze è più sbilanciata verso destra (così come accade per il vettore dati3 e dati4); se invece la *media campionaria è sensibilmente minore mediana campionaria* la distribuzione di frequenze è più sbilanciata verso sinistra.

4.4.1 Mediana per una distribuzione di frequenze

Un modo di procedere diverso per definire la mediana consiste nel considerare le frequenze relative cumulative. Sia X una variabile quantitativa e siano $z_1, z_2, \ldots, z_k$ le modalità distinte da essa assunte, con $z_1 < z_2 < \ldots < z_k$. Considerato un campione $(x_1, x_2, \ldots, x_n)$, siano

$$F_i = f_1 + f_2 + \ldots + f_i \qquad (i = 1, 2, \ldots, k)$$

le frequenze relative cumulative.

Definizione 4.5 *La mediana per una distribuzione di frequenze è definita come la modalità i-esima (i=1,2,. . . , k) che soddisfa la doppia disuguaglianza:*

$$F_{i-1} < 0.5, \qquad F_i \geq 0.5.$$

Come si evince dalla definizione, la mediana di una distribuzione di frequenze è un valore di sintesi che indica un punto centrale intorno al quale si dispone

la distribuzione di frequenze. La mediana per una distribuzione di frequenze indica sempre un *valore realmente osservato nel campione*.

Riferendoci ad esempio ai vettori dati1, dati2, dati3 e dati4, le frequenze relative cumulative sono:

```
> Fdati1<-cumsum(table(dati1))/length(dati1)
> round(Fdati1,2)
   1    2    3    4    5    6    7
0.07 0.17 0.37 0.63 0.83 0.93 1.00
> Fdati2<-cumsum(table(dati2))/length(dati2)
> round(Fdati2,2)
   1    2    3    4    5    6    7
0.10 0.17 0.30 0.70 0.83 0.93 1.00
> Fdati3<-cumsum(table(dati3))/length(dati3)
> round(Fdati3,2)
   1    2    3    4    5    6    7
0.07 0.37 0.57 0.73 0.87 0.97 1.00
> Fdati4<-cumsum(table(dati4))/length(dati4)
> round(Fdati4,2)
   1    2    3    4    6
0.33 0.60 0.80 0.93 1.00
```

Utilizzando la Definizione 4.5 segue che la mediana per la distribuzione di frequenze per i dati1 e per dati2 è 4, per dati3 è 3 e per dati4 è 2. Come vedremo nel seguito, la funzione

$$\mathsf{quantile}(\mathsf{v}, 0.5, \mathsf{type} = 1)$$

calcola la mediana per la distribuzione di frequenze di un vettore v.

La mediana di una distribuzione di frequenze può essere individuata graficamente a partire dalla funzione di distribuzione empirica discreta. Si traccia la funzione di distribuzione empirica e sull'asse delle ordinate si individua il punto 0.5 e da questo si traccia una linea orizzontale. Il minimo valore osservato (sulle ascisse) la cui funzione di distribuzione empirica supera 0.5 è proprio la mediana per una distribuzione di frequenze. Riferendoci ad esempio ai vettori dati1, dati2, dati3 e dati4, le seguenti linee di codice

```
> par(mfrow=c(2,2))
> plot(ecdf(dati1), main="Funzione di distribuzione empirica\n
    discreta di dati1",verticals=TRUE, col="red")
> abline(h=0.5, lty=2,col="blue")
> plot(ecdf(dati2), main="Funzione di distribuzione empirica\n
    discreta di dati2",verticals=TRUE, col="red")
> abline(h=0.5, lty=2,col="blue")
> plot(ecdf(dati3), main="Funzione di distribuzione empirica\n
    discreta di dati 3",verticals=TRUE, col="red")
> abline(h=0.5, lty=2,col="blue")
> plot(ecdf(dati4), main="Funzione di distribuzione empirica\n
    discreta di dati4",verticals=TRUE, col="red")
> abline(h=0.5, lty=2,col="blue")
```

producono il grafico in Figura 4.5 che evidenzia i valori delle mediane per le distribuzioni di frequenze.

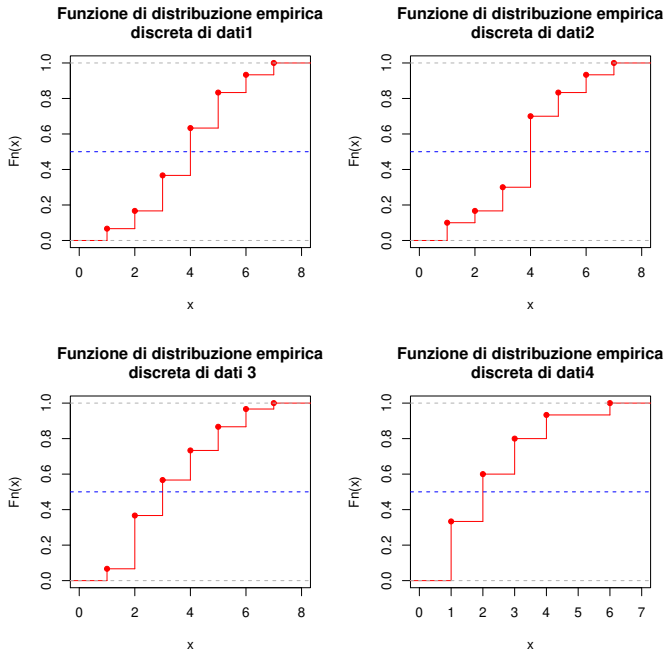

Figura 4.5: Mediana per la distribuzione di frequenze dei vettori dati1, dati2, dati3 e dati4.

4.5 Quantili, percentili, decili e quartili

Oltre la mediana, che è quel valore che divide a metà un insieme di dati ordinati, si possono definire altri indici di posizione, detti *quantili*, che dividono l'insieme dei dati ordinati in un fissato numero di parti uguali.

Sia X una variabile quantitativa e sia $x_1, x_2, \ldots, x_n$ un campione di n osservazioni disposte in *ordine crescente*. Supponiamo di suddividere i dati ordinati in α *gruppi*, ognuno dei quali contenga (circa) lo stesso numero di osservazioni; gli $\alpha - 1$ numeri che consentono tale suddivisione sono i *quantili* di ordine α. Ad esempio, possiamo suddividere i dati in $\alpha = 4$ parti mediante 3 quantili (detti *quartili*), oppure in $\alpha = 10$ parti mediante 9 quantili (detti *decili*) oppure anche in $\alpha = 100$ parti mediante 99 quantili (detti *percentili*).

I quantili (percentili) sono *indici di posizione non centrali* utilizzati per insiemi numerosi di dati. I *decili* sono un caso particolare dei percentili e si ottengono dividendo l'insieme dei dati ordinati in dieci parti uguali; il decile D_j $(j = 1, 2, \ldots, 9)$ corrisponde al $(j * 10)$-esimo percentile. I *quartili* sono un caso particolare dei percentili e si ottengono invece dividendo l'insieme dei dati ordinati in quattro parti uguali; il quartile Q_j $(j = 1, 2, 3)$ corrisponde al $(j * 25)$-esimo percentile.

In R esistono *9 differenti algoritmi*[1] per calcolare i quantili ottenibili utilizzando la funzione

$$\mathsf{quantile}(\mathsf{v}, \mathsf{probs} =, \mathsf{type} = \mathsf{j})$$

dove v è un vettore numerico, probs è il vettore delle probabilità e $j = 1, 2, \ldots, 9$ denota il tipo di algoritmo selezionato. R utilizza di default per il calcolo dei quantili l'algoritmo di tipo 7, basato su tecniche di interpolazione tra i punti. Occorre osservare che se il numero di osservazioni è elevato i valori dei quantili tendono a coincidere qualsiasi sia il tipo di algoritmo scelto.

I percentili che vengono maggiormente utilizzati sono il 25-*esimo*, il 50-*esimo* e il 75-*esimo*, detti rispettivamente primo quartile (Q_1), il secondo quartile (Q_2) e il terzo quartile (Q_3).

Se si omette probs e si scrive

$$\mathsf{quantile}(\mathsf{v}, \mathsf{type} = \mathsf{j}) \qquad (j = 1, 2, \ldots, 9)$$

vengono di default calcolati i quartili con type = j e la funzione restituisce il minimo, il massimo e i tre quartili Q_1, Q_2 e Q_3. Se si omette type e si scrive

$$\mathsf{quantile}(\mathsf{v}, \mathsf{probs} =)$$

vengono di default calcolati i percentili specificati nel vettore delle probabilità utilizzando di default l'algoritmo di tipo 7.

Nel seguito per il calcolo dei quantili descriveremo l'algoritmo di tipo 2 (che generalizza la definizione di mediana campionaria), l'algoritmo di tipo 7 (di default in R, basato su tecniche di interpolazione tra punti) e l'algoritmo di tipo 1 (che generalizza la definizione di mediana per una distribuzione di frequenze).

[1] Rob J. Hyndman e Yanan Fan (1996) Sample Quantiles in Statistical Packages, The American Statistician, 50, 361–365.

4.5.1 Quantili con l'algoritmo di tipo 2

Per calcolare il percentile k-esimo ($k = 0, 1, \ldots, 100$) P_k con l'algoritmo di tipo 2 si utilizza la seguente procedura.

Algoritmo (*type* = 2): Calcolo del percentile P_k
STEP 1: Ordinare i dati del campione di ampiezza n in ordine crescente (dal valore più piccolo al valore più grande) e sia v *il vettore ordinato;*
STEP 2: Calcolare l'indice h

$$h = n\,p = n\,\frac{k}{100},$$

in cui P_k è il percentile di interesse e n è il numero di osservazioni (ampiezza del campione);
STEP 3:
→ Se $h = n\,p$ è un intero, il percentile k-esimo si ottiene effettuando la media aritmetica dei valori nelle posizioni h e $h+1$ nell'insieme dei dati ordinati, ossia $P_k = (\mathsf{v}[h] + \mathsf{v}[h+1])/2$;
→ Se $h = n\,p$ non è un intero, si arrotonda $h = n\,p$ per eccesso al primo intero successivo ottenendo

$$h^* = ceiling(h).$$

Il percentile k-esimo è quello che corrisponde alla posizione h^*, ossia $P_k = \mathsf{v}[h^*]$.

Questo algoritmo *generalizza il concetto di mediana*, ottenibile ponendo $p = 0.5$ e $k = 50$. Infatti, in questo caso se n è pari ($h = n/2$ è intero) si effettua la media aritmetica dei valori in posizione $n/2$ e $n/2 + 1$, mentre se n è dispari ($h = n/2$ non è intero) si arrotonda $h = n/2$ per eccesso e la mediana corrisponde al valore in posizione $(n + 1)/2$.

L'algoritmo è implementato in R scegliendo $j = 2$ e usando la funzione

$$\mathsf{quantile}(\mathsf{v}, \mathsf{probs} =, \mathsf{type} = 2)$$

Ad esempio,

```
> quantile(dati1,c(0,0.25,0.5,0.75,1), type=2)
  0%  25%  50%  75% 100%
   1    3    4    5    7
>
> quantile(dati2,c(0,0.25,0.5,0.75,1), type=2)
  0%  25%  50%  75% 100%
   1    3    4    5    7
>
> quantile(dati3,c(0,0.25,0.5,0.75,1), type=2)
  0%  25%  50%  75% 100%
   1    2    3    5    7
>
> quantile(dati4,c(0,0.25,0.5,0.75,1), type=2)
  0%  25%  50%  75% 100%
   1    1    2    3    6
```

Ad esempio, se siamo interessati al terzo quartile Q_3 del vettore ordinato dati3, osserviamo che $n = 30$, $k = 75$, $p = 0.75$; applicando l'Algoritmo si ha $h = np = 22.5$ non è intero e quindi $Q_3 = P_{75} = \mathsf{dati3}[23] = 5$.

4.5.2 Quantili con l'algoritmo di tipo 7

R utilizza di default per il calcolo dei quantili l'algoritmo di tipo 7, basato su una tecnica lineare di interpolazione tra i punti. Per calcolare il percentile k-esimo ($k = 0, 1, \ldots, 100$) con type = 7 si utilizza il seguente algoritmo.

Algoritmo (*type* = 7): Calcolo del percentile P_k
STEP 1: Ordinare i dati del campione di ampiezza n in ordine crescente (dal valore più piccolo al valore più grande) e sia v il vettore ordinato;
STEP 2: Calcolare l'indice h

$$h = (n-1)\,p + 1 = (n-1)\,\frac{k}{100} + 1,$$

in cui P_k è il percentile di interesse e n è il numero di osservazioni (ampiezza del campione);
STEP 3: Calcolare il più grande intero minore o uguale di h (si arrotonda per difetto):

$$h^* = floor(h)$$

STEP 4:
→ Il percentile k-esimo si ottiene calcolando

$$P_k = v[h^*] + (h - h^*) \cdot \{v[h^* + 1] - v[h^*]\}.$$

Si nota che tale algoritmo si basa su una tecnica di interpolazione di punti. Infatti, l'equazione della retta che passa per i punti $(h^*, v[h^*])$ e $(h^*+1, v[h^*+1])$ è:

$$\frac{y - v[h^*]}{x - h^*} = \frac{v[h^* + 1] - v[h^*]}{(h^* + 1) - h^*}.$$

Imponendo che il punto $(x, y) = (h, P_k)$ sia sulla retta, il percentile P_k è calcolato come mostrato nello *STEP 4* dell'algoritmo.

Si può facilmente mostrare che per $p = 0.5$ e $k = 50$, l'algoritmo di tipo 2 e l'algoritmo di tipo 7 conducono allo stesso risultato nel calcolo della mediana campionaria. Infatti, applicando l'algoritmo di tipo 7 con $p = 0.5$ e $k = 50$ risulta $h = (n + 1)/2$; quindi, se n è pari si ha $h^* = n/2$ e la mediana è $P_{50} = (v[n/2] + v[n/2 + 1])/2$, mentre se n è dispari si ha $h^* = (n + 1)/2$ e la mediana è $P_{50} = v[(n + 1)/2]$.

L'istruzione quantile(v, probs =, type = 7) è equivalente a quantile(v, probs =) in cui, per default, viene implementato l'algoritmo di tipo 7. Se si omette probs e type e e si scrive quantile(v) la funzione restituisce il minimo, il massimo e i tre quartili Q_1, Q_2 e Q_3 utilizzando di default l'algoritmo di tipo 7.

Relativamente ai vettori dati1, dati2, dati3 e dati4, si ottiene:

```
> quantile(dati1)
  0%  25%  50%  75% 100%
   1    3    4    5    7
>
> quantile(dati2)
```

```
  0%  25%  50%  75% 100%
   1    3    4    5    7
>
> quantile(dati3)
  0%  25%  50%  75% 100%
1.00 2.00 3.00 4.75 7.00
>
 > quantile(dati4)
  0%  25%  50%  75% 100%
   1    1    2    3    6
```

Ad esempio, se siamo interessati al terzo quartile Q_3 del vettore ordinato dati3, osserviamo che $n = 30$, $k = 75$, $p = 0.75$; applicando l'Algoritmo si ha $(n-1)\,p+1 = 22.75$ e $h^* = 22$ da cui $Q_3 = P_{75} = \text{dati3}[22] + 0.75 * (\text{dati3}[23] - \text{dati3}[22]) = 4.75$. La seguente linea di codice

```
> boxplot(dati1,dati2,dati3,dati4,
+ names=c("dati1","dati2","dati3","dati4"),
+ col=c("red","orange","green","yellow"))
```

produce il grafico illustrato in Figura 4.6 in cui sono confrontati nella stessa finestra grafica i boxplot dei quattro vettori dati1, dati2, dati3 e dati4.

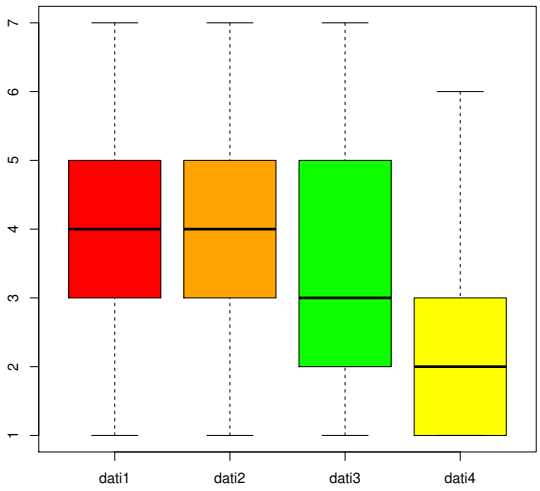

Figura 4.6: Confronto tra i boxplot dei vettori dati1, dati2, dati3 e dati4.

La funzione summary(v) restituisce oltre al minimo, massimo e ai tre quartili Q_1, Q_2 e Q_3 (implementati con l'algoritmo di tipo 7) anche la media campionaria.

4.5.3 Quantili per una distribuzione di frequenze ($type = 1$)

Un modo di procedere diverso per definire i quantili consiste nel considerare le frequenze relative cumulative. Sia X una variabile quantitativa e siano $z_1, z_2, \ldots, z_k$ le modalità distinte da essa assunte, con $z_1 < z_2 < \ldots < z_k$. Considerato un campione $(x_1, x_2, \ldots, x_n)$, siano $F_1, F_2, \ldots, F_k$ le frequenze relative cumulative.

Definizione 4.6 *Assegnata una probabilità p, $0 < p < 1$, il quantile di ordine p, è definito come la modalità i-esima (i=1,2,..., k) che soddisfa la doppia disuguaglianza:*

$$F_{i-1} < p, \qquad F_i \geq p.$$

Si nota che i quantili di una distribuzione di frequenze forniscono sempre valori realmente osservati nel campione.

Il caso che si presenta più di frequente è quello dei quartili, che suddividono la distribuzione in quattro parti. Il *primo quartile* di una distribuzione di frequenze è definita come la modalità i-esima (i=1,2,..., k) che soddisfa la doppia disuguaglianza:

$$F_{i-1} < 0.25, \qquad F_i \geq 0.25.$$

Il *secondo quartile* (mediana) di una distribuzione di frequenze è definita come la modalità i-esima (i=1,2,..., k) che soddisfa la doppia disuguaglianza:

$$F_{i-1} < 0.5, \qquad F_i \geq 0.5.$$

Il *terzo quartile* di una distribuzione di frequenze è definita come la modalità i-esima (i=1,2,..., k) che soddisfa la doppia disuguaglianza:

$$F_{i-1} < 0.75, \qquad F_i \geq 0.75.$$

L'algoritmo della Definizione 4.6 è implementato in R scegliendo $j = 1$ e usando la funzione

$$\mathsf{quantile}(\mathsf{v}, \mathsf{prob} =, \mathsf{type} = 1)$$

I quartili per una distribuzione di frequenze possono essere visualizzati graficamente a partire dalla funzione di distribuzione empirica discreta. Si traccia la funzione di distribuzione empirica e sull'asse delle ordinate si individuano i punti 0.25, 0.5, 0.75 e da questi si tracciano delle linee orizzontali:

- il minimo valore osservato la cui funzione di distribuzione empirica supera 0.25 è il primo quartile Q_1 per una distribuzione di frequenze;

- il minimo valore osservato la cui funzione di distribuzione empirica supera 0.5 è il secondo quartile Q_2 (mediana) per una distribuzione di frequenze;

- il minimo valore osservato la cui funzione di distribuzione empirica supera 0.75 è il terzo quartile Q_3 per una distribuzione di frequenze.

Ad esempio, le seguenti linee di codice

```
>  quantile(dati1,type=1)
  0%  25%  50%  75% 100%
   1    3    4    5    7
> quantile(dati2,type=1)
  0%  25%  50%  75% 100%
   1    3    4    5    7
> quantile(dati3,type=1)
  0%  25%  50%  75% 100%
   1    2    3    5    7
> quantile(dati4,type=1)
  0%  25%  50%  75% 100%
   1    1    2    3    6
```

mostrano i quartili per i vettori dati1, dati2, dati3 e dati4 utilizzando l'algoritmo di tipo 1. Inoltre, le seguenti linee di codice

```
> par(mfrow=c(2,2))
> plot(ecdf(dati1), main="Funzione di distribuzione empirica\n
    discreta di dati1",verticals=TRUE, col="red")
> abline(h=0.25, lty=1,col="blue")
> abline(h=0.5, lty=2,col="blue")
> abline(h=0.75, lty=3,col="blue")
> plot(ecdf(dati2), main="Funzione di distribuzione empirica\n
    discreta di dati2",verticals=TRUE, col="red")
> abline(h=0.25, lty=1,col="blue")
> abline(h=0.5, lty=2,col="blue")
> abline(h=0.75, lty=3,col="blue")
> plot(ecdf(dati3), main="Funzione di distribuzione empirica\n
    discreta di dati 3",verticals=TRUE, col="red")
> abline(h=0.25, lty=1,col="blue")
> abline(h=0.5, lty=2,col="blue")
> abline(h=0.75, lty=3,col="blue")
> plot(ecdf(dati4), main="Funzione di distribuzione empirica\n
    discreta di dati4",verticals=TRUE, col="red")
> abline(h=0.25, lty=1,col="blue")
> abline(h=0.5, lty=2,col="blue")
> abline(h=0.75, lty=3,col="blue")
```

producono il grafico in Figura 4.7 che evidenzia i valori dei quartili per le distribuzioni di frequenze.

4.5.4 Quartili con i differenti algoritmi

La seguente funzione permette di calcolare i quartili Q_0 (minimo), Q_1 (primo quartile), Q_2 (secondo quartile), Q_3 (terzo quartile), Q_4 (massimo) per un vettore v di dati per i nove algoritmi:

```
> tipiquartili<-function(x){
+ y<-numeric(0)
+ for(i in 1:9){
+ y<-rbind(y,c(quantile(x,0,type=i),quantile(x,0.25,type=i),
+ quantile(x,0.5,type=i),quantile(x,0.75,type=i),
+ quantile(x,1,type=i)))}
+ rownames(y)<-paste("type",1:9)
+ y # visualizza y}
```

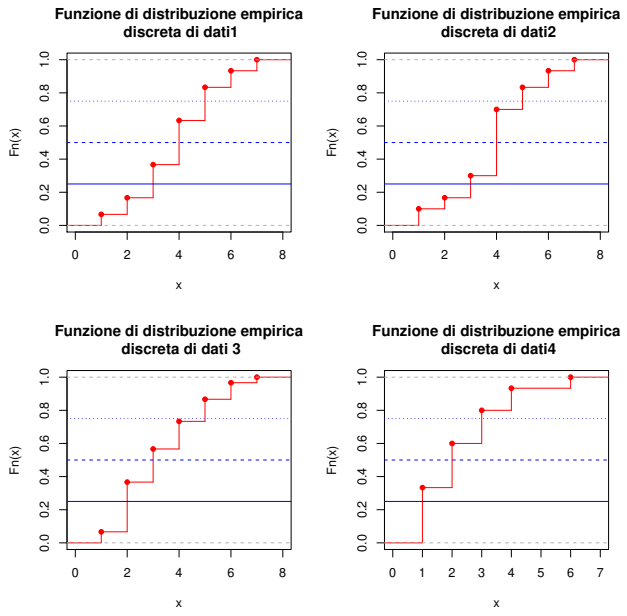

Figura 4.7: Quartili per la distribuzione di frequenze dei vettori dati1, dati2, dati3 e dati4.

Ad esempio, applicando tale funzione al vettore dati3 si ottiene:

```
> tipiquartili(dati3)
       0% 25% 50%  75% 100%
type 1  1    2    3 5.00    7
type 2  1    2    3 5.00    7
type 3  1    2    3 4.00    7
type 4  1    2    3 4.50    7
type 5  1    2    3 5.00    7
type 6  1    2    3 5.00    7
type 7  1    2    3 4.75    7
type 8  1    2    3 5.00    7
type 9  1    2    3 5.00    7
```

In questo caso, alcune differenze nei risultati dei diversi algoritmi si riscontrano soltanto nel calcolo del terzo quartile Q_3.

4.6 Varianza, deviazione standard e coefficiente di variazione

Gli indici di posizione non tengono conto della variabilità dei dati; infatti esistono distribuzioni di frequenze che sono molto diverse tra loro, pur avendo la stessa media campionaria. Ad esempio, consideriamo i quattro campioni di numerosità 20 elencati in Tabella 4.4.

Tabella 4.4: Quattro differenti campioni di ampiezza 20.

v1	60	60	60	60	60	60	60	60	60	60	60	60	60	60	60	60	60	60	60	60
v2	10	10	10	10	20	20	20	20	60	60	60	60	100	100	100	100	110	110	110	110
v3	10	10	10	10	20	20	20	20	65	65	65	65	100	100	100	100	105	105	105	105
v4	50	50	50	50	55	55	55	55	60	60	60	60	65	65	65	65	70	70	70	70

Le seguenti linee di codice

```
> v1<-c(60,60,60,60,60,60,60,60,60,60,
+ 60,60,60,60,60,60,60,60,60,60)
>
> v2<-c(10,10,10,10,20,20,20,20,60,60,
+ 60,60,100,100,100,100,110,110,110,110)
>
> v3<-c(10,10,10,10,20,20,20,20,65,65,
+ 65,65,100,100,100,100,105,105,105,105)
>
> v4<-c(50,50,50,50,55,55,55,55,60,60,
+ 60,60,65,65,65,65,70,70,70,70)
>
> table(v1)
v1
60
20
> table(v2)
v2
 10  20  60 100 110
  4   4   4   4   4
> table(v3)
v3
 10  20  65 100 105
  4   4   4   4   4
> table(v4)
v4
50 55 60 65 70
 4  4  4  4  4
>
> par(mfrow=c(2,2))
> plot(table(v1),col="red")
> plot(table(v2),col="blue")
> plot(table(v3),col="brown")
> plot(table(v4),col="magenta")
```

producono il grafico delle frequenze illustrato in Figura 4.8:

Applicando ora la funzione summary() ai vettori v1, v2, v3 e v4: si ha

```
> summary(v1)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
     60      60      60      60      60      60
>
> summary(v2)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
     10      20      60      60     100     110
>
> summary(v3)
```

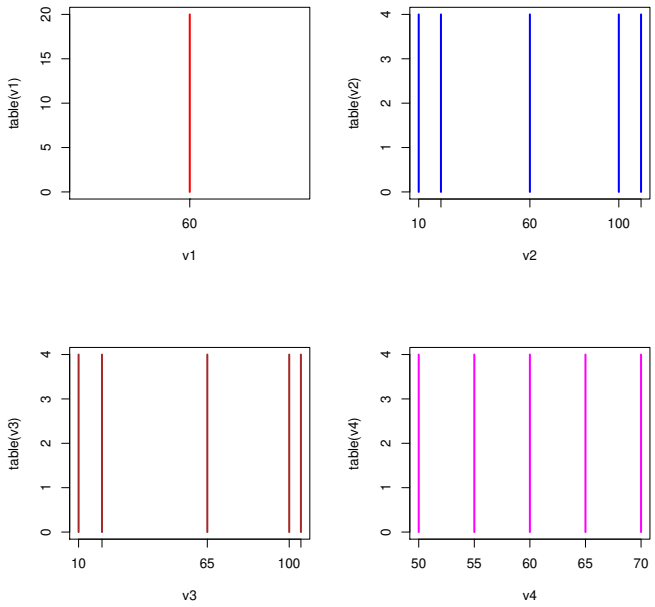

Figura 4.8: Grafico della distribuzione delle frequenze assolute dei vettori v1, v2, v3 e v4.

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
     10      20      65      60     100     105
>
> summary(v4)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
     50      55      60      60      65      70
```

Si nota che i vettori v1, v2, v3 e v4 sono tutti caratterizzati dalla stessa media campionaria, ma contengono insiemi di dati molto diversi; il primo vettore è composto da dati tutti uguali; la media campionaria e la mediana campionaria coincidono nel primo, secondo e quarto vettore, mentre sono differenti nel terzo vettore; inoltre, il secondo vettore presenta una maggiore variabilità tra il minimo e il massimo. La seguente linea di codice

```
> boxplot(v1,v2,v3,v4,
+ names=c("v1","v2","v3","v4"),
+ col=c("red","orange","green","yellow"))
```

produce il grafico illustrato in Figura 4.9 in cui sono confrontati i boxplot dei quattro vettori v1, v2, v3 e v4.

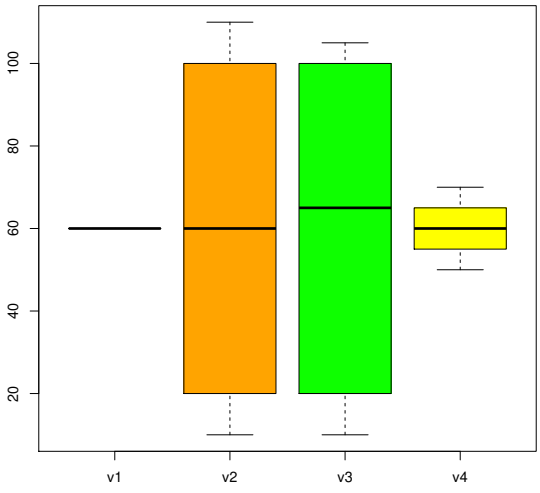

Figura 4.9: Confronto tra i boxplot dei vettori v1, v2, v3 e v4.

Indici significativi per misurare la variabilità di una distribuzione di frequenze sono la *varianza campionaria* e la *deviazione standard campionaria*, detta anche *scarto quadratico medio campionario*.

Definizione 4.7 *Assegnato un insieme di dati numerici $x_1, x_2, \ldots, x_n$, si definisce varianza campionaria, e si denota con s^2, la quantità:*

$$s^2 = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \overline{x})^2 \qquad (n = 2, 3, \ldots), \tag{4.7}$$

dove $\overline{x}$ denota la media campionaria dei dati. Inoltre, si definisce deviazione standard campionaria la radice quadrata della varianza campionaria, ossia:

$$s = \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} (x_i - \overline{x})^2} \qquad (n = 2, 3, \ldots). \tag{4.8}$$

Varianza campionaria e deviazione standard campionaria sono detti *indici di dispersione* o *indici di variabilità* poiché misurano la dispersione dei dati intorno alla media. Dalle (4.7) e (4.8) si nota che la varianza e la deviazione standard sono tanto più grandi quando più i dati si discostano dalla media. I valori della varianza campionaria s^2 e della deviazione standard campionaria s *dipendono dall'unità di misura dei dati*. In particolare, la deviazione standard campionaria s *misura la dispersione dei dati con la stessa unità di misura dei dati sperimentali* e quindi con la *stessa unità di misura della media campionaria*.

In R la varianza campionaria di un vettore numerico `v` si calcola utilizzando la funzione `var(v)` e la deviazione standard campionaria si calcola utilizzando la funzione `sd(v)`. Ad esempio, riferendoci ai vettori `v1`, `v2`, `v3` e `v4`, si nota che

```
> var(v1)
[1] 0
> sd(v1)
[1] 0
> var(v2)
[1] 1726.316
> sd(v2)
[1] 41.54896
> var(v3)
[1] 1631.579
> sd(v3)
[1] 40.39281
> var(v4)
[1] 52.63158
> sd(v4)
[1] 7.254763
```

che mostra che per il vettore `v1` la varianza è nulla poiché tutti i dati coincidono con la media campionaria, mentre per il vettore `v2` la varianza è più elevata poiché i dati si discostano maggiormente dalla media.

Per un insieme di dati numerici $x_1, x_2, \ldots, x_n$ la varianza campionaria può anche essere calcolata nel seguente modo:

$$s^2 = \frac{1}{n-1} \left[\sum_{i=1}^{n} x_i^2 - n\,\overline{x}^2 \right] = \frac{n}{n-1} \big(M_2 - \overline{x}^2 \big), \tag{4.9}$$

dove M_2 è il momento campionario del secondo ordine. Infatti, la (4.9) può essere così dimostrata:

$$\begin{aligned} s^2 &= \frac{1}{n-1}\sum_{i=1}^{n}(x_i-\overline{x})^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i^2-2\,x_i\overline{x}+\overline{x}^2) \\ &= \frac{1}{n-1}\left[\sum_{i=1}^{n}x_i^2-2\,\overline{x}\sum_{i=1}^{n}x_i+\sum_{i=1}^{n}\overline{x}^2\right] = \frac{1}{n-1}\left[\sum_{i=1}^{n}x_i^2-2\,n\,\overline{x}^2+n\,\overline{x}^2\right] \\ &= \frac{n}{n-1}\Big[\frac{1}{n}\sum_{i=1}^{n}x_i^2-\overline{x}^2\Big] = \frac{n}{n-1}\big(M_2-\overline{x}^2\big). \end{aligned}$$

Si nota inoltre che se si considera un nuovo insieme di dati $y_i = a\,x_i + b$ ($i = 1, 2, \ldots, n$), con $a, b \in \mathbb{R}$, allora la varianza campionaria s_y^2 di $y_1, y_2, \ldots, y_n$ è legata alla varianza campionaria s_x^2 dei dati iniziali $x_1, x_2, \ldots, x_n$ dalla relazione:

$$s_y^2 = a^2\; s_x^2. \tag{4.10}$$

Infatti, ricordando la proprietà (4.5) di linearità della media campionaria, dalla (4.7) segue che

$$\begin{aligned} s_y^2 &= \frac{1}{n-1}\sum_{i=1}^{n}(y_i-\overline{y})^2 = \frac{1}{n-1}\sum_{i=1}^{n}(a\,x_i+b-a\,\overline{x}-b)^2 \\ &= a^2\frac{1}{n-1}\sum_{i=1}^{n}(x_i-\overline{x})^2 = a^2\; s_x^2 \end{aligned}$$

Pertanto, sommare una costante a ciascuno dei dati non fa cambiare la varianza campionaria, mentre moltiplicare ciascuno dei dati per un fattore costante fa sì che la varianza campionaria dell'insieme iniziale dei dati risulta moltiplicata per il quadrato di tale fattore.

La media campionaria e la deviazione standard campionaria sono i due indici di posizione e di dispersione dei dati maggiormente utilizzati. Per confrontare le variazioni esistenti tra diversi campioni di dati è utile introdurre *coefficiente di variazione*.

Definizione 4.8 *Assegnato un insieme di dati numerici $x_1, x_2, \ldots, x_n$, si definisce coefficiente di variazione il rapporto tra la deviazione standard campionaria e il modulo della media campionaria, ossia:*

$$CV = \frac{s}{|\overline{x}|}. \tag{4.11}$$

Si nota che il coefficiente di variazione è un *numero puro*, ossia è un indice *indice adimensionale che non dipende dall'unità di misura utilizzata*, poiché la media campionaria e la deviazione standard campionaria sono espressi in identiche unità di misura. Dalla (4.11) segue che il coefficiente di variazione è un indice di dispersione che ha senso soltanto per campioni aventi la *media campionaria non*

nulla. Il coefficiente di variazione è utilizzato quando è necessario confrontare tra loro le dispersioni (variabilità) di insiemi di dati espressi in *differenti unità di misura* (peso, altezza, redditi, ...) oppure insiemi di dati aventi *differenti range di variazione* (il range di variazione è dato dalla differenza tra il massimo e il minimo dei dati). Ad esempio, la deviazione standard di un campione di redditi espressi in Dollari è completamente diversa della deviazione standard degli stessi redditi espressi in Euro, mentre il coefficiente di variazione è lo stesso in entrambi i casi.

In R non è definita una funzione che calcola il coefficiente di variazione. Tale funzione può essere comunque facilmente implementata in R nel seguente modo:

```
> cv <- function(x){
+ sd(x)/abs(mean(x))}
```

Riferendoci ai dati dei vettori v1, v2, v3 e v4, si nota che

```
> cv(v1)
[1] 0
> cv(v2)
[1] 0.6924826
> cv(v3)
[1] 0.6732135
> cv(v4)
[1] 0.1209127
```

In Tabella 4.5 sono riportati la media, la varianza, la deviazione standard e il coefficiente di variazione dei vettori v1, v2, v3 e v4.

Tabella 4.5: Media e indici di dispersione dei vettori v1, v2, v3 e v4.

	v1	v2	v3	v4
Media	60	60	60	60
Varianza	0	1726.316	1631.579	52.63158
Deviazione standard	0	41.54896	40.39281	7.254763
Coefficiente di variazione	0	0.6924826	0.6732135	0.1209127

Come si evince in Tabella 4.5 il coefficiente di variazione più alto si ottiene per il vettore v2 in cui i dati si discostano maggiormente dalla media campionaria, che è 60 per tutti i vettori; ciò si può anche intuire confrontando i boxplot di Figura 4.9.

Tabella 4.6: Media e indici di dispersione dei vettori dati1, dati2, dati3 e dati4.

	dati1	dati2	dati3	dati4
Media	4	3.966667	3.433333	2.4
Varianza	2.482759	2.516092	2.598851	2.041379
Deviazione standard	1.575677	1.586219	1.612095	1.428768
Coefficiente di variazione	0.3939193	0.3998872	0.4695423	0.5953202

Procedendo in modo analogo per i vettori dati1, dati2, dati3 e dati4 si ottengono i risultati riportati in Tabella 4.6. In questo caso la media campionaria è

differente per tutti i vettori e il coefficiente di variazione più alto si ottiene per il vettore dati4, come si può anche intuire confrontando i boxplot di Figura 4.6.

4.7 Momenti campionari e momenti centrati

Vogliamo ora definire i momenti campionari e i momenti centrati intorno alla media campionaria.

Definizione 4.9 *Assegnato un insieme di dati numerici $x_1, x_2, \ldots, x_n$, si definisce momento campionario di ordine j e si denota con M_j la quantità:*

$$M_j = \frac{1}{n}\sum_{i=1}^{n} x_i^j \qquad (j = 1, 2, \ldots). \tag{4.12}$$

Quando $j = 1$, il momento campionario di ordine 1 coincide con la media campionaria, ossia $M_1 = \overline{x}$.

Definizione 4.10 *Assegnato un insieme di dati numerici $x_1, x_2, \ldots, x_n$, si definisce momento campionario centrato intorno alla media di ordine j e si denota con m_j la quantità:*

$$m_j = \frac{1}{n}\sum_{i=1}^{n} (x_i - \overline{x})^j \qquad (j = 1, 2, \ldots). \tag{4.13}$$

Quando $j = 1$, il momento centrato di ordine 1 è nullo, ossia $m_1 = 0$; quando $j = 2$, il momento centrato di ordine 2 è connesso alla varianza campionaria tramite la relazione:

$$m_2 = \frac{1}{n}\sum_{i=1}^{n} (x_i - \overline{x})^2 = \frac{n-1}{n}\, s_x^2.$$

Inoltre, se consideriamo un campione $y_1, y_2, \ldots, y_n$, dove $y_i = a\, x_i + b \quad (a, b \in \mathbb{R})$, allora sussiste la relazione:

$$\widetilde{m}_j = \frac{1}{n}\sum_{i=1}^{n} (y_i - \overline{y})^j = \frac{1}{n}\sum_{i=1}^{n} (a\, x_i + b - a\overline{x} - b)^j = \frac{a^j}{n}\sum_{i=1}^{n} (x_i - \overline{x})^j = a^j\, m_j$$
$$(j = 1, 2, \ldots).$$

4.8 Forma di una distribuzione di frequenze

La media, la mediana e la moda sono utili a comprendere la forma delle distribuzioni di frequenze, nel senso che differenze sostanziali tra questi indici indicano uno *sbilanciamento eccessivo della distribuzione di frequenze verso destra o verso sinistra*. Abbiamo inoltre mostrato che se la media e la mediana coincidono, le misure di dispersione dei dati possono essere sostanzialmente differenti implicando una differente forma delle distribuzioni di frequenze. Esistono degli indici

statistici che permettono di misurare quando una distribuzione di frequenze presenta *simmetria o asimmetria* oppure se essa è *più o meno piccata*.

Un indice che permette di misurare la simmetria di una distribuzione di frequenze è la *skewness campionaria* (coefficiente di simmetria).

Definizione 4.11 *Assegnato un insieme di dati numerici $x_1, x_2, \ldots, x_n$, si definisce skewness campionaria il valore:*

$$\gamma_1 = \frac{m_3}{m_2^{3/2}} \tag{4.14}$$

dove m_3 denota il momento centrato campionario di ordine 3.

Dalla Definizione 4.11 si nota che se la distribuzione di frequenze è *simmetrica* γ_1 assume il valore 0, altrimenti $\gamma_1 > 0$ per l'*asimmetria positiva* (ossia la distribuzione di frequenze ha la coda di destra più allungata) e $\gamma_1 < 0$ per l'*asimmetria negativa* (ossia la distribuzione di frequenze ha la coda di sinistra più allungata). Si nota che γ_1 è un *indice adimensionale*, ossia è indipendente dall'unità di misura dei dati.

Occorre sottolineare che *ogni distribuzione di frequenze simmetrica ha una skewness nulla*, ma esistono anche distribuzioni di frequenze non simmetriche con skewness nulla. Inoltre, se $\gamma_1 = 0$ ciò non significa che mediana, media e moda coincidano (nel caso di una curva di frequenza unimodale).

Il calcolo della skewness campionaria può essere così implementato in R:

```
> skw<-function(x){
+ n<-length(x)
+ m2<-(n-1)*var(x)/n
+ m3<- (sum( (x-mean(x))^3))/n
+ m3/(m2^1.5)
+ }
```

Riferendoci ai dati dei vettori dati1, dati2, dati3 e dati4, si nota che

```
> skw(dati1)
[1] 0
> skw(dati2)
[1] -0.1028353
> skw(dati3)
[1] 0.4736654
> skw(dati4)
[1] 0.9985399
```

che mostra che la skewness è nulla per il vettore dati1, presenta un'asimmetria negativa per il vettore dati2 e presenta un'asimmetria positiva per i rimanenti due vettori.

Un indice che permette di misurare la densità dei dati intorno alla media è la *curtosi campionaria*.

Definizione 4.12 *Assegnato un insieme di dati numerici $x_1, x_2, \ldots, x_n$, si definisce curtosi campionaria il valore:*

$$\gamma_2 = \beta_2 - 3, \tag{4.15}$$

dove

$$\beta_2 = \frac{m_4}{m_2^2}, \tag{4.16}$$

è l'indice di Pearson, avendo denotato con m_2 *il momento centrato campionario di ordine* 2 *e con* m_4 *il momento centrato campionario di ordine* 4.

Si nota che anche β_2 è un *indice adimensionale*, ossia è indipendente dall'unità di misura dei dati.

Gli indici γ_2 e β_2 permettono di *confrontare la distribuzione di frequenze dei dati con una densità di probabilità normale* standard, caratterizzata da $\beta_2 = 3$ e indice di curtosi $\gamma_2 = 0$. Se risulta

$\beta_2 < 3$ ($\gamma_2 < 0$): la distribuzione di frequenze si definisce *platicurtica*, ossia la distribuzione di frequenze è *più piatta di una normale*;

$\beta_2 > 3$ ($\gamma_2 > 0$): la distribuzione di frequenze si definisce *leptocurtica*, ossia la distribuzione di frequenze è *più piccata di una normale*;

$\beta_2 = 3$ ($\gamma_2 = 0$): la distribuzione di frequenze si definisce *normocurtica* (mesocurtica), ossia piatta come una normale.

Il calcolo della curtosi campionaria ha significato soltanto per distribuzioni di frequenze unimodali, dato che tale indice è confrontato con quello di una normale standard.

Il calcolo della curtosi campionaria può essere implementato in R nel seguente modo:

```
> curt<-function(x){
+ n<-length(x)
+ m2<-(n-1)*var(x)/n
+ m4<- (sum( (x-mean(x))^4))/n
+ m4/(m2^2)-3
+ }
```

Riferendoci ai dati dei vettori dati1, dati2, dati3 e dati4, si nota che

```
> curt(dati1)
[1] -0.5
> curt(dati2)
[1] -0.2336167
> curt(dati3)
[1] -0.7555819
> curt(dati4)
[1] 0.4371804
```

Quindi, la curtosi è negativa per i vettore dati1, dati2 e dati3 (la distribuzione di frequenze è più piatta di una normale, ossia platicurtica) ed è positiva per il vettore dati4 (la distribuzione di frequenze è più piccata di una normale, ossia leptocurtica).

Concludiamo questo capitolo introducendo la media ponderata, che si rivela utile in alcune applicazioni.

4.9 Media ponderata

A differenza della media aritmetica, nella *media ponderata* (media pesata) ciascun numero ha una determinata importanza (peso) che influisce sul calcolo.

Il valore della media ponderata è fornito dalla somma dei prodotti di ciascun numero per il suo peso diviso la somma dei pesi. Considerati n numeri $x_1, x_2, \ldots, x_n$ di rispettivi pesi $w_1, w_2, \ldots, w_n$, la media ponderata è così definita:

$$M_p = \frac{x_1 w_1 + x_2 w_2 + \ldots x_n w_n}{w_1 + w_2 + \ldots w_n} = \frac{\sum_{i=1}^{n} x_i\, w_i}{\sum_{i=1}^{n} w_i}. \tag{4.17}$$

Assegnando a tutti i pesi valore 1 si ritrova la media aritmetica. La media ponderata è più flessibile rispetto alla media aritmetica e si presta a numerose applicazioni. In R la media ponderata di un vettore v con pesi w si calcola con la funzione

weighted.mean(v, w)

Un classico esempio di utilizzazione della media ponderata è il calcolo della media dei voti universitari. Infatti, ad ogni insegnamento è attribuito un peso che esprime i CFU (Crediti Formativi Universitari). Ad esempio, consideriamo i voti e i crediti ottenuti da uno studente universitario in 10 esami

```
> votiStudente<-data.frame(voti=c(24,27,28,30,24,23,29,25,20,26),
+ crediti=c(9,6,12,12,6,6, 9,9,12,6))
> rownames(votiStudente)=c("E1", "E2","E3","E4","E5","E6","E7","E8"
    ,"E9","E10")
> votiStudente # visualizza i voti
    voti crediti
E1    24       9
E2    27       6
E3    28      12
E4    30      12
E5    24       6
E6    23       6
E7    29       9
E8    25       9
E9    20      12
E10   26       6
```

Calcoliamo la media aritmetica e la media pesata

```
> mean(votiStudente$voti)
[1] 25.6
> weighted.mean(votiStudente$voti,votiStudente$crediti)
[1] 25.72414
```

Capitolo 5

Statistica descrittiva bivariata

5.1 Introduzione

In questo capitolo sviluppiamo la statistica descrittiva bivariata, ossia il ramo della statistica che si occupa dei metodi grafici e statistici atti descrivere le relazioni che intercorrono tra due variabili. Particolare attenzione sarà dedicata alla *regressione lineare bivariata*. Estenderemo poi i risultati alla *regressione lineare multipla*. Esamineremo infine alcuni problemi di *regressione non lineare*.

Siano X e Y due variabili di tipo quantitativo. Considerato un campione $(x_1, y_1), (x_2, y_2), \ldots (x_n, y_n)$ costituito da n osservazioni di (X, Y).

Le relazioni tra variabili quantitative possono essere rappresentate graficamente mediante *diagrammi di dispersione* (*scatterplot*) in cui ogni coppia di osservazioni viene rappresentata sotto forma di un punto o di un cerchietto in un piano euclideo. Dopo aver scelto la variabile da porre sulle *ascisse* (*variabile indipendente*) e la variabile da porre sulle *ordinate* (*variabile dipendente*), si disegnano dei punti in corrispondenza delle coppie (x_i, y_i). Il risultato finale è una *nuvola di punti* che può essere ottenuto con la funzione plot(x, y), dove x è il vettore contenente i valori $(x_1, x_2, \ldots, x_n)$ e y è il vettore contenente i valori $(y_1, y_2, \ldots, y_n)$. Il grafico che si ottiene mira ad evidenziare se *le coppie di punti presentano qualche forma di regolarità*. Inoltre, il grafico di dispersione mostra se *esiste una relazione tra le variabili* e di quale tipo è tale relazione (lineare, quadratica, ...).

Consideriamo, ad esempio, i voti conseguiti da 15 studenti negli esami di Programmazione 1 e di Programmazione 2.

```
> df<-data.frame(
+ progr1=c(24,26,30,25,29,27,20,29,27,28,18,21,26,30,28),
+ progr2=c(27,26,29,26,30,27,22,29,27,28,20,20,27,30,30))
> df
   progr1 progr2
1      24     27
2      26     26
3      30     29
4      25     26
5      29     30
```

```
6       27      27
7       20      22
8       29      29
9       27      27
10      28      28
11      18      20
12      21      20
13      26      27
14      30      30
15      28      30
```

Successivamente calcoliamo gli indici statistici di posizione e di dispersione relativi alle singole variabili

```
> median(df$progr1)
[1] 27
> mean(df$progr1)
[1] 25.86667
> sd(df$progr1)
[1] 3.681356
>
> median(df$progr2)
[1] 27
> mean(df$progr2)
[1] 26.53333
> sd(df$progr2)
[1] 3.356586
```

Nella Tabella 5.1 sono riportati la mediana campionaria, media campionaria e la deviazione standard dei vettori progr1 e progr2.

Tabella 5.1: Mediana, media campionaria e deviazione standard dei vettori progr1 e progr2.

	progr1	progr2
Mediana campionaria	27	27
Media campionaria	25.86667	26.53333
Deviazione standard	3.681356	3.356586

Si nota che la mediana campionaria è la stessa per i due insegnamenti, mentre la media campionaria è maggiore per il secondo insegnamento. Inoltre, la deviazione standard campionaria è minore per il secondo insegnamento. Successivamente realizziamo lo scatterplot considerando progr1 come variabile indipendente e progr2 come variabile dipendente; nello scatterplot sono visualizzate le 15 coppie del data frame df. Tracciamo anche nello scatterplot delle linee orizzontali e verticali in corrispondenza delle mediane campionarie e delle medie campionarie dei vettori progr1 e progr2. Le seguenti linee di codice

```
> plot(df$progr1,df$progr2,main="Programmazione 2 in funzione
+ di Programmazione 1",
+ xlab="Programmazione 1",ylab="Programmazione 2", col="red")
> abline(v=median(df$progr1),lty=1, col="magenta")
> abline(v=mean(df$progr1),lty=2, col="blue")
> abline(h=median(df$progr2),lty=1, col="magenta")
```

```
> abline(h=mean(df$progr2),lty=2, col="blue")
> legend(18,30,c("Mediana","Media"),pch=0,col=c("magenta","blue"),
+ cex=0.8)
```

producono il diagramma di dispersione (scatterplot) di Figura 5.1. Si nota che i dati sembrano posizionati intorno ad una retta ascendente e ciò induce a pensare che esista una correlazione lineare positiva tra le variabili.

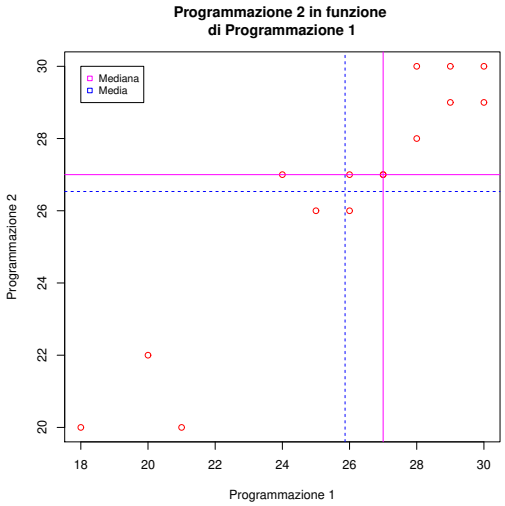

Figura 5.1: Scatterplot dei vettori progr1 e progr2.

5.2 Covarianza e correlazione campionaria

Spesso nelle indagini statistiche si osservano *più variabili quantitative per uno stesso gruppo di individui* ed in tal caso è necessario vedere se esiste una *correlazione tra le variabili*. Un primo passo per indagare l'eventuale dipendenza tra due variabili quantitative X e Y consiste nel disegnare il *diagramma di dispersione* o *scatterplot*.

Per ottenere una misura quantitativa della correlazione tra le variabili si considera la *covarianza campionaria*:

Definizione 5.1 *Assegnato un campione bivariato* $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ *di una variabile quantitativa bidimensionale* (X, Y)*, siano* $\overline{x}$ *e* $\overline{y}$ *rispettivamente le medie campionarie di* $x_1, x_2, \ldots, x_n$ *e di* $y_1, y_2, \ldots, y_n$*. La covarianza campionaria tra le due variabili* X *e* Y *è così definita:*

$$C_{xy} = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \overline{x})\,(y_i - \overline{y}) \qquad (n = 2, 3, \ldots). \tag{5.1}$$

Dalla Definizione 5.1 si nota che se un valore x_i della variabile quantitativa X è grande rispetto a $\overline{x}$, allora la differenza $x_i - \overline{x}$ sarà positiva, mentre se x_i è piccolo rispetto a $\overline{x}$, allora la differenza $x_i - \overline{x}$ sarà negativa. È possibile ragionare in modo analogo per i valori della variabile quantitativa Y. Se si considera il prodotto $(x_i - \overline{x})\,(y_i - \overline{y})$, esso sarà positivo per quelle osservazioni (x_i, y_i) tali che x_i e y_i sono correlate positivamente, ossia per cui $x_i > \overline{x},\ y_i > \overline{y}$ oppure $x_i < \overline{x},\ y_i < \overline{y}$; invece, il prodotto $(x_i - \overline{x})\,(y_i - \overline{y})$ sarà negativo per quelle osservazioni (x_i, y_i) tali che x_i e y_i sono correlate negativamente, ossia per cui $x_i > \overline{x},\ y_i < \overline{y}$ oppure $x_i < \overline{x},\ y_i > \overline{y}$. Quindi, se l'intero campione presenta una *forte correlazione*, c'è da aspettarsi che $\sum_{i=1}^{n}(x_i - \overline{x})\,(y_i - \overline{y})$ assuma un valore molto positivo o molto negativo. Di norma si usa normalizzare tale sommatoria dividendo per $n-1$, in maniera tale da ottenere la varianza campionaria nel caso in cui $x_i = y_i$ per ogni $i = 1, 2, \ldots, n$.

La covarianza campionaria può avere segno positivo, negativo o nullo. Quando $C_{xy} > 0$ si dice che le variabili sono *correlate positivamente*, se $C_{xy} < 0$ le variabili sono *correlate negativamente* e, infine, se $C_{xy} = 0$ le variabili sono *non correlate*.

Dalla (5.1) segue anche che

$$C_{xy} = \frac{1}{n-1}\sum_{i=1}^{n}(x_i\,y_i - y_i\,\overline{x} - x_i\,\overline{y} + \overline{x}\,\overline{y}) = \frac{1}{n-1}\left[\sum_{i=1}^{n} x_i\,y_i - n\,\overline{x}\,\overline{y}\right]$$
$$= \frac{n}{n-1}\left[\frac{1}{n}\sum_{i=1}^{n} x_i\,y_i - \overline{x}\,\overline{y}\right], \tag{5.2}$$

che fornisce un modo alternativo per calcolare la covarianza campionaria.

Per ottenere una misura quantitativa della correlazione tra le variabili si può anche considerare il *coefficiente di correlazione campionario*:

Definizione 5.2 *Assegnato un campione bivariato* $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ *di una variabile quantitativa bidimensionale* (X, Y)*, siano* $\overline{x}$ *e* s_x *la media campionaria e la deviazione standard campionaria di* $x_1, x_2, \ldots, x_n$ *ed inoltre siano* $\overline{y}$ *e* s_y *la media campionaria e la deviazione standard campionaria di* $y_1, y_2, \ldots, y_n$*. Il coefficiente di correlazione campionario tra le due variabili* X *e* Y *è così definito:*

$$r_{xy} = \frac{C_{xy}}{s_x\,s_y}. \tag{5.3}$$

Si nota che il coefficiente di correlazione campionario

- è un indice adimensionale;
- non fa distinzione tra variabile dipendente e variabile indipendente, ossia $r_{xy} = r_{yx}$;
- può essere calcolato soltanto se entrambe le variabili sono quantitative;
- non cambia al variare dell'unità di misura con cui sono espresse le variabili;

- è fortemente influenzato dalla presenza di eventuali valori anomali, così come accade per la media campionaria e la varianza campionaria.

Il coefficiente di correlazione campionario ha lo *stesso segno della covarianza*. Quando $r_{xy} > 0$ si dice che le variabili sono *correlate positivamente*, se $r_{xy} < 0$ le variabili sono *correlate negativamente* e, infine, se $r_{xy} = 0$ le variabili sono *non correlate*.

Proposizione 5.1 *Il coefficiente di correlazione campionario r_{xy} gode delle seguenti proprietà:*

(1) $-1 \leq r_{xy} \leq 1$*;*

(2) se esistono due numeri reali a e b, con $a > 0$, tali che $y_i = a\,x_i + b$ per ogni $i = 1, 2, \ldots, n$, allora $r_{xy} = 1$;

(3) se esistono due numeri reali a e b, con $a < 0$, tali che $y_i = a\,x_i + b$ per ogni $i = 1, 2, \ldots, n$, allora $r_{xy} = -1$;

(4) se esistono quattro numeri reali a, b, c, d e se risulta $z_i = a\,x_i + b$ e $w_i = c\,y_i + d$ per $i = 1, 2, \ldots, n$, allora $r_{zw} = r_{xy}$ se $ac > 0$ e $r_{zw} = -r_{xy}$ se invece $ac < 0$.

Dimostrazione Siano $u_i = x_i - \overline{x}$ e $v_i = y_i - \overline{y}$ per $i = 1, 2, \ldots, n$. Dalla disuguaglianza di Cauchy-Schwarz segue che

$$\Big|\sum_{i=1}^{n} u_i v_i\Big| \leq \sqrt{\sum_{i=1}^{n} u_i^2 \sum_{i=1}^{n} v_i^2},$$

con l'uguaglianza che sussiste se e solo se le sequenze $u_1, u_2, \ldots, u_n$ e $v_1, v_2, \ldots, v_n$ sono proporzionali. Pertanto, ricordando la (4.7) e la (5.1), dalla (5.3) segue che

$$|r_{xy}| = \left| \frac{\sum_{i=1}^{n} u_i v_i}{\sqrt{\sum_{i=1}^{n} u_i^2 \ \sum_{i=1}^{n} v_i^2}} \right| \leq 1,$$

che dimostra la *(1)*.

Dimostriamo ora i punti *(2)* e *(3)*. Se $y_i = a\,x_i + b$ per ogni $i = 1, 2, \ldots, n$, per la linearità della media campionaria si ha:

$$C_{xy} = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \overline{x})\,(y_i - \overline{y}) = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \overline{x})\,(a x_i + b - a\overline{x} - b) = a\,s_x^2.$$

Ricordando la (4.10) segue che $s_y^2 = a^2\,s_x^2$ e quindi

$$r_{xy} = \frac{C_{xy}}{s_x\,s_y} = \frac{a\,s_x^2}{|a|\,s_x^2} = \frac{a}{|a|},$$

da cui seguono *(2)* e la *(3)*.

Infine, per dimostrare il punto *(4)* osserviamo che

$$C_{zw} = \frac{1}{n-1} \sum_{i=1}^{n} (z_i - \overline{z})\,(w_i - \overline{w}) = \frac{a\,c}{n-1} \sum_{i=1}^{n} (x_i - \overline{x})\,(y_i - \overline{y}) = a\,c\,C_{xy}.$$

Ricordando la (4.10) segue che $s_z^2 = a^2\,s_x^2$ e $s_w^2 = c^2\,s_y^2$ e quindi

$$r_{zw} = \frac{C_{zw}}{s_z\,s_w} = \frac{a\,c\,C_{xy}}{|a|\,s_x\,|c|\,s_y} = \frac{a\,c}{|a|\,|c|}\,r_{xy},$$

che coincide con r_{xy} se $ac > 0$ e con $-r_{xy}$ se $ac < 0$. □

La proprietà *(1)* afferma che il coefficiente di correlazione campionario è compreso nell'intervallo $[-1, 1]$. Le proprietà *(2)* e *(3)* mostrano che i valori limite -1 e $+1$ sono effettivamente raggiunti solo quando tra X e Y sussiste una *relazione lineare*, ossia quando *i punti dello scatterplot giacciono tutti su di una retta*. La proprietà *(4)* afferma che il quadrato del coefficiente di correlazione non cambia se sommiamo costanti o moltiplichiamo per costanti tutti i valori di X e/o di Y. Ciò significa che *il coefficiente di correlazione non dipende dalle unità di misura scelta per rappresentare i dati*.

Nel linguaggio R le covarianze campionarie e le correlazioni campionarie fra una coppie di variabili numeriche X e Y possono essere immediatamente ottenute con le rispettive funzioni $\mathsf{cov(X, Y)}$ e $\mathsf{cor(X, Y)}$.

Occorre ricordare che il coefficiente di correlazione campionario r_{xy} misura la *forza del legame di natura lineare esistente tra due variabili quantitative*. Eventuali relazioni tra le variabili che assumono una forma curvilinea non possono pertanto essere individuati con tale coefficiente. Riassumendo si può dire che il *segno* di r_{xy} indica la *direzione della retta interpolante* e indica la presenza di una tra le seguenti situazioni:

- $r_{xy} = 1$ (correlazione perfetta positiva) tutti i punti sono allineati su una linea retta ascendente;

- r_{xy} compreso tra 0 e 1 estremi esclusi (correlazione positiva) i punti (x_i, y_i) sono posizionati in una nuvola attorno ad una *linea retta interpolante ascendente* (in tal caso x_i e y_i tendono ad essere grandi e piccoli insieme);

- $r_{xy} = 0$ (nessuna correlazione) i punti sono completamente dispersi in una nuvola che non presenta alcuna evidente direzione di natura lineare;

- r_{xy} compreso tra -1 e 0 estremi esclusi (correlazione negativa) i punti (x_i, y_i) sono posizionati in una nuvola attorno ad una *linea retta interpolante discendente* (in tal caso x_i è grande y_i è piccolo e viceversa);

- $r_{xy} = -1$ (correlazione perfetta negativa) tutti i punti sono allineati su una linea retta discendente;

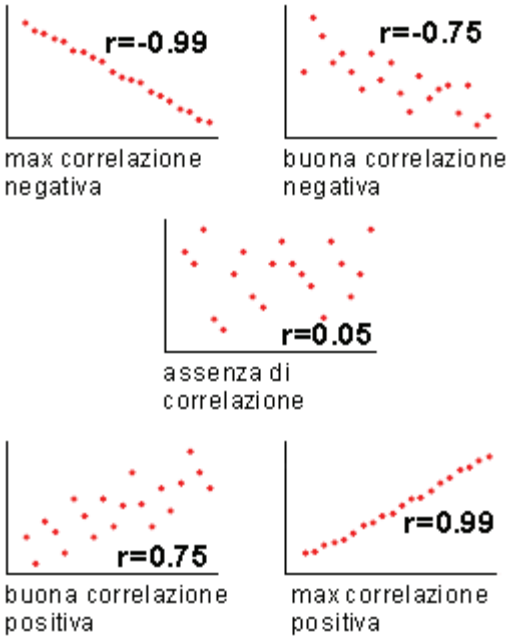

Figura 5.2: Scatterplot e correlazione campionaria tra due variabili

In Figura 5.2 sono mostrati alcuni scatterplot e sono evidenziate le relative correlazioni campionarie.

Consideriamo, ad esempio, i voti conseguiti da 15 studenti negli esami di Programmazione 1 e di Programmazione 2. Calcoliamo la covarianza campionaria e il coefficiente di correlazione campionario.

```
> cov(df$progr1,df$progr2)
[1] 11.71905
> cor(df$progr1,df$progr2)
[1] 0.9483896
```

I dati dei due vettori progr1 e progr2 sono positivamente correlati essendo la covarianza campionaria uguale a 11.71905, ossia i valori assunti dal primo e dal secondo vettore tendono ad essere grandi e piccoli insieme. Inoltre, il coefficiente di correlazione è uguale a 0.9483896 ed è prossimo all'unità. Ciò indica, come è evidenziato nel diagramma di dispersione (scatterplot) di Figura 5.3, che esiste una forte correlazione lineare tra i voti di Programmazione 1 e Programmazione 2.

Lo scatterplot di Figura 5.3, è stato ottenuto con le seguenti linee di codice:

```
> plot(df$progr1,df$progr2,main="Retta di regressione",
+ xlab="Programmazione 1",ylab="Programmazione 2", col="red")
> abline(lm(df$progr2~df$progr1), col="blue")
```

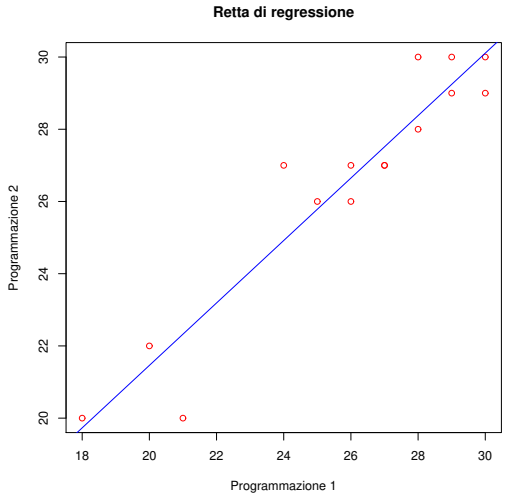

Figura 5.3: Scatterplot retta interpolante stimata (retta di regressione).

La funzione abline(lm(df$progr2~df$progr1)) permette di aggiungere allo scatterplot relativo ai voti dei due esami la linea interpolante stimata. Il nome lm() della funzione presente in abline() è l'acronimo di *linear model*. Dell'individuazione di tale retta interpolante si discuterà nel prossimo paragrafo.

Gli scatterplot sono dei potenti mezzi per visualizzare le eventuali relazioni che possono intercorrere tra variabili quantitative. Infatti, alcune volte osservando il grafico si nota che i punti si dispongono attorno a qualche linea orientata in qualche direzione, come ad esempio si verifica in Figura 5.3. Tuttavia per avere un'idea più accurata del fenomeno, è necessario utilizzare altre tecniche statistiche in grado di misurare con maggiore precisione questo legame.

Il *modello lineare* viene di solito utilizzato per spiegare, descrivere, o anche prevedere un andamento futuro sulla base della relazione che si instaura tra una variabile Y, chiamata variabile dipendente, e una o più altre variabili che assumono il significato di variabili indipendenti $X_1, X_2, \ldots, X_p$. Nel caso in cui $p = 1$, l'analisi prende il nome di *regressione semplice*, mentre se $p = 2, 3 \ldots$ si parla di *regressione multipla*. Per poter utilizzare un modello di regressione è fondamentale individuare in primo luogo quali sono le variabili indipendenti e quale è la variabile dipendente.

5.3 Regressione lineare semplice

Il *modello di regressione lineare semplice* è esprimibile attraverso l'equazione di una retta che riesce ad interpolare la nuvola di punti dello scatterplot meglio di tutte e altre possibili rette. Consideriamo l'equazione della retta:

$$Y = \alpha + \beta X$$

dove

- α è l'intercetta;

- β è il coefficiente angolare.

Il *coefficiente angolare* β esprime quantitativamente la *pendenza (inclinazione) della retta*: un coefficiente angolare positivo ($\beta > 0$) indica una retta di regressione crescente, un coefficiente angolare negativo ($\beta < 0$) indica una retta decrescente; un coefficiente angolare nullo ($\beta = 0$) indica una retta orizzontale. L'*intercetta* α invece corrisponde all'ordinata del punto di intersezione della retta interpolante (di regressione) con l'asse delle ordinate.

L'identificazione di questa retta viene ottenuta applicando il *metodo dei minimi quadrati*. I *coefficienti di regressione* sono i valori α e β per i quali la somma Q dei quadrati degli errori

$$Q = \sum_{i=1}^{n} \Big[y_i - (\alpha + \beta\, x_i) \Big]^2$$

sia minima, dove n è il numero di osservazioni, $(x_1, x_2, \ldots, x_n)$ sono i valori osservati della variabile X e $(y_1, y_2, \ldots, y_n)$ sono i valori osservati della variabile Y. Derivando Q rispetto a α e β e uguagliando a zero le derivate parziali ottenute si ha:

$$\frac{\partial Q}{\partial \alpha} = -2 \sum_{i=1}^{n} \Big[y_i - (\alpha + \beta\, x_i) \Big] = 0,$$
$$\frac{\partial Q}{\partial \beta} = -2 \sum_{i=1}^{n} x_i \Big[y_i - (\alpha + \beta\, x_i) \Big] = 0,$$

da cui si ottiene il sistema di equazioni lineari nelle incognite α e β:

$$n\,\alpha + \beta \sum_{i=1}^{n} x_i = \sum_{i=1}^{n} y_i,$$
$$\alpha \sum_{i=1}^{n} x_i + \beta \sum_{i=1}^{n} x_i^2 = \sum_{i=1}^{n} x_i\, y_i.$$

Ricordando la (4.9) e la (5.2) risulta:

$$\sum_{i=1}^{n} x_i = n\,\overline{x}, \qquad \sum_{i=1}^{n} y_i = n\,\overline{y},$$
$$\sum_{i=1}^{n} x_i\, y_i = (n-1)\, C_{xy} + n\,\overline{x}\,\overline{y}, \qquad \sum_{i=1}^{n} x_i^2 = (n-1) s_x^2 + n\,\overline{x}^2,$$

e quindi il sistema diventa:

$$\alpha + \beta\,\overline{x} = \overline{y},$$
$$n\,\alpha\,\overline{x} + \beta \Big[(n-1)\, s_x^2 + n\,\overline{x}^2 \Big] = (n-1)\, C_{xy} + n\,\overline{x}\,\overline{y}.$$

Sostituendo $\alpha = \overline{y} - \beta\,\overline{x}$ nella seconda equazione si ottiene:

$$n\left(\overline{y} - \beta\,\overline{x}\right)\overline{x} + \beta\Big[(n-1)\,s_x^2 + n\,\overline{x}^2\Big] = (n-1)\,C_{xy} + n\,\overline{x}\,\overline{y},$$

ossia

$$\beta = \frac{C_{xy}}{s_x^2} = \frac{C_{xy}}{s_x\,s_y}\,\frac{s_y}{s_x} = \frac{s_y}{s_x}\,r_{xy}.$$

Pertanto il metodo dei minimi quadrati conduce a:

$$\beta = \frac{s_y}{s_x}\,r_{xy}, \qquad\qquad \alpha = \overline{y} - \beta\,\overline{x}. \tag{5.4}$$

Si nota quindi che le medie campionarie, le deviazioni standard campionarie e il coefficiente di correlazione permettono di stimare i parametri α e β della retta di regressione. Ricordiamo che in R la funzione C_{xy} viene calcolata tramite cov(x, y), r_{xy} tramite la funzione cor(x, y), la funzione s_x^2 tramite la funzione var(x), le funzioni $\overline{x}$ e $\overline{y}$ rispettivamente tramite mean(x) e mean(y).

Ad esempio, riferendoci ai voti di Programmazione 1 e di Programmazione 2 le seguenti linee di codice calcolano il coefficiente angolare β e l'intercetta α:

```
> beta<-(sd(df$progr2)/sd(df$progr1))*cor(df$progr1,df$progr2)
>
> alpha<-mean(df$progr2)-beta*mean(df$progr1)
> c(alpha,beta)
[1] 4.1658468 0.8647224
```

La funzione

lm(y~x)

è utilizzata per eseguire le analisi di regressione lineari e fornisce i valori dell'intercetta α e del coefficiente angolare β. Ricordiamo che il nome lm() della funzione rappresenta l'acronimo di *linear model*. L'argomento y~x passato alla funzione lm() indica che y *dipende da* x, ossia che x è la variabile indipendente e y la variabile dipendente.

La funzione

abline(lm(y~x))

permette di aggiungere la retta di regressione al grafico dello scatterplot, con y che dipende da x.

Ad esempio, per il data frame dei voti dei 15 studenti le seguenti linee di codice

```
> lm(df$progr2~df$progr1)

Call:
lm(formula = df$progr2 ~ df$progr1)

Coefficients:
(Intercept)    df$progr1
     4.1658       0.8647
```

mostrano che la retta di regressione ha intercetta $\alpha = 4.1658$ e coefficiente angolare $\beta = 0.8647$. La retta di regressione ha quindi equazione

$$y = 4.1658 + 0.8647\, x$$

La rappresentazione della retta così calcolata può essere aggiunta allo scatterplot facendo uso della funzione abline(lm(progr2~progr1)); in questo modo il grafico che si ottiene risulta arricchito della linea di regressione, così come mostrato in Figura 5.3.

Le seguenti linee di codice

```
> linearmodel<- lm(df$progr2~df$progr1)
> attributes(linearmodel)
$names
 [1] "coefficients"  "residuals"     "effects"       "rank"
 [5] "fitted.values"    "assign"        "qr"
 [8]  "df.residual"   "xlevels"       "call"
[11] "terms"         "model"

$class
[1] "lm"
```

forniscono la *lista degli attributi* dell'oggetto linearmodel ottenuto tramite la funzione lm(). Da tale oggetto possiamo estrarre i singoli valori

```
> linearmodel$coefficients
(Intercept)   df$progr1
  4.1658468   0.8647224
>
> linearmodel$coefficients[[1]]
[1] 4.165847
>
> linearmodel$coefficients[[2]]
[1] 0.8647224
```

Residui

Una volta calcolati i valori dei coefficienti α e β e disegnata la retta di regressione che interpola la nuvola dei punti nel corrispondente scatterplot, è possibile osservare quanto questa retta si adatta ai punti che individuano le osservazioni. Denotiamo con

$$\widehat{y}_i = \alpha + \beta\, x_i \qquad (i = 1, 2, \ldots, n) \tag{5.5}$$

i valori stimati ottenuti mediante la retta di regressione. I punti $(x_1, \widehat{y}_1)$, $(x_2, \widehat{y}_2)$, …, $(x_n, \widehat{y}_n)$ sono posizionati sulla retta di regressione.

In generale, esisteranno degli scostamenti (*residui*) tra le ordinate dei punti y_i (*valori osservati*) e i corrispondenti *valori stimati* $\widehat{y}_i$.

La *media campionaria dei valori stimati* $(\widehat{y}_1, \widehat{y}_2, \ldots, \widehat{y}_n)$ *è uguale alla media campionaria* $\overline{y}$ *delle osservazioni* $(y_1, y_2, \ldots, y_n)$. Infatti, risulta che

$$\frac{1}{n}\sum_{i=1}^{n} \widehat{y}_i = \frac{1}{n}\sum_{i=1}^{n} (\alpha + \beta\, x_i) = \alpha + \beta\, \overline{x} = (\overline{y} - \beta\, \overline{x}) + \beta\, \overline{x} = \overline{y}.$$

I *residui* sono così definiti

$$E_i = y_i - \widehat{y}_i = y_i - (\alpha + \beta\, x_i) \qquad (i = 1, 2, \ldots, n) \tag{5.6}$$

e mostrano di quanto si discostano i valori osservati y_i dai valori stimati $\widehat{y}_i$ con la retta di regressione. La *media campionaria dei residui* $\overline{E}$ *è nulla*, ossia in media gli scostamenti positivi e negativi si compensano. Infatti, ricordando (5.6) risulta:

$$\overline{E} = \frac{1}{n}\sum_{i=1}^{n} E_i = \frac{1}{n}\sum_{i=1}^{n}\big(y_i - \widehat{y}_i\big) = \overline{y} - \frac{1}{n}\sum_{i=1}^{n}\widehat{y}_i = 0. \tag{5.7}$$

La *varianza campionaria dei residui* è

$$s_E^2 = \frac{1}{n-1}\sum_{i=1}^{n}\big(E_i - \overline{E}\big)^2 = \frac{1}{n-1}\sum_{i=1}^{n} E_i^2, \tag{5.8}$$

essendo $\overline{E} = 0$.

In R per calcolare il vettore dei valori stimati $(\widehat{y}_1, \widehat{y}_2, \ldots, \widehat{y}_n)$ si utilizza la funzione

fitted(lm(y~x))

con y che dipende da x. Invece, per calcolare il vettore dei residui $(E_1, E_2, \ldots, E_n)$ si utilizza la funzione

resid(lm(y~x))

Ad esempio, per il data frame dei voti dei 15 studenti le seguenti linee di codice

```
 > stime<-fitted(lm(df$progr2~df$progr1))
> stime # visualizza le ordinate stimate
       1        2        3        4        5        6        7
24.91918 26.64863 30.10752 25.78391 29.24280 27.51335 21.46030

       8        9       10       11       12       13       14
 29.24280 27.51335 28.37807 19.73085 22.32502 26.64863 30.10752
       15
 28.37807
```

determinano i valori stimati $(\widehat{y}_1, \widehat{y}_2, \ldots, \widehat{y}_{15})$. Ad esempio, il primo valore del vettore stime può essere così estratto:

```
> linearmodel <- lm(df$progr2~df$progr1)
> linearmodel$fitted.values[[1]]
[1] 24.91918
```

e corrisponde a

$$\widehat{y}_1 = \alpha + \beta\, x_1 = 4.165846 + 0.8647224 \cdot 24 = 24.91918,$$

poiché il primo studente ha ottenuto 24 all'esame di Programmazione 1.

Inoltre, le seguenti linee di codice

```
> residui<-resid(lm(df$progr2~df$progr1))
> residui # visualizza il vettore dei residui
         1          2          3          4          5          6
 2.0808152 -0.6486297 -1.1075193  0.2160928  0.7572031 -0.5133521
         7          8          9         10         11         12
 0.5397048 -0.2427969 -0.5133521 -0.3780745  0.2691497 -2.3250176
        13         14         15
0.3513703 -0.1075193  1.6219255
```

determinano i residui $(E_1, E_2, \ldots, E_{15})$. Ad esempio, il primo valore del vettore residui può essere così estratto:

```
> linearmodel<- lm(df$progr2~df$progr1)
> linearmodel$residuals[[1]]
[1] 2.080815
```

e corrisponde a

$$E_1 = y_1 - \widehat{y}_1 = 27 - 24.91918 = 2.08082.$$

poiché il primo studente ha ottenuto 27 all'esame di Programmazione 2. Come mostrato precedentemente, la media dei residui è zero. Invece, la mediana, la varianza campionaria e la deviazione standard campionaria dei residui sono:

```
> median(linearmodel$residuals)
[1] -0.1075193
>
> var(linearmodel$residuals)
[1] 1.132943
>
> sd(linearmodel$residuals)
[1] 1.064398
```

Non si può invece calcolare il coefficiente di variazione, essendo la media campionaria dei residui nulla.

È possibile rappresentare graficamente i residui:

- tracciando dei segmenti verticali che congiungono i valori stimati $\widehat{y}_i$ (sulla retta di regressione) e i valori osservati y_i $(i = 1, 2, \ldots, n)$;
- rappresentando i valori dei residui E_i rispetto alle osservazioni x_i (variabile indipendente) $(i = 1, 2, \ldots, n)$;
- rappresentando i residui standardizzati $E_i^{(s)} = E_i / s_E$ rispetto ai valori stimati $\widehat{y}_i$ $(i = 1, 2, \ldots, n)$.

→ Segmenti che congiungono i valori stimati e i valori osservati

Vogliamo ora realizzare il grafico dei residui ottenuto aggiungendo, al grafico contenente lo scatterplot e la retta di regressione, dei segmenti verticali che visualizzano i residui. Questi segmenti sono ottenuti sovrapponendo al grafico ottenuto in Figura 5.3 dei segmenti che congiungono i seguenti due punti (x_i, y_i) e $(x_i, \widehat{y}_i)$ $(i = 1, 2, \ldots, 15)$. Le seguenti linee di codice

```
> plot(df$progr1,df$progr2,main="Retta di regressione e residui",
+ xlab="Programmazione 1",ylab="Programmazione 2", col="red")
> abline(lm(df$progr2~df$progr1), col="blue")
> stime<-fitted(lm(df$progr2~df$progr1))
> segments(df$progr1,stime,df$progr1,df$progr2,
+ col="magenta")
```

producono il grafico illustrato in Figura 5.4.

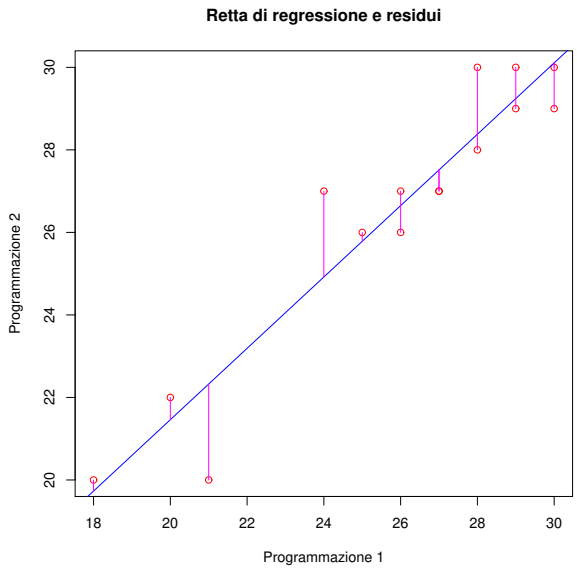

Figura 5.4: Grafico dei residui (tra valori osservati e valori stimati con la retta di regressione).

→ Valori dei residui rispetto alle osservazioni della variabile indipendente

Un esame più accurato del modo con cui la retta di regressione interpola i dati e di come i residui si dispongano intorno alla retta interpolante influenzandone la posizione, può essere ottenuto attraverso il *diagramma dei residui* che è un grafico in cui *i valori dei residui sono posti sull'asse delle ordinate e quelli della variabile indipendente sull'asse delle ascisse.*

Desideriamo realizzare il diagramma dei residui ponendo i valori dei residui sull'asse delle ordinate e quelli della variabile indipendente sull'asse delle ascisse.

Dopo aver definito il data frame df, le seguenti linee di codice

```
> residui<-resid(lm(df$progr2~df$progr1))
> plot(df$progr1,residui, main="Diagramma dei residui",
+ xlab="Programmazione 1",ylab="Residui",pch=9,col="red")
> abline(h=0,col="blue",lty=2)
```

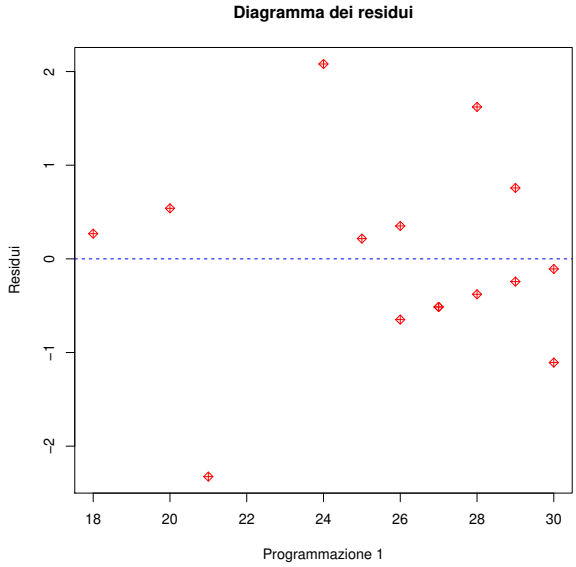

Figura 5.5: Residui in funzione dei voti di Programmazione 1.

producono il grafico illustrato in Figura 5.5. I punti indicano la posizione dove si collocano i residui rispetto ai voti di Programmazione 1. La retta orizzontale è posizionata nello zero e corrisponde alla media campionaria dei residui, che è nulla. Si nota che i punti sono disposti quasi casualmente attorno alla linea orizzontale e non si evidenzia nessun comportamento particolare nella distribuzione dei punti.

Il diagramma dei residui aiuta a comprendere quale è l'adattamento della retta di regressione rispetto ai dati, consentendo di identificare quali sono le informazioni che hanno una forte influenza sulla collocazione e direzione della retta di regressione. Occorre notare che la posizione della retta di regressione è fortemente influenzata dalla presenza di eventuali *valori anomali* che si discostano in modo significativo dagli altri. L'analisi dei residui aiuta ad individuare eventuali punti isolati (valori anomali) dovuti ad errori nella stima. Tali valori possono perturbare significativamente la stima dei parametri di regressione e influenzare l'interpretazione dei residui. Eliminando i valori anomali la varianza campionaria dei residui diminuisce.

→ Valori dei residui standardizzati rispetto ai valori stimati

È spesso interessante calcolare i *residui standardizzati* così definiti:

$$E_i^{(s)} = \frac{E_i - \overline{E}}{s_E} = \frac{E_i}{s_E},$$

che risultano essere caratterizzati da media campionaria nulla e varianza unitaria. Ad esempio, per il data frame dei voti dei 15 studenti risulta:

```
> residui<-resid(lm(df$progr2~df$progr1))
> residuistandard<-residui/sd(residui)
> residuistandard # visualizza il vettore dei residui
    standardizzati
         1          2          3          4          5          6
 1.9549218 -0.6093863 -1.0405122  0.2030187  0.7113908 -0.4822933
         7          8          9         10         11         12
 0.5070516 -0.2281072 -0.4822933 -0.3552002  0.2528656 -2.1843494
        13         14         15
0.3301117 -0.1010142  1.5237958
```

È poi possibile realizzare un grafico in cui i residui standardizzati (ordinate) vengono disegnati in funzione dei valori stimati (ascisse) mediante la retta di regressione. Il seguente codice

```
> stime<-fitted(lm(df$progr2~df$progr1))
> residui<-resid(lm(df$progr2~df$progr1))
> residuistandard<-residui/sd(residui)
> plot(stime,residuistandard, main="Residui standard rispetto ai
    valori stimati",
+ xlab="Valori stimati",ylab="Residui standard",pch=5,col="red")
> abline(h=0,col="blue",lty=2)
```

produce il grafico in Figura 5.6. I punti indicano la posizione dove si collocano i residui standardizzati rispetto ai valori stimati con la retta di regressione. La retta orizzontale è posizionata nello zero, che corrisponde alla media campionaria dei residui standardizzati. Anche in questo caso i punti sono disposti quasi casualmente attorno alla linea orizzontale e non si evidenzia nessuna tendenza particolare nella distribuzione dei punti.

5.3.1 Coefficiente di determinazione

Poiché si è interessati a vedere quanto la retta si adatta ai dati, l'accento può essere posto sul quadrato del coefficiente di correlazione e su quanto esso si avvicini ad uno. È chiaro che r^2_{xy} molto vicino ad 1 indicherà che tutti i punti tenderanno ad allinearsi lungo la retta di regressione, mentre r^2_{xy} prossimo a 0 esprime una completa incapacità della retta di rappresentare la distribuzione dei dati considerati.

Definizione 5.3 *Il coefficiente di determinazione per la regressione lineare semplice è il rapporto tra la varianza dei valori stimati tramite la retta di regressione e la varianza dei valori osservati. Pertanto, se si denota con* $(y_1, y_2, \ldots, y_n)$ *il vettore dei dati della variabile dipendente, con* $\overline{y}$ *la sua media campionaria e con* $(\widehat{y}_1, \widehat{y}_2, \ldots, \widehat{y}_n)$ *i valori stimati attraverso la retta di regressione (la cui media campionaria è* $\overline{y}$*), il* coefficiente di determinazione *(detto anche* r-square*) è così definito:*

$$D^2 = \frac{\frac{1}{n-1}\sum_{i=1}^{n}(\widehat{y}_i - \overline{y})^2}{\frac{1}{n-1}\sum_{i=1}^{n}(y_i - \overline{y})^2} = \frac{\sum_{i=1}^{n}(\widehat{y}_i - \overline{y})^2}{\sum_{i=1}^{n}(y_i - \overline{y})^2}. \tag{5.9}$$

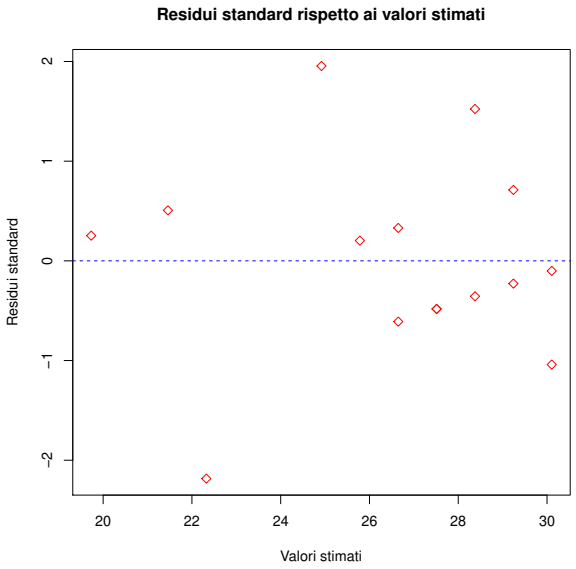

Figura 5.6: Residui standardizzati in funzione dei valori stimati con la retta di regressione.

Proposizione 5.2 *Nel caso di regressione lineare semplice,* il coefficiente di determinazione coincide con il quadrato del coefficiente di correlazione*, ossia*

$$D^2 = r_{xy}^2 \tag{5.10}$$

Dimostrazione Ricordando (5.4) e (5.5), risulta

$$\widehat{y}_i - \overline{y} = \alpha + \beta x_i - \overline{y} = (\overline{y} - \beta\,\overline{x}) + \beta x_i - \overline{y} = \beta\,(x_i - \overline{x}).$$

Pertanto, il numeratore della (5.9) diventa:

$$\frac{1}{n-1}\sum_{i=1}^{n}(\widehat{y}_i - \overline{y})^2 = \frac{\beta^2}{n-1}\sum_{i=1}^{n}(x_i - \overline{x})^2 = \beta^2\, s_x^2 = \Big(\frac{s_y^2}{s_x^2} r_{xy}^2\Big) s_x^2 = s_y^2 r_{xy}^2,$$

mentre il denominatore è:

$$\frac{1}{n-1}\sum_{i=1}^{n}(y_i - \overline{y})^2 = s_y^2.$$

Sussiste quindi la (5.10). □

In R, nel caso di regressione lineare semplice, il coefficiente di determinazione D^2 si può calcolare utilizzando il quadrato del coefficiente di correlazione oppure la funzione `summary(lm(y~x))$r.square`. Riferendoci ai voti degli studenti si ha:

```
> (cor(df$progr1,df$progr2))^2
[1] 0.8994429
>
> summary(lm(df$progr2~df$progr1))$r.square
[1] 0.8994429
```

5.4 Regressione lineare multipla

In molte applicazioni dell'analisi di regressione sono coinvolte situazioni con più di una singola variabile indipendente. Il modello di regressione lineare multipla viene utilizzato per spiegare la relazione tra una variabile quantitativa Y, detta variabile dipendente, e le variabili quantitative indipendenti $X_1, X_2, \ldots, X_p$.

Se si definisce un data frame dfm, contenente n osservazioni delle $p+1$ variabili $Y, X_1, X_2, \ldots, X_p$, allora cov(dfm) e cor(dfm) forniscono due matrici di dimensioni $(p+1)\cdot(p+1)$ i cui elementi solo le covarianze e le correlazioni tra coppie di variabili. In particolare, tali matrici sono simmetriche; la matrice delle covarianze contiene sulla diagonale principale la varianza delle singole colonne del data frame, mentre la matrice delle correlazioni contiene il numero 1 sulla diagonale principale. La matrice di correlazione evidenzia tutte le correlazioni lineari tra le coppie di variabili, ossia misura la forza del legame di natura lineare esistente tra tutte le coppie di variabili quantitative.

Ad esempio, consideriamo il voto di diploma e i voti conseguiti negli esami di Programmazione 1 e di Programmazione 2 da 15 studenti universitari.

```
> dfm<-data.frame(
+ diploma=c(92,94,100,90,98,90,87,89,100,91,
+ 83,85,97,95,99),
+ progr1=c(24,26,30,25,29,27,20,29,27,28,18,21,26,30,28),
+ progr2=c(27,26,29,26,30,27,22,29,27,28,20,20,27,30,30))
> dfm
   diploma progr1 progr2
1       92     24     27
2       94     26     26
3      100     30     29
4       90     25     26
5       98     29     30
6       90     27     27
7       87     20     22
8       89     29     29
9      100     27     27
10      91     28     28
11      83     18     20
12      85     21     20
13      97     26     27
14      95     30     30
15      99     28     30
```

In questo caso la matrice delle covarianze campionarie è

```
> cov(dfm)
         diploma   progr1   progr2
```

```
diploma  29.80952  15.09524  14.19048
progr1   15.09524  13.55238  11.71905
progr2   14.19048  11.71905  11.26667
```

mentre la matrice delle correlazioni è

```
> cor(dfm)
          diploma    progr1    progr2
diploma 1.0000000 0.7510254 0.7743222
progr1  0.7510254 1.0000000 0.9483896
progr2  0.7743222 0.9483896 1.0000000
```

Si nota che esiste una forte correlazione lineare tra Programmazione 1 e Programmazione 2 ed è anche sufficientemente alta la correlazione lineare tra il voto di diploma e di Programmazione 1 e Programmazione 2.

La funzione pairs() è in grado di visualizzare in un'unica finestra grafica una pluralità di scatterplot ottenuti mettendo in relazione tutte le coppie di variabili quantitative definite all'interno di un data frame. Riferendosi all'esempio precedente, la seguente linea di codice

```
> pairs(dfm,main="Scatterplot per le coppie di variabili",col="red"
    )
```

produce il grafico in Figura 5.7.

Quindi, per ogni coppia di variabili possiamo utilizzare il modello di regressione lineare semplice esprimibile attraverso l'equazione $Y = \alpha + \beta X$. Invece, il modello di regressione lineare multipla con p variabili indipendenti è esprimibile attraverso l'equazione:

$$Y = \alpha + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p.$$

dove

- α è l'intercetta, ossia il valore di Y quando $X_1 = X_2 = \ldots = X_p = 0$;
- $\beta_1, \beta_2, \ldots, \beta_p$ sono i *regressori*. In particolare, β_1 rappresenta l'inclinazione di Y rispetto alla variabile X_1 tenendo costanti le variabili $X_2, X_3, \ldots, X_p$, ... β_p rappresenta l'inclinazione di Y rispetto alla variabile X_p tenendo costanti le variabili $X_1, X_2, \ldots, X_{p-1}$.

Ad esempio, la spesa familiare mensile (Y) dipende da un insieme di variabili, quali il reddito familiare mensile (X_1), il numero di componenti della famiglia (X_2), l'età del capofamiglia (X_3), ...

Per determinare le stime di $\alpha, \beta_1, \ldots, \beta_p$ ricorriamo al metodo dei minimi quadrati. Occorre quindi minimizzare la quantità

$$Q = \sum_{i=1}^{n} \Big[y_i - (\alpha + \beta_1 x_{i,1} + \beta_2 x_{i,2} + \ldots \beta_p x_{i,p}) \Big]^2,$$

dove n è il numero di osservazioni, $(x_{1,j}, x_{2,j}, \ldots, x_{n,j})$ sono i valori osservati della variabile X_j ($j = 1, 2, \ldots, p$) e $(y_1, y_2, \ldots, y_n)$ i valori osservati della variabile Y. Derivando rispetto ai parametri $\alpha, \beta_1, \beta_2, \ldots, \beta_p$ si perviene ad un

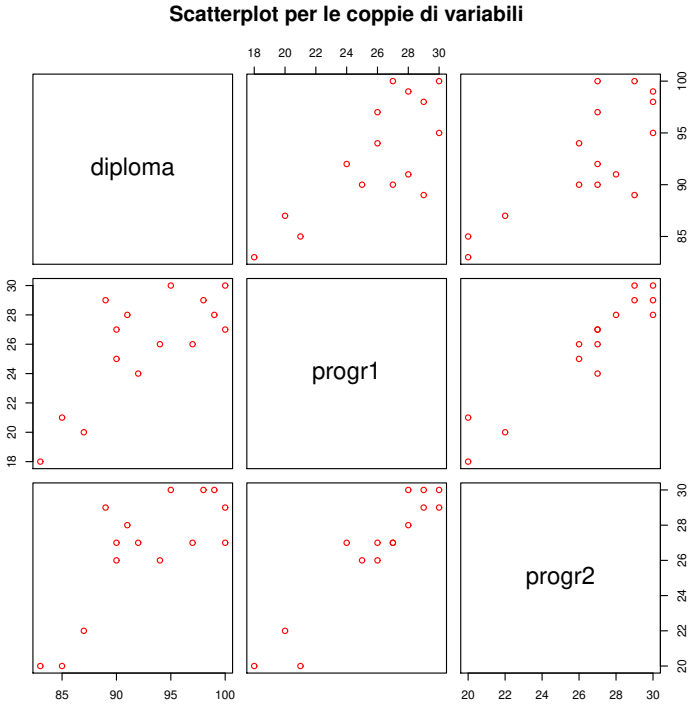

Figura 5.7: Scatterplot ottenuti mettendo in relazione i voti del diploma e di Programmazione 1 e Programmazione 2.

sistema di $p+1$ equazioni:

$$\begin{aligned}
&\sum_{i=1}^{n}\Big[y_i - (\alpha + \beta_1\, x_{i,1} + \beta_2\, x_{i,2} + \ldots \beta_p\, x_{i,p})\Big] = 0,\\
&\sum_{i=1}^{n} x_{i,1}\Big[y_i - (\alpha + \beta_1\, x_{i,1} + \beta_2\, x_{i,2} + \ldots \beta_p\, x_{i,p})\Big] = 0,\\
&\sum_{i=1}^{n} x_{i,2}\Big[y_i - (\alpha + \beta_1\, x_{i,1} + \beta_2\, x_{i,2} + \ldots \beta_p\, x_{i,p})\Big] = 0,\\
&\cdots\cdots\cdots\cdots\\
&\cdots\cdots\cdots\cdots\\
&\sum_{i=1}^{n} x_{i,k}\Big[y_i - (\alpha + \beta_1\, x_{i,1} + \beta_2\, x_{i,2} + \ldots \beta_p\, x_{i,p})\Big] = 0,
\end{aligned}$$

ossia

$$\begin{aligned}
&\alpha\, n + \beta_1 \sum_{i=1}^{n} x_{i,1} + \beta_2 \sum_{i=1}^{n} x_{i,2} + \ldots + \beta_p \sum_{i=1}^{n} x_{i,p} = \sum_{i=1}^{n} y_i\\
&\alpha \sum_{i=1}^{n} x_{i,1} + \beta_1 \sum_{i=1}^{n} x_{i,1}^2 + \beta_2 \sum_{i=1}^{n} x_{i,1}x_{i,2} + \ldots + \ldots + \beta_p \sum_{i=1}^{n} x_{i,1}x_{i,p} = \sum_{i=1}^{n} x_{i,1}y_i\\
&\alpha \sum_{i=1}^{n} x_{i,2} + \beta_1 \sum_{i=1}^{n} x_{i,1}x_{i,2} + \beta_2 \sum_{i=1}^{n} x_{i,2}^2 + \ldots + \ldots + \beta_p \sum_{i=1}^{n} x_{i,2}x_{i,p} = \sum_{i=1}^{n} x_{i,2}y_i\\
&\cdots\cdots\cdots\cdots\\
&\cdots\cdots\cdots\cdots\\
&\alpha \sum_{i=1}^{n} x_{i,p} + \beta_1 \sum_{i=1}^{n} x_{i,1}x_{i,p} + \beta_2 \sum_{i=1}^{n} x_{i,2}x_{i,p} + \ldots + \ldots + \beta_p \sum_{i=1}^{n} x_{i,p}^2 = \sum_{i=1}^{n} x_{i,p}y_i.
\end{aligned}$$

Il sistema può essere scritto in forma matriciale nel seguente modo:

$$\begin{pmatrix}
n & \sum_{i=1}^{n} x_{i,1} & \sum_{i=1}^{n} x_{i,2} & \ldots & \sum_{i=1}^{n} x_{i,p}\\
\sum_{i=1}^{n} x_{i,1} & \sum_{i=1}^{n} x_{i,1}^2 & \sum_{i=1}^{n} x_{i,1}x_{i,2} & \ldots & \sum_{i=1}^{n} x_{i,1}x_{i,p}\\
\vdots & \vdots & \ddots & \vdots & \\
\sum_{i=1}^{n} x_{i,p} & \sum_{i=1}^{n} x_{i,1}x_{i,p} & \sum_{i=1}^{n} x_{i,2}x_{i,p} & \ldots & \sum_{i=1}^{n} x_{i,p}^2
\end{pmatrix}
\begin{pmatrix}
\alpha\\ \beta_1\\ \beta_2\\ \vdots\\ \vdots\\ \beta_p
\end{pmatrix}
=
\begin{pmatrix}
\sum_{i=1}^{n} y_i\\
\sum_{i=1}^{n} x_{i,1}y_i\\
\vdots\\
\sum_{i=1}^{n} x_{i,p}y_i
\end{pmatrix} \tag{5.11}$$

Dalla prima equazione ricaviamo

$$\overline{y} - \big(\alpha + \beta_1\overline{x}_1 + \beta_2\overline{x}_2 + \ldots + \beta_p\overline{x}_p\big) = 0, \tag{5.12}$$

dove

$$\overline{x}_j = \frac{1}{n}\sum_{i=1}^{n} x_{i,j} \qquad (j = 1, 2, \ldots, p). \tag{5.13}$$

è la media campionaria relativa alla variabile X_j (effettuata sulla colonna j-esima). Risulta

$$\alpha = \overline{y} - \beta_1 \overline{x}_1 - \beta_2 \overline{x}_2 - \ldots - \beta_p \overline{x}_p. \tag{5.14}$$

Utilizziamo ora la regressione lineare multipla in R, La funzione

$$\mathsf{lm}(\mathsf{y} \tilde{} \mathsf{x_1} + \mathsf{x_2} + \ldots + \mathsf{x_p})$$

è utilizzata per eseguire le analisi di regressione lineari multiple. L'argomento $\mathsf{y} \tilde{} \mathsf{x_1} + \mathsf{x_2} + \ldots \mathsf{x_p}$ passato alla funzione $\mathsf{lm}()$ indica che y *dipende da* $\mathsf{x_1}, \mathsf{x_2}, \ldots, \mathsf{x_p}$, ossia che $\mathsf{x_1}, \mathsf{x_2}, \ldots, \mathsf{x_p}$ sono le variabili indipendenti e y la variabile dipendente. Dapprima si specifica la variabile dipendente (y), quindi si specificano le variabili indipendenti, tra loro unite da un segno più, che indica l'*inclusione della variabile nel modello* (il segno più non corrisponde ad una addizione).

Riferendoci al data frame contenente i voti di Diploma, Programmazione 1 e Programmazione 2 di 15 studenti, supponiamo che Diploma e Programmazione 1 sono le variabili indipendenti e Programmazione 2 la variabile dipendente.

```
> lm(dfm$progr2~dfm$diploma+dfm$progr1)

Call:
lm(formula = dfm$progr2 ~ dfm$diploma + dfm$progr1)

Coefficients:
(Intercept)  dfm$diploma   dfm$progr1
   -1.42224      0.08751      0.76725
```

mostrano che l'intercetta $\alpha = -1.42224$ e i due regressori sono $\beta_1 = 0.08751$ e $\beta_2 = 0.76725$. Pertanto il modello di regressione multipla stimato è

$$y = -1.42224 + 0.08751\, x_1 + 0.76725\, x_2.$$

Osserviamo che i segni di entrambi i regressori β_1 e β_2 sono positivi; quindi i voti di Diploma e di Programmazione 1 hanno un effetto positivo sul voto di Programmazione 2 (all'aumentare del voto di Diploma e di Programmazione 1 aumenta anche il voto di Programmazione 2). Si nota anche che il regressore β_1 è prossimo allo zero indicando che il voto di Diploma non incide in modo significativo su Programmazione 2.

Nel caso multivariato gli attributi dell'oggetto creato con $\mathsf{lm}()$ sono gli stessi del caso bivariato. Da tale oggetto possiamo estrarre i singoli valori. Ad esempio,

```
> multiplelinearmodel<-lm(dfm$progr2~dfm$diploma+dfm$progr1)
>
> multiplelinearmodel$coefficients
(Intercept) dfm$diploma  dfm$progr1
-1.42224471  0.08751188  0.76724784
>
> multiplelinearmodel$coefficients[[1]]
[1] -1.422245
>
> multiplelinearmodel$coefficients[[2]]
```

```
[1] 0.08751188
>
> multiplelinearmodel$coefficients[[3]]
[1] 0.7672478
```

5.4.1 Residui

Una volta calcolati i valori dei coefficienti α e $\beta_1, \beta_2, \ldots, \beta_p$, è possibile osservare gli scostamenti (*residui*) tra le ordinate dei punti y_i (*valori osservati*) e i corrispondenti valori stimati

$$\widehat{y}_i = \alpha + \beta_1\, x_{i,1} + \beta_2\, x_{i,2} + \ldots + \beta_p\, x_{i,p} \qquad (i = 1, 2, \ldots, n) \tag{5.15}$$

ottenuti mediante la regressione lineare multipla.

Anche per la regressione lineare multipla, si ha che l*a media campionaria dei valori stimati coincide con la media campionaria dei valori osservati.* Infatti,

$$\begin{aligned}\frac{1}{n}\sum_{i=1}^{n}\widehat{y}_i &= \frac{1}{n}\sum_{i=1}^{n}(\alpha + \beta_1\, x_{i,1} + \beta_2\, x_{i,2} + \ldots + \beta_p\, x_{i,p}) \\ &= \alpha + \beta_1\,\overline{x}_1 + \beta_2\,\overline{x}_2 + \ldots + \beta_p\,\overline{x}_p \\ &= (\overline{y} - \beta_1\overline{x}_1 - \beta_2\overline{x}_2 - \ldots - \beta_p\overline{x}_p) + \beta_1\,\overline{x}_1 + \beta_2\,\overline{x}_2 + \ldots + \beta_p\,\overline{x}_p = \overline{y}.\end{aligned}$$

I residui

$$E_i = y_i - \widehat{y}_i = y_i - (\alpha + \beta_1\, x_{i,1} + \beta_2\, x_{i,2} + \ldots + \beta_p\, x_{i,p}) \qquad (i = 1, 2, \ldots, n) \tag{5.16}$$

mostrano di quanto si discostano i valori osservati dai valori stimati con la regressione lineare multipla.

La *media campionaria dei residui* $\overline{E}$ *è nulla*, ossia in media gli scostamenti positivi e negativi si compensano. Infatti, ricordando (5.16), risulta:

$$\overline{E} = \frac{1}{n}\sum_{i=1}^{n} E_i = \frac{1}{n}\sum_{i=1}^{n}\big(y_i - \widehat{y}_i\big) = \overline{y} - \frac{1}{n}\sum_{i=1}^{n}\widehat{y}_i = 0. \tag{5.17}$$

La varianza campionaria dei residui è

$$s_E^2 = \frac{1}{n-1}\sum_{i=1}^{n}\big(E_i - \overline{E}\big)^2 = \frac{1}{n-1}\sum_{i=1}^{n} E_i^2 \tag{5.18}$$

In R per calcolare il vettore dei valori stimati tramite regressione lineare multipla $(\widehat{y}_1, \widehat{y}_2, \ldots, \widehat{y}_n)$ si utilizza la funzione

$$\mathsf{fitted}(\mathsf{lm}(\mathsf{y} \tilde{}\, \mathsf{x_1} + \mathsf{x_2} + \ldots + \mathsf{x_p})$$

con y che dipende da $\mathsf{x_1}, \mathsf{x_2}, \ldots, \mathsf{x_p}$. Invece, per calcolare il vettore dei residui $(E_1, E_2, \ldots, E_n)$ si utilizza la funzione

$$\mathsf{resid}(\mathsf{lm}(\mathsf{y} \tilde{}\, \mathsf{x_1} + \mathsf{x_2} + \ldots + \mathsf{x_p})$$

Ad esempio, per il data frame dfm dei voti dei 15 studenti, le seguenti linee di codice

```
> stimemult<-fitted(lm(dfm$progr2~dfm$diploma+dfm$progr1))
> stimemult # visualizza il vettore delle stime
       1        2        3        4        5        6        7
25.04280 26.75232 30.34638 25.63502 29.40411 27.16952 21.53625
       8        9       10       11       12       13       14
 28.61650 28.04463 28.02428 19.65170 22.12847 27.01485 29.90882
      15
28.72437
```

determinano i valori stimati $(\widehat{y}_1, \widehat{y}_2, \ldots, \widehat{y}_{15})$. Ad esempio, il primo valore del vettore stime può essere così estratto:

```
> multiplelinearmodel<-lm(dfm$progr2~dfm$diploma+dfm$progr1)
> multiplelinearmodel$fitted.values[[1]]
[1] 25.0428
```

e corrisponde a

$$\widehat{y}_1 = \alpha + \beta_1\, x_{1,1} + \beta_2\, x_{1,2} = -1.422245 + 0.08751188 \cdot 92 + 0.7672478 \cdot 24 = 25.0428,$$

dove 92 è il voto di diploma del primo studente e 24 è il voto di Programmazione 1 del primo studente.

Inoltre, le seguenti linee di codice

```
> residuimult<-resid(lm(dfm$progr2~dfm$diploma+dfm$progr1))
> residuimult # visualizza i residui
          1           2           3           4           5
 1.95720388 -0.75231555 -1.34637817  0.36497980  0.59589343

          6           7           8           9          10
-0.16951588  0.46375463  0.38350032 -1.04463465 -0.02427559
         11          12          13          14          15
 0.34829781 -2.12846946 -0.01485118  0.09118122  1.27562939
```

determinano i residui $(E_1, E_2, \ldots, E_{15})$. Ad esempio, il primo valore del vettore residui può essere così estratto:

```
> multiplelinearmodel<-lm(dfm$progr2~dfm$diploma+dfm$progr1)
> multiplelinearmodel$residuals[[1]]
[1] 1.957204
```

e corrisponde a

$$E_1 = y_1 - \widehat{y}_1 = 27 - 25.0428 = 1.9572,$$

dove 27 è il voto di Programmazione 2 del primo studente.

Abbiamo mostrato che la media dei residui è nulla. Invece, la mediana, la varianza campionaria e la deviazione standard campionaria dei residui sono:

```
> median(multiplelinearmodel$residuals)
[1] 0.09118122
>
> var(multiplelinearmodel$residuals)
[1] 1.033417
>
> sd(multiplelinearmodel$residuals)
[1] 1.016571
```

Anche nel caso multivariato è interessante calcolare i *residui standardizzati* così definiti:

$$E_i^{(s)} = \frac{E_i - \overline{E}}{s_E} = \frac{E_i}{s_E},$$

che risultano essere caratterizzati da media campionaria nulla e varianza unitaria. Ad esempio, per il data frame dei voti dei 15 studenti risulta:

```
> multiplelinearmodel<-lm(dfm$progr2~dfm$diploma+dfm$progr1)
> residuimult<-resid(lm(dfm$progr2~dfm$diploma+dfm$progr1))
> residuimultstandard<-residuimult/sd(residuimult) visualizza il
    vettore dei residui standardizzati
         1          2          3          4          5          6
 1.92529890 -0.74005182 -1.32443044  0.35903015  0.58617959
     -0.16675255
         7          8          9         10         11         12
 0.45619482  0.37724876 -1.02760574 -0.02387987  0.34262010
     -2.09377263
        13         14         15
-0.01460909 0.08969485  1.25483496
```

È poi possibile realizzare un grafico in cui i residui standardizzati (ordinate) vengono disegnati in funzione dei valori stimati (ascisse) con il metodo dei minimi quadrati. Il seguente codice

```
> stimemult<-fitted(lm(dfm$progr2~dfm$diploma+dfm$progr1))
> residuimult<-resid(lm(dfm$progr2~dfm$diploma+dfm$progr1))
> residuimultstandard<-residuimult/sd(residuimult)
> plot(stimemult,residuimultstandard, main="Residui standard
+ rispetto ai valori stimati",
+ xlab="Valori stimati",ylab="Residui standard",pch=5,col="red")
> abline(h=0,col="blue",lty=2)
```

produce il grafico in Figura 5.8. I punti indicano la posizione dove si collocano i residui standardizzati rispetto ai valori stimati con la retta di regressione. La retta orizzontale è posizionata nello zero, che corrisponde alla media campionaria dei residui standardizzati. Anche in questo caso i punti sono disposti quasi casualmente attorno alla linea orizzontale e non si evidenzia nessuna tendenza particolare nella distribuzione dei punti.

5.4.2 Coefficiente di determinazione

Il coefficiente di determinazione in un modello di regressione lineare multipla è il rapporto tra la varianza dei valori stimati tramite la funzione di regressione multipla e la varianza i valori osservati della variabile dipendente. Se si denota con $(y_1, y_2, \ldots, y_n)$ il vettore dei dati della variabile dipendente, con $\overline{y}$ la sua media campionaria e con $(\widehat{y}_1, \widehat{y}_2, \ldots, \widehat{y}_n)$ i valori stimati attraverso la funzione di regressione (la cui media campionaria è $\overline{y}$), il coefficiente di determinazione è:

$$D^2 = \frac{\frac{1}{n-1}\sum_{i=1}^{n}(\widehat{y}_i - \overline{y})^2}{\frac{1}{n-1}\sum_{i=1}^{n}(y_i - \overline{y})^2} = \frac{\sum_{i=1}^{n}(\widehat{y}_i - \overline{y})^2}{\sum_{i=1}^{n}(y_i - \overline{y})^2}. \tag{5.19}$$

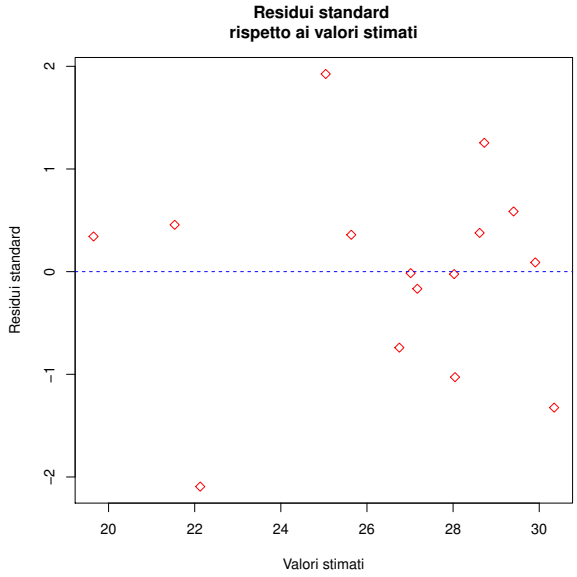

Figura 5.8: Residui in funzione dei valori stimati.

L'indice D^2 è adimensionale e risulta $0 \leq D^2 \leq 1$. Quando $D^2 = 0$ il modello di regressione multipla utilizzato non spiega per nulla i dati. Invece, quando $D^2 = 1$ il modello di regressione multipla utilizzato spiega perfettamente i dati. In R per calcolare l'indice D^2 per la regressione lineare multipla basta utilizzare la funzione:

$$\mathsf{summary(lm(y \tilde{} x_1 + x_2 + \ldots + x_p))\$r.square}$$

Riferendoci all'esempio dei voti di Diploma, Programmazione 1 e Programmazione 2 calcoliamo D^2 in due diversi modi:

```
> num<-sum((stimemult-mean(dfm$progr2))^2)
> den<-sum((dfm$progr2-mean(dfm$progr2))^2)
> d2<-num/den
> d2
[1] 0.9082766
>
> summary(lm(dfm$progr2~dfm$diploma+dfm$progr1))$r.square
[1] 0.9082766
```

Il coefficiente di determinazione è quindi 0.9082766, ossia il modello di regressione multipla utilizzato può spiegare significativamente i dati. Inoltre, non si è ottenuto un significativo miglioramento di tale indice rispetto al modello di regressione semplice in cui $D^2 = 0.8994429$.

5.5 Regressione non lineare

Spesso, osservando uno scatterplot, si nota che l'ipotesi di linearità di un modello non è accettabile poiché i dati sperimentali non evidenziano una correlazione di tipo lineare. In questo caso occorre ricorrere a modelli di regressione non lineare. Vediamo come calcolare il coefficiente di determinazione in tali modelli.

5.5.1 Coefficiente di determinazione

Il coefficiente di determinazione per regressioni non lineari è stato definito in letteratura in tre modi differenti:

$$\text{Definizione 1}: \qquad D_1^2 = 1 - \frac{\sum_{i=1}^n (y_i - \widehat{y}_i)^2}{\sum_{i=1}^n (y_i - \overline{y})^2},$$

$$\text{Definizione 2}: \qquad D_2^2 = \frac{\sum_{i=1}^n (\widehat{y}_i - \overline{y})^2}{\sum_{i=1}^n (y_i - \overline{y})^2},$$

$$\text{Definizione 3}: \qquad D_3^2 = \frac{\sum_{i=1}^n (\widehat{y}_i - \overline{\widehat{y}})^2}{\sum_{i=1}^n (y_i - \overline{y})^2},$$

dove

- $y_1, y_2, \ldots, y_n$ sono i valori osservati;
- $\widehat{y}_1, \widehat{y}_2, \ldots, \widehat{y}_n$ sono i valori stimati tramite la regressione;
- $\overline{y}$ è la media campionaria dei valori osservati $y_1, y_2, \ldots, y_n$
- $\overline{\widehat{y}}$ è la media campionaria dei valori stimati $\widehat{y}_1, \widehat{y}_2, \ldots, \widehat{y}_n$.

Nel caso di regressione lineare semplice e multipla abbiamo precedentemente mostrato che la media campionaria dei valori stimati coincide con la media campionaria dei valori osservati e quindi $D_2^2 = D_3^2$. Inoltre, in Jones[1] si mostra che nel caso di regressione lineare semplice e multipla risulta $D_1^2 = D_2^2 = D_3^2$, ossia le tre definizioni conducono allo stesso risultato. Invece, nel caso di regressioni non lineari le tre definizioni di coefficiente di determinazione possono condurre a risultati differenti. In particolare, D_2^2 e D_3^2 possono essere anche maggiori di 1, mentre $0 \leq D_1^2 \leq 1$. Pertanto, *per regressioni non lineari si preferisce utilizzare come definizione di coefficiente di determinazione* D_1^2. Quando $D_1^2 = 0$ il modello di regressione non lineare utilizzato non spiega per nulla i dati. Invece, quando $D_1^2 = 1$ il modello di regressione non lineare utilizzato spiega perfettamente i dati.

[1] A.R. Jones, Best fit lines & curves and some mathe-magical transformations, Taylor & Francis Group, London (2019)

5.5.2 Modelli linearizzabili

In alcuni casi, modelli che sembrano non lineari sono linearizzabili attraverso opportune trasformazioni. Ad esempio, come vedremo nel seguito, sono da considerare linearizzabili le seguenti funzioni:

$$\begin{array}{ll} (i) & Y = \alpha + \beta\, X + \gamma\, X^2, \qquad (\textit{regressione quadratica}) \\ (ii) & Y = \alpha + \beta\, e^X, \qquad (\textit{regressione esponenziale}) \\ (iii) & Y = e^{\alpha+\beta\, X}, \qquad (\textit{regressione semilogaritmica}) \\ (iv) & Y = \alpha_0\, X^\beta \qquad (\alpha_0 > 0) \qquad (\textit{regressione logaritmica}). \end{array}$$

Nel seguito, analizziamo i quattro tipi di regressione non lineare precedentemente indicati.

(i) Regressione quadratica

Il modello non lineare più semplice è il modello polinomiale del secondo ordine:

$$Y = \alpha + \beta\, X + \gamma\, X^2. \tag{5.20}$$

Ad esempio, consideriamo il seguente data frame dove Y è la variabile dipendente e X la variabile indipendente:

```
> dfp<-data.frame(x=c(0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1.0,
+ 1.1,1.2,1.3,1.4,1.5),
+ y=c(0.74,0.46,0.2,0.1,-0.1,-0.15,-0.2, -0.12,0.0,0.5,
+ 0.2,0.3,0.7, 0.6,0.9))
>
> dfp # visualizza il data frame
     x     y
1  0.1  0.74
2  0.2  0.46
3  0.3  0.20
4  0.4  0.10
5  0.5 -0.10
6  0.6 -0.15
7  0.7 -0.20
8  0.8 -0.12
9  0.9  0.00
10 1.0  0.50
11 1.1  0.20
12 1.2  0.30
13 1.3  0.70
14 1.4  0.60
15 1.5  0.90
```

Visualizziamo i punti in uno scatterplot. La seguente linea di codice

```
> plot(dfp$x,dfp$y, col="red",main="Scatterplot")
```

fornisce lo scatterplot di Figura 5.9.

Osservando lo scatterplot di Figura 5.9 si nota che un modello lineare non può approssimare efficacemente i dati. Il coefficiente di correlazione lineare

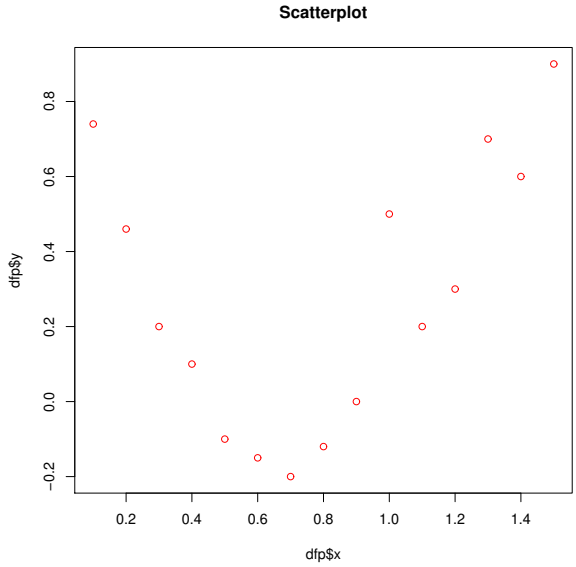

Figura 5.9: Scatterplot.

```
> cor(dfp$x,dfp$y)
[1] 0.3413009
```

fornisce un valore positivo basso e il coefficiente di determinazione è $D^2 = (0.3413009)^2 = 0.1164863$.

Utilizziamo quindi il modello di regressione polinomiale del secondo ordine (5.20). Per la stima dei parametri α, β e γ si può ricorrere alla regressione multipla

$$Y = \alpha + \beta X_1 + \gamma X_2$$

con intercetta α e regressori β e γ per le variabili $X_1 = X$ e $X_2 = X^2$.

Con R è facile stimare i parametri α, β, γ tramite la funzione

$$\mathsf{lm(y \tilde{} x + I(x \wedge 2))}$$

dove $I()$ è un *identificatore di variabile* e viene inserito quando si debbono effettuare operazioni matematiche (divisione, elevamento a potenza) nelle variabili della regressione.

Vediamo se l'approssimazione polinomiale del secondo ordine è adeguata per stimare i nostri dati. Le seguenti linee di codice

```
> pol2<-lm(dfp$y~dfp$x+I((dfp$x)^2))
> pol2 # visualizza i coefficienti stimati

Call:
```

```
lm(formula = dfp$y ~ dfp$x + I((dfp$x)^2))

Coefficients:
 (Intercept)          dfp$x   I((dfp$x)^2)
      0.8524        -2.5355         1.7557
```

mostrano che $\alpha = 0.8524$, $\beta = -2.5355$ e $\gamma = 1.7557$. Il modello polinomiale del secondo ordine è quindi

$$Y = 0.8524 - 2.5355\, X + 1.7557\, X^2$$

Per poter recuperare i parametri procediamo nel seguente modo:

```
> alpha<-pol2$coefficients[[1]]
> beta<-pol2$coefficients[[2]]
> gamma<-pol2$coefficients[[3]]
> c(alpha,beta,gamma)
[1]  0.8523736 -2.5354783  1.7556561
```

In questo caso il coefficiente di determinazione D^2 può essere determinato utilizzando una qualsiasi delle tre definizioni oppure la funzione r.square ottenendo lo stesso risultato:

```
%
> stime<-alpha+beta*dfp$x+gamma*(dfp$x)^2
>
> # Definizione 1
> num1<-sum(((dfp$y-stime)^2))
> den1<-sum(((dfp$y-mean(dfp$y))^2))
> d2<-1-num1/den1
> d2
[1] 0.8233134
> # Definizione 2
> num<-sum((stime-mean(dfp$y))^2)
> den<-sum((dfp$y-mean(dfp$y))^2)
> d2<-num/den
> d2
[1] 0.8233134
> # Definizione 3
> num<-sum((stime-mean(stime))^2)
> den<-sum((dfp$y-mean(dfp$y))^2)
> d2<-num/den
> d2
[1] 0.8233134
>
> summary(lm(dfp$y~dfp$x+I((dfp$x)^2)))$r.square
[1] 0.8233134
```

ossia $D^2 = 0.8233134$. Si è ottenuto un coefficiente di determinazione migliore rispetto a quello del modello lineare $D^2 = 0.1164863$.

Disegniamo ora la curva stimata sullo scatterplot di Figura 5.9. Le seguenti linee di codice

```
> plot(dfp$x,dfp$y, col="red",main="Scatterplot e curva stimata")
> curve(alpha+beta*x+gamma*x^2, add=TRUE)
```

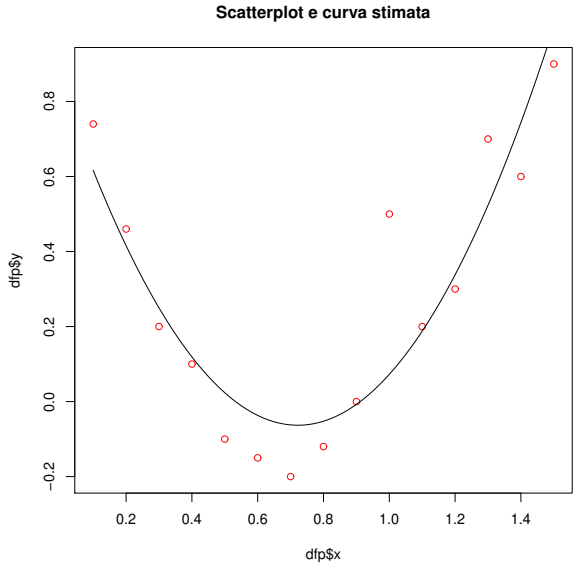

Figura 5.10: Rappresentazione dello scatterplot e della curva stimata $y = \alpha + \beta\, x + \gamma\, x^2$.

producono il grafico in Figura 5.10.

Il metodo utilizzato può essere esteso anche a polinomi di ordine p qualsiasi utilizzando una regressione lineare multipla per la stima dell'intercetta e dei regressori.

(ii) Regressione con funzione esponenziale

Consideriamo il modello non lineare

$$Y = \alpha + \beta\, e^X \tag{5.21}$$

Ad esempio, consideriamo il seguente data frame con Y variabile dipendente e X variabile indipendente:

```
> dfe<-data.frame(x=c(0.25,0.5,0.75,1.0,1.25,1.5,1.75,2.0,2.25,2.5,
+ 2.75,3.0,3.25,3.5,3.75,4.0),
+ y=c(5,7,8,10,14,18,23,29,37,49,63,81,
+ 104,133,169,219))
> dfe
      x   y
1  0.25   5
2  0.50   7
3  0.75   8
4  1.00  10
5  1.25  14
6  1.50  18
```

```
7   1.75   23
8   2.00   29
9   2.25   37
10  2.50   49
11  2.75   63
12  3.00   81
13  3.25  104
14  3.50  133
15  3.75  169
16  4.00  219
```

Visualizziamo i punti in uno scatterplot. La seguente linea di codice

```
> plot(dfe$x,dfe$y, col="red",main="Scatterplot")
```

fornisce lo scatterplot di Figura 5.11. Osservando lo scatterplot di Figura 5.11 si

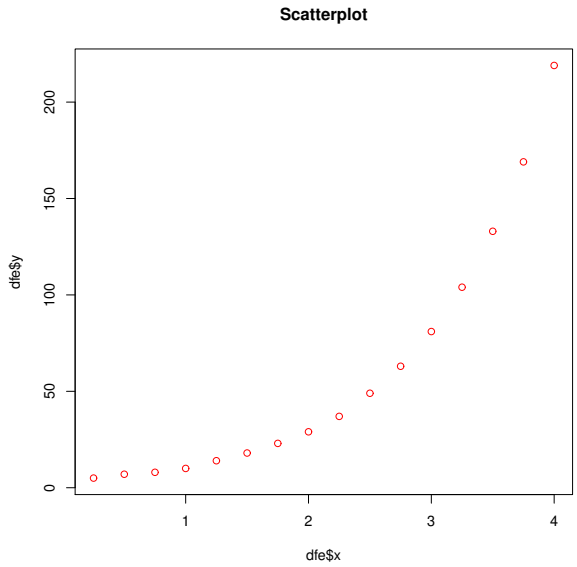

Figura 5.11: Scatterplot.

nota che un modello lineare non approssima perfettamente i dati. Il coefficiente di correlazione lineare

```
> cor(dfe$x,dfe$y)
[1] 0.9001635
```

fornisce un valore positivo e il coefficiente di determinazione è $D^2 = (0.9001635)^2 = 0.8102943$.

Utilizziamo ora il modello di regressione esponenziale (5.21). Per la stima dei parametri α e β si può ricorrere alla regressione lineare

$$Y = \alpha + \beta X_1$$

con intercetta α e regressore β associato alla variabile $X_1 = e^X$.

Vediamo se l'approssimazione con funzione esponenziale (5.21) è più adeguata per stimare i nostri dati. Le seguenti linee di codice

```
> lexp<-lm(dfe$y~I(exp(dfe$x)))
> lexp # visualizza i coefficienti stimati

Call:
lm(formula = dfe$y ~ I(exp(dfe$x)))

Coefficients:
  (Intercept)  I(exp(dfe$x))
      -0.1599          4.0096
```

mostrano che $\alpha = -0.1599$ e $\beta = 4.0096$. Il modello è quindi

$$Y = -0.1599 + 4.0096\, e^X.$$

Per poter recuperare i parametri procediamo nel seguente modo:

```
> alpha<-lexp$coefficients[[1]]
> beta<-lexp$coefficients[[2]]
> c(alpha,beta)
[1] -0.1599231  4.0096167
```

Anche in questo caso il coefficiente di determinazione D^2 può essere determinato utilizzando una qualsiasi delle tre definizioni oppure la funzione r.square ottenendo lo stesso risultato:

```
> stime<-alpha+beta*exp(dfe$x)
>
> # Definizione 1
> num1<-sum(((dfe$y-stime)^2))
> den1<-sum(((dfe$y-mean(dfe$y))^2))
> d2<-1-num1/den1
> d2
[1] 0.9999164
> # Definizione 2
> num<-sum((stime-mean(dfe$y))^2)
> den<-sum((dfe$y-mean(dfe$y))^2)
> d2<-num/den
> d2
[1] 0.9999164
> # Definizione 3
> num<-sum((stime-mean(stime))^2)
> den<-sum((dfe$y-mean(dfe$y))^2)
> d2<-num/den
> d2
[1] 0.9999164
>
> summary(lm(dfe$y~I(exp(dfe$x))))$r.square
[1] 0.9999164
```

che mostra che $D^2 = 0.9999164$ è prossimo all'unità. Si nota che si è ottenuto un coefficiente di determinazione migliore rispetto a quello del modello lineare $D^2 = 0.8102943$.

Disegniamo ora la curva stimata sullo scatterplot di Figura 5.11. Le seguenti linee di codice

```
> plot(dfe$x,dfe$y, col="red",main="Scatterplot")
> curve(alpha+beta*exp(x), add=TRUE)
```

producono il grafico in Figura 5.12.

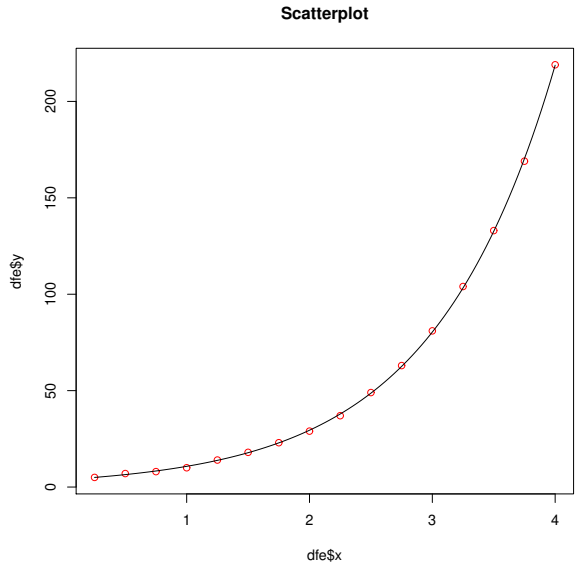

Figura 5.12: Rappresentazione dello scatterplot e della curva stimata $y = \alpha + \beta\, e^x$

(iii) Regressione con trasformazione semilogaritmica

Alcuni modelli sono non lineari, ma possono essere linearizzati mediante un'opportuna trasformazione. Consideriamo il modello non lineare

$$Y = e^{\alpha+\beta\, X}. \tag{5.22}$$

Applicando il logaritmo ad entrambi i membri, si ottiene:

$$\log(Y) = \alpha + \beta\, X$$

Ad esempio, consideriamo il seguente data frame con Y la variabile dipendente e X la variabile indipendente:

```
> dfex<-data.frame(x=c(0.25,0.5,0.75,1.0,1.25,1.5,1.75,2.0,2.25,
+ 2.5,2.75,3.0),
+ y=c(1.1,0.6,0.3,0.2,0.14,0.07,0.05,0.03,0.015,
```

```
+ 0.01,0.006,0.004))
> dfex
      x     y
1  0.25 1.100
2  0.50 0.600
3  0.75 0.300
4  1.00 0.200
5  1.25 0.140
6  1.50 0.070
7  1.75 0.050
8  2.00 0.030
9  2.25 0.015
10 2.50 0.010
11 2.75 0.006
12 3.00 0.004
```

Visualizziamo i punti in uno scatterplot. La seguente linea di codice

```
> plot(dfex$x,dfex$y, col="red",main="Scatterplot")
```

fornisce lo scatterplot di Figura 5.13

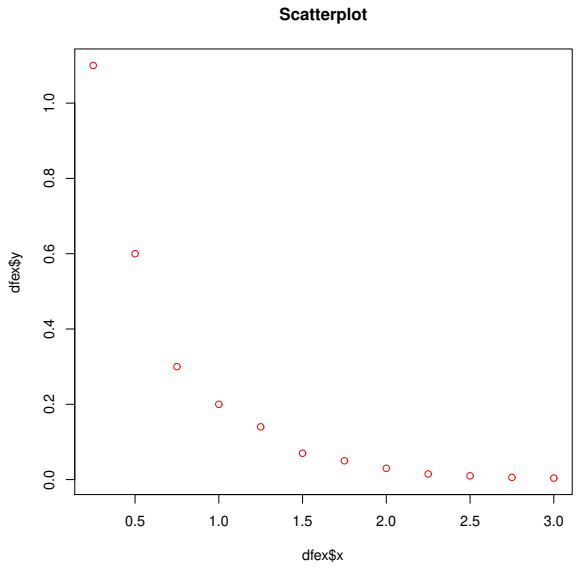

Figura 5.13: Scatterplot.

Osservando lo scatterplot di Figura 5.13 si nota che un modello lineare non approssima bene i dati. Il coefficiente di correlazione lineare

```
> cor(dfex$x,dfex$y)
[1] -0.7913646
```

fornisce un valore negativo e il coefficiente di determinazione è $D^2 = (-0.7913646)^2 = 0.626258$.

Consideriamo quindi il modello di regressione semilogaritmica (5.22). Per la stima dei parametri α e β si può ricorrere alla regressione lineare nel seguente modo:

```
> lex<-lm(I(log(dfex$y))~dfex$x)
> lex # visualizza i coefficienti stimati

Call:
lm(formula = I(log(dfex$y)) ~ dfex$x)

Coefficients:
(Intercept)        dfex$x
     0.4777       -2.0273
```

mostrano che $\alpha = 0.4777$ e $\beta = -2.0273$. Il modello è quindi

$$Y = e^{0.4777 - 2.0273\, X}.$$

Per poter recuperare i parametri procediamo nel seguente modo:

```
> alpha<-lex$coefficients[[1]]
> beta<-lex$coefficients[[2]]
> c(alpha,beta)
[1]  0.4777082 -2.0272523
```

Disegniamo ora la curva stimata sullo scatterplot di Figura 5.13. Le seguenti linee di codice

```
> plot(dfe$x,dfe$y, col="red",main="Scatterplot")
> curve(exp(alpha+beta*x), add=TRUE)
```

producono il grafico in Figura 5.14.

Per il modello con regressione semilogaritmica, il coefficiente di determinazione può essere calcolato utilizzando la Definizione 1:

```
> stime<-exp(alpha+beta*dfex$x)
> # Definizione 1
> num1<-sum(((dfex$y-stime)^2))
> den1<-sum(((dfex$y-mean(dfex$y))^2))
> d2<-1-num1/den1
> d2
[1] 0.9833706
```

Si è ottenuto un coefficiente di determinazione $D^2 = 0.9833706$ migliore rispetto a quello del modello lineare $D^2 = 0.626258$.

(iv) Regressione con trasformazione logaritmica

Consideriamo ora il modello non lineare di tipo geometrico

$$Y = \alpha_0\, X^{\beta} \qquad (\alpha_0 > 0) \tag{5.23}$$

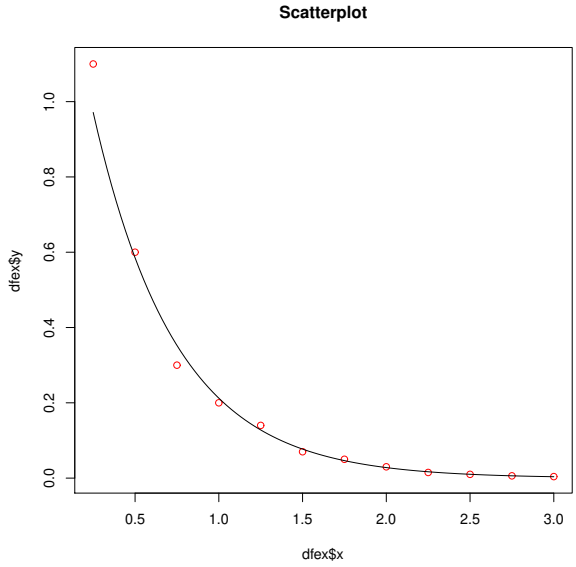

Figura 5.14: Rappresentazione dello scatterplot e della curva stimata $y = e^{\alpha+\beta\,x}$.

Applicando il logaritmo ad entrambi i membri, si ottiene:

$$\log(Y) = \log(\alpha_0) + \beta \log(X)$$

Ad esempio, consideriamo il seguente data frame con Y la variabile dipendente e X la variabile indipendente:

```
> dfg<-data.frame(x=c(0.25,0.5,0.75,1.0,1.25,1.5,1.75,2.0,2.25,2.5,
+ 2.75,3.0,3.25,3.5,3.75,4.0),
+  y=c(0.05,0.2,0.4,1.0,1.5,2.5,3.1,3.9,5.2,6.0,7.0,8.5,
+ 10,12.5,13.5,16.5))
> dfg # visualizza il data frame
      x     y
1  0.25  0.05
2  0.50  0.20
3  0.75  0.40
4  1.00  1.00
5  1.25  1.50
6  1.50  2.50
7  1.75  3.10
8  2.00  3.90
9  2.25  5.20
10 2.50  6.00
11 2.75  7.00
12 3.00  8.50
13 3.25 10.00
14 3.50 12.50
15 3.75 13.50
```

```
16  4.00  16.50
```

Visualizziamo i punti in uno scatterplot. La seguente linea di codice

```
> plot(dfg$x,dfg$y, col="red",main="Scatterplot")
```

fornisce lo scatterplot di Figura 5.15

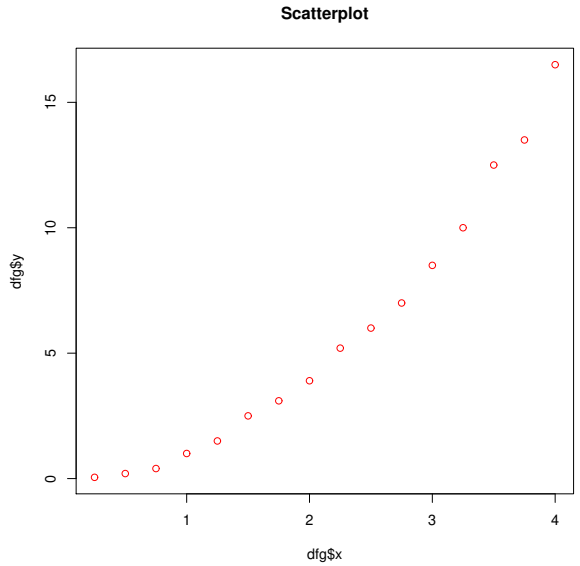

Figura 5.15: Scatterplot.

Osservando lo scatterplot di Figura 5.13 si nota che un modello lineare non approssima perfettamente i dati. Il coefficiente di correlazione lineare

```
> cor(dfg$x,dfg$y)
[1] 0.9669876
```

fornisce un valore positivo sufficientemente alto e il coefficiente di determinazione è $D^2 = (0.9669876)^2 = 0.935065$.

Utilizziamo il modello di regressione basato su (5.23). Per la stima dei parametri α, β si può ricorrere alla regressione geometrica nel seguente modo:

```
> lgeom<-lm(I(log(dfg$y))~I(log(dfg$x)))
> lgeom # visualizza i coefficienti stimati

Call:
lm(formula = I(log(dfg$y)) ~ I(log(dfg$x)))

Coefficients:
  (Intercept)  I(log(dfg$x))
      -0.1075         2.0932
```

mostrano che $\alpha = \log(\alpha_0) = -0.1075$ e $\beta = 2.0932$. Il modello è quindi

$$Y = e^{-0.1075}\, x^{2.0932} = 0.8980765\, x^{2.0932}$$

Per poter recuperare i parametri procediamo nel seguente modo:

```
> alpha0<-exp(lgeom$coefficients[[1]])
> beta<-lgeom$coefficients[[2]]
> c(alpha0,beta)
[1] 0.8980562 2.0932138
```

Disegniamo ora la curva stimata sullo scatterplot di Figura 5.15. Le seguenti linee di codice

```
> plot(dfg$x,dfg$y, col="red",main="Scatterplot")
> curve(alpha0*x^beta, add=TRUE)
```

producono il grafico in Figura 5.16.

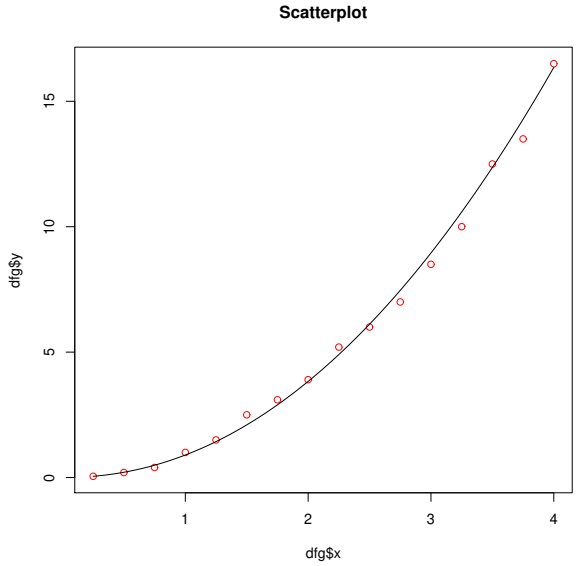

Figura 5.16: Rappresentazione dello scatterplot e della curva stimata $y = \alpha_0\, x^{\beta}$.

Per il modello con regressione logaritmica, il coefficiente di determinazione può essere calcolato utilizzando la Definizione 1:

```
> stime<-alpha0*((dfg$x)^beta)
>
>  # Definizione 1
> num1<-sum(((dfg$y-stime)^2))
> den1<-sum(((dfg$y-mean(dfg$y))^2))
> d2<-1-num1/den1
> d2
[1] 0.9956416
```

Si è ottenuto un coefficiente di determinazione $D^2 = 0.9956416$ migliore rispetto a quello del modello lineare $D^2 = 0.935065$.

Come già visto nella regressione lineare semplice, anche nella regressione non lineare è possibile rappresentare graficamente i residui:

- tracciando dei segmenti verticali che congiungono i valori stimati $\widehat{y}_i$ (che giacciono sulla curva di regressione) e i valori osservati y_i $(i = 1, 2, \ldots, n)$;

- rappresentando i valori dei residui E_i rispetto alle osservazioni x_i (variabile indipendente) $(i = 1, 2, \ldots, n)$;

- rappresentando i residui standardizzati $E_i^{(s)} = (E_i - \overline{E})/s_E$ rispetto ai valori stimati $\widehat{y}_i$ $(i = 1, 2, \ldots, n)$.

Occorre sottolineare che nel caso di una regressione non lineare la media dei valori stimati può non coincidere con la media dei valori osservati e la media di residui può non essere nulla.

Il metodo di linearizzazione può essere applicato in molti altri casi. Nella Tabella 5.2 sono elencate alcune tra le funzioni più comunemente utilizzate nella regressione non lineare e sono indicati i corrispondenti cambiamenti di variabili necessari per applicare il metodo di linearizzazione.

Tabella 5.2: Alcune funzioni linearizzabili.

Funzione $y = f(x)$	Forma linearizzata $Y = \alpha + \beta\, X$	Cambiamenti di variabili e costanti
$y = C \cdot x^{\beta}$	$\log y = \log C + \beta\, \log x$	$X = \log x, \quad Y = \log y$ $\alpha = \log C$
$y = C \cdot e^{\beta x}$	$\log y = \log C + \beta \cdot x$	$X = x, \quad Y = \log y$ $\alpha = \log C$
$y = \alpha + \beta \cdot \log x$	$y = \alpha + \beta \cdot \log x$	$X = \log x, \quad Y = y$
$y = \frac{\beta}{x} + \alpha$	$y = \alpha + \beta \cdot \frac{1}{x}$	$X = \frac{1}{x}, \quad Y = y$
$y = \frac{H}{C\,x+D}$	$\frac{1}{y} = \frac{D}{H} + \frac{C}{H}\, x$	$X = x, \quad Y = \frac{1}{y}$ $\alpha = \frac{D}{H}, \qquad \beta = \frac{C}{H}$
$y = \frac{x}{\beta+\alpha\, x}$	$\frac{1}{y} = \alpha + \beta\, \frac{1}{x}$	$X = \frac{1}{x}, \quad Y = \frac{1}{y}$
$y = \frac{1}{\alpha+\beta\, e^{-x}}$	$\frac{1}{y} = \alpha + \beta\, e^{-x}$	$X = e^{-x}, \quad Y = \frac{1}{y}$

Capitolo 6

Analisi dei cluster: parte 1

6.1 Introduzione

L'analisi dei cluster è una metodologia che permette di raggruppare in sottoinsiemi, detti cluster, entità (unità) appartenenti ad un insieme più ampio. L'insieme originario delle entità su cui si attua l'analisi per ricavare i cluster non è sottoposto ad alcuna restrizione. Può infatti contenere variabili, individui, osservazioni, dati, misure, ...

I vari metodi attraverso cui si attua l'analisi dei cluster hanno in comune uno scopo generale: ottenere raggruppamenti in base alla somiglianza in modo che gli elementi di uno stesso gruppo siano tra loro il più possibile simili e gli elementi appartenenti a gruppi distinti siano tra loro il più possibile diversi. In altre parole tale analisi ha lo scopo di distribuire le osservazioni in gruppi in modo tale che il grado di *naturale associazione* sia alto tra i membri dello stesso gruppo e basso tra i membri di gruppi diversi. In questo modo si otterrà quindi un'alta omogeneità all'interno dei gruppi e un'alta eterogeneità tra gruppi distinti.

L'analisi dei cluster si può applicare in tutti i casi in cui è richiesto riunire in maniera non intuitiva, con il sostegno di metodi quantitativi, dati di qualsiasi natura. Lo scopo è quello di organizzare in una struttura un insieme di dati e di scoprire quali siano i legami esistenti tra essi. Alcune discipline scientifiche che utilizzano l'analisi dei cluster sono: l'*informatica* e l'*ingegneria* (per creare delle classificazioni o formare delle categorie), le *scienze naturali* (per affrontare problemi di tassonomia, per descrivere l'ecologia di comunità naturali), la *medicina* (come ausilio nella diagnosi dei quadri clinici, nelle previsioni sulle malattie di individui o popolazioni e a scopo di diagnosi), l'*economia* (per classificare regioni e identificare aree omogenee sulla base di particolari indici), l'*archeologia*, la *linguistica*, le *scienze sociali*. La flessibilità e la varietà dei metodi di analisi dei cluster consente il raggiungimento di scopi diversi in funzione del metodo usato, dei dati a disposizione, del campo di ricerca ed infine degli interessi del ricercatore.

I metodi di analisi dei cluster nel loro complesso permettono di raggiungere i seguenti obiettivi:

- individuazione di una reale tipologia;
- previsioni basate su gruppi;
- esplorazione dei dati;
- generazione di ipotesi di ricerca;
- verifica di ipotesi;
- riduzione della complessità dei dati.

In molti campi di studio il ricercatore ha a disposizione un *grande numero di osservazioni* che sono praticamente intrattabili a meno che non siano classificate in gruppi che in qualche senso possono essere considerati come singole unità. In tal caso possono essere usate tecniche di clustering per effettuare questa *riduzione dei dati*, permettendo così di trasformare le informazioni dall'insieme completo di n individui all'informazione circa m gruppi di individui (ovviamente m deve essere molto più piccolo di n). In questo modo è possibile fornire una *più concisa* e *più comprensibile* descrizione delle osservazioni considerate. In altre parole, occorre ricercare delle *semplificazioni del problema originario con la minima perdita di informazione*. L'analisi dei cluster può anche essere usata per *generare ipotesi* sulla natura dei dati. In questo caso occorre successivamente *verificare le ipotesi* fatte e ogni test deve essere effettuato su nuove osservazioni senza usare i dati da cui le ipotesi sono state generate. In alcuni campi di indagine scientifica si possono usare i metodi dell'analisi dei cluster per *produrre gruppi* che formano la base di uno schema di classificazione utile in studi successivi per *scopi di previsioni* di un qualche tipo.

6.2 Nozioni di base e definizioni

Sia $I = \{I_1, I_2, \ldots, I_n\}$ un insieme di n individui (entità o unità) appartenenti ad una popolazione. Assumiamo che esista un *insieme di caratteristiche* (*features*) $C = \{C_1, C_2, \ldots, C_p\}$ che sono osservabili e sono possedute da ogni individuo in I. Il termine *osservabile* denota caratteristiche che danno origine a dati sia di *tipo qualitativo* che di *tipo quantitativo* (detti anche misure).

Denotiamo con il simbolo x_{ij} il valore della misura della caratteristica j–esima relativa all'individuo I_i e con $X_i = (x_{i1}, x_{i2}, \ldots, x_{ip}) \quad (i = 1, 2, \ldots, n)$ il vettore di cardinalità $1 \times p$ di tali misure. Quindi, in generale l'iniziale naturale collezione dei dati su cui il ricercatore deve operare consiste di un insieme di n vettori di misure $\{X_1, X_2, \ldots, X_n\}$ che descrive l'insieme I degli individui a cui è associata una matrice di misure X di cardinalità $n \times p$, ossia

$$X = \begin{pmatrix} x_{11} & x_{12} & \ldots & x_{1p} \\ x_{21} & x_{22} & \ldots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \ldots & x_{np} \end{pmatrix} = \begin{pmatrix} X_1 \\ X_2 \\ \vdots \\ X_n \end{pmatrix}, \tag{6.1}$$

dove $X_i = (x_{i1}, x_{i2}, \ldots, x_{ip})$ $(i = 1, 2, \ldots, n)$. Nella Tabella 6.1 sono indicati i dati osservati per gli n individui, le medie campionarie $\overline{x}_j$ e le varianze campionarie s_j^2 della j-esima caratteristica effettuate su tutti gli n individui:

$$\overline{x}_j = \frac{1}{n}\sum_{i=1}^{n} x_{ij}, \qquad s_j^2 = \frac{1}{n-1}\sum_{i=1}^{n}\big(x_{ij} - \overline{x}_j\big)^2 \qquad (j = 1, 2, \ldots, p). \tag{6.2}$$

Tabella 6.1: Dati osservati per n individui.

	x_{11}	x_{12}	$\ldots$	x_{1j}	$\ldots$	x_{1p}
	x_{21}	x_{22}	$\ldots$	x_{2j}	$\ldots$	x_{2p}
	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$
	x_{i1}	x_{i2}	$\ldots$	x_{ij}	$\ldots$	x_{ip}
	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$
	x_{n1}	x_{n2}	$\ldots$	x_{nj}	$\ldots$	x_{np}
mean	$\overline{x}_1$	$\overline{x}_2$	$\ldots$	$\overline{x}_j$	$\ldots$	$\overline{x}_p$
var	s_1^2	s_2^2	$\ldots$	s_j^2	$\ldots$	s_p^2

Il problema dell'analisi dei cluster consiste nel determinare m sottoinsiemi, detti cluster, di individui in I, con m intero minore di n, tali che I_i appartenga soltanto ad un unico sottoinsieme. Gli individui che sono assegnati allo stesso cluster sono detti *simili* mentre gli individui che sono assegnati a differenti cluster sono detti *dissimili*.

È importante osservare che la *scelta delle variabili* (delle caratteristiche da osservare) è strettamente condizionata allo scopo dell'indagine e presuppone l'esistenza, seppure ad uno stato iniziale, di un modello logico. Essa è effettuata senza alcun ricorso a criteri matematici o statistici e riflette il giudizio del ricercatore sull'importanza delle proprietà utili a descrivere il fenomeno in funzione del quale deve essere svolta l'analisi dei cluster. Ad esempio, se si desidera effettuare l'analisi dei cluster sui *comuni di una regione* in funzione dell'*ambiente socio–economico*, si dovranno prendere in considerazione i parametri rilevati in ciascun comune che descrivono la situazione sanitaria, la situazione demografica, l'occupazione, ...; se, invece, si desidera effettuare l'analisi dei cluster sui comuni di una regione in funzione del *comportamento elettorale*, si dovranno prendere in considerazione i parametri rilevati che descrivono le percentuali di voti conseguite dai diversi partiti politici in ciascun comune.

Uno dei problemi che si presenta nell'analisi dei cluster riguarda la standardizzazione o meno delle variabili poiché attribuire un peso diverso a ciascuna caratteristica potrebbe condurre a risultati differenti circa la classificazione a seconda delle tecniche di clustering utilizzate. In molti metodi di clustering si raccomanda la *standardizzazione di ogni variabile* (caratteristica) usando la *media campionaria* e la *deviazione standard campionaria* entrambe derivate dall'insieme completo di individui della popolazione. Se, ad esempio, si desidera

effettuare l'analisi dei cluster sui *comuni di una regione in funzione dell'ambiente socio–economico* è preferibile utilizzare misure in forma standardizzata poiché potrebbe risultare difficile stabilire se parametri relativi all'occupazione debbano avere un peso maggiore o minore di parametri relativi alla situazione demografica; se, invece, si desidera effettuare l'analisi dei cluster sui *comuni di una regione in funzione del comportamento elettorale* è preferibile utilizzare pesi proporzionali alle percentuali medie dei suffragi ottenuti da ciascun partito politico, ossia delle misure non standardizzate.

Un ulteriore problema è che, relativamente al fenomeno osservato, il numero delle caratteristiche misurate per ogni individuo può essere grande e non sempre è necessario utilizzarle tutte. Infatti, includere variabili con un piccolo potere discriminante può appiattire le differenze tra gruppi, mentre includere variabili con un elevatissimo potere discriminante può rendere inutile l'inclusione di altre variabili strettamente collegate al fenomeno analizzato dal punto di vista logico. Poiché in molte tecniche di clustering il tempo di calcolo cresce drammaticamente con il crescere del numero delle variabili, in più casi è desiderabile, prima di utilizzare tale analisi, *ridurre il numero di variabili* a quelle più direttamente collegate al fenomeno in esame. Un metodo che permette di effettuare tale riduzione delle variabili originarie è *l'analisi delle componenti principali*. Le tecniche di clustering possono essere applicate alle prime q componenti principali ($q < p$) che possono essere considerate come nuove variabili di input. Ovviamente, l'analisi delle componenti principali permette di ridurre il numero delle variabili ma non risolve il problema della standardizzazione e della correlazione delle variabili. Inoltre, le variabili di input determinano la classificazione ed è possibile determinare una diversa classificazione utilizzando le prime q componenti principali invece che tutte le caratteristiche iniziali. Nel caso di dati ben strutturati (gruppi ben differenziati) le differenze tra le due classificazioni sono piccole, ma nel caso di gruppi non ben differenziati, possono essere riscontrate delle grandi differenze.

Per risolvere il problema di clustering è chiaramente desiderabile definire i termini *somiglianza* o *differenza* in modo quantitativo, ossia occorre precisare cosa significa la somiglianza di due individui I_i e I_j assegnati allo stesso cluster e la differenza di due individui assegnati a differenti cluster. La somiglianza può essere definita mediante un *coefficiente di similarità* $s_{ij} = s(X_i, X_j)$ oppure mediante una misura di distanza $d_{ij} = d(X_i, X_j)$ tra due individui I_i e I_j ($i \neq j$). Mentre i coefficienti di similarità assumono valori compresi tra 0 e 1, le misure di distanza possono assumere qualsiasi valore reale maggiore o uguale a zero. Un criterio per risolvere il problema di clustering potrebbe essere quello di assegnare due individui I_i e I_j ($i \neq j$) allo stesso cluster se il coefficiente di similarità tra i punti X_i e X_j è prossimo ad 1 oppure se la distanza tra i punti X_i e X_j è sufficientemente piccola e a differenti cluster se il coefficiente di similarità tra i punti è prossimo ad 0 oppure se la distanza tra i punti è sufficientemente grande.

Nel prossimo paragrafo definiamo la funzione distanza tra i punti X_i eX_j appartenenti ad uno spazio Euclideo E_p a p dimensioni e il coefficiente di similarità.

6.3 Distanza e similarità

Le *misure metriche di somiglianza* sono soprattutto basate sulle *funzioni distanza* tra i vettori delle caratteristiche. Occorre quindi definire tale funzione.

Definizione 6.1 *Una funzione a valori reali $d(X_i, X_j)$ è detta funzione distanza se e soltanto se essa soddisfa le seguenti condizioni:*

(i) $d(X_i, X_j) = 0$ *se e solo se* $X_i = X_j$, *con* X_i *e* X_j *in* E_p;

(ii) $d(X_i, X_j) \geq 0$ *per ogni* X_i *e* X_j *in* E_p;

(iii) $d(X_i, X_j) = d(X_j, X_i)$ *per ogni* X_i *e* X_j *in* E_p;

(iv) $d(X_i, X_j) \leq d(X_i, X_k) + d(X_k, X_j)$ *per ogni* X_i, X_j *e* X_k *in* E_p.

La proprietà *(i)* implica che X_i è a distanza zero da se stesso e che ogni due punti a distanza nulla debbono essere identici. La proprietà *(ii)* afferma che la funzione distanza è non negativa. La proprietà *(iii)* impone la simmetria richiedendo che la distanza tra X_i e X_j deve essere la stessa della distanza tra X_j e X_i. La proprietà *(iv)*, nota come *disuguaglianza triangolare*, richiede che la distanza tra X_i e X_j debba essere sempre minore o uguale della somma delle distanze di ognuno dei due vettori considerati da qualunque altro terzo vettore X_k.

In generale, le distanze tra tutte le possibili coppie di unità sono inserite in una matrice simmetrica D di cardinalità $n \times n$, ossia

$$D = \begin{pmatrix} 0 & d_{12} & \dots & d_{1n} \\ d_{21} & 0 & \dots & d_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ d_{n1} & d_{n2} & \dots & 0 \end{pmatrix}, \tag{6.3}$$

dove $d_{ij} = d(X_i, X_j) \quad (i, j = 1, 2, \dots, n)$. Dalla Definizione 6.1 segue che i termini sulla diagonale principale sono tutti uguali a zero mentre i termini simmetrici sono uguali a due a due. Il numero di distanze che è necessario conoscere affinché sia definita la posizione di ciascuna delle n variabili rispetto alle rimanenti $n-1$ è, dunque, $n(n-1)/2$. È sufficiente, pertanto, considerare la matrice triangolare al di sopra o al di sotto della diagonale principale di D.

Non esiste una sola funzione distanza ma esiste un'intera famiglia di funzioni che rispettano almeno le quattro proprietà della Definizione 6.1. Infatti, si possono facilmente dimostrare le seguenti proprietà:

(a) se d e d' sono due metriche anche $d + d'$ è una metrica;

(b) se d è una metrica e c un numero reale positivo allora anche $c\,d$ è una metrica;

(c) se d è una metrica e c un numero reale positivo allora anche $d' = d/(c + d)$ è una metrica.

Il *prodotto di due metriche* (in particolare il *quadrato di una metrica*) non necessariamente soddisfa la disuguaglianza triangolare e quindi può non essere una misura di distanza.

In R la funzione:

```
> dist(X, method = "euclidean", diag = FALSE, upper = FALSE)
```

ritorna la matrice delle distanze D calcolata utilizzando le misure di distanza tra le righe della matrice X dei dati, dove:

- X rappresenta una matrice numerica o un data frame;
- method seleziona la misura di distanza da utilizzare (di default è euclidean);
- diag è posta uguale aTRUE se si desidera che la matrice delle distanze contenga anche i valori nulli sulla diagonale (di default è FALSE);
- upper è posta uguale a TRUE se si desidera che la matrice delle distanze contenga anche i valori al di sopra della diagonale principale (di default è FALSE).

Le opzioni disponibili per il calcolo della matrice delle distanze sono:

(1) metrica euclidea (euclidean);

(2) metrica del valore assoluto o metrica di Manhattan (manhattan);

(3) metrica del massimo o metrica di Chebycev (maximum);

(4) metrica di Minkowski (minkowski);

(5) distanza di Canberra (canberra);

(6) distanza di Jaccard (binary);

Nella funzione dist() potrebbe essere presente anche un ulteriore parametro finale r che indica la potenza della metrica di Minkowski (di default è r = 2, che corrisponde alla metrica euclidea).

Una volta calcolata una matrice d delle distanze, per visualizzare la matrice completa si può utilizzare la funzione as.matrix(d).

6.3.1 Metrica Euclidea

La più familiare misura di distanza è la *metrica Euclidea*, così definita:

$$d_2(X_i, X_j) = \left[\sum_{k=1}^{p} (x_{ik} - x_{jk})^2 \right]^{1/2}, \tag{6.4}$$

dove x_{ik} è il valore della k–esima caratteristica dell'individuo I_i.

Se si considerano due caratteristiche, ossia $p = 2$, l'espressione (6.4) corrisponde alla radice quadrata della somma dei quadrati costruiti sui cateti di un

triangolo rettangolo e per il teorema di Pitagora tale radice fornisce la misura dell'ipotenusa del triangolo stesso.

La distanza Euclidea usata su tutti i dati è fortemente *influenzata dall'unità di misura* in base alla quale è valutata ciascuna delle p caratteristiche. Consideriamo, ad esempio, tre individui I_1, I_2 e I_3 e due caratteristiche associate ad ogni individuo, ossia il loro peso e la loro altezza. Supponiamo che la matrice delle misure sia

$$X = \begin{array}{c} \\ I_1 \\ I_2 \\ I_3 \end{array} \begin{array}{c} \begin{array}{cc} C_1 & C_2 \end{array} \\ \begin{pmatrix} 60 & 160 \\ 65 & 165 \\ 63 & 170 \end{pmatrix} \end{array}, \tag{6.5}$$

dove il *peso è misurato in chilogramm*i e l'*altezza in centimetri*. Le seguenti linee di codice

```
> X<-data.frame(peso=c(60,65,63),altezza=c(160,165,170))
> row.names(X)<-c("I1","I2","I3")
> X #visualizza il data frame X
   peso altezza
I1   60     160
I2   65     165
I3   63     170
>
> dist(X,method="euclidean",diag=TRUE,upper=TRUE)
          I1        I2        I3
I1  0.000000  7.071068 10.440307
I2  7.071068  0.000000  5.385165
I3 10.440307  5.385165  0.000000
```

definiscono il data frame X dei dati e calcolano la matrice delle distanze. Si nota che $d_2(X_1, X_2) = 7.071068$, $d_2(X_1, X_3) = 10.440307$, $d_2(X_2, X_3) = 5.385165$.

Se, invece, gli stessi individui vengono misurati per quanto riguarda l'*altezza in metri*, il data frame delle misure diventa

$$Y = \begin{array}{c} \\ I_1 \\ I_2 \\ I_3 \end{array} \begin{array}{c} \begin{array}{cc} C_1 & C_2 \end{array} \\ \begin{pmatrix} 60 & 1.60 \\ 65 & 1.65 \\ 63 & 1.70 \end{pmatrix} \end{array} \tag{6.6}$$

e con seguenti linee di codice

```
> Y<-data.frame(peso=c(60,65,63),altezza=c(1.60,1.65,1.70))
> row.names(Y)<-c("I1","I2","I3")
> Y #visualizza il data frame Y
   peso altezza
I1   60    1.60
I2   65    1.65
I3   63    1.70
>
> dist(Y,method="euclidean",diag=TRUE,upper=TRUE)
         I1       I2       I3
I1 0.000000 5.000250 3.001666
I2 5.000250 0.000000 2.000625
I3 3.001666 2.000625 0.000000
```

si ha $d_2(X_1, X_2) = 5.000250$, $d_2(X_1, X_3) = 3.001666$, $d_2(X_2, X_3) = 2.000625$.

Si nota che mentre nel primo caso l'individuo I_1 era più vicino all'individuo I_2, nel secondo caso lo stesso individuo I_1 è più vicino all'individuo I_3. Inoltre, non esiste una trasformazione che permetta di passare dai primi valori della distanza ai secondi valori, il che significa che la funzione distanza è legata alle unità di misura in maniera non invariante. Il metodo più utilizzato per ovviare a questo inconveniente è quello di scalare e standardizzare inizialmente le misure in maniera tale da realizzare la possibilità di un confronto tra le misure, ossia considerare delle nuove variabili

$$x^*_{ij} = \frac{x_{ij} - \overline{x}_j}{s_j} \qquad (i = 1, 2, \ldots, n; j = 1, 2, \ldots, p) \tag{6.7}$$

dove $\overline{x}_j$ e s_j sono rispettivamente la media campionaria e la deviazione standard campionaria della j–esima caratteristica. In Tabella 6.2 sono riportati i dati scalati e standardizzati osservati per n individui.

Tabella 6.2: Dati scalati e standardizzati osservati per n individui.

	x^*_{11}	x^*_{12}	$\ldots$	x^*_{1j}	$\ldots$	x^*_{1p}
	x^*_{21}	x^*_{22}	$\ldots$	x^*_{2j}	$\ldots$	x^*_{2p}
	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$
	x^*_{i1}	x^*_{i2}	$\ldots$	x^*_{ij}	$\ldots$	x^*_{ip}
	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$
	x^*_{n1}	x^*_{n2}	$\ldots$	x^*_{nj}	$\ldots$	x^*_{np}
mean	0	0	$\ldots$	0	$\ldots$	0
var	1	1	$\ldots$	1	$\ldots$	1

Mediante lo scalamento e la standardizzazione si ottengono dei nuovi dati le cui medie campionarie sono nulle e le varianze campionarie unitarie. Infatti, ricordando le (6.2) dalla (6.7) per $j = 0, 1, \ldots, p$ si ha:

$$\overline{x}^*_j = \frac{1}{n}\sum_{i=1}^{n} x^*_{ij} = \frac{1}{n}\sum_{i=1}^{n} \frac{x_{ij} - \overline{x}_j}{s_j} = \frac{1}{n\, s_j}\left[\sum_{i=1}^{n} x_{ij} - n\overline{x}_j\right] = 0,$$

$$(s^*_j)^2 = \frac{1}{n-1}\sum_{i=1}^{n}\left(x^*_{ij} - \overline{x}^*_j\right)^2 = \frac{1}{n-1}\sum_{i=1}^{n}\left(\frac{x_{ij} - \overline{x}_j}{s_j}\right)^2 = 1.$$

In R per scalare e standardizzare le variabili si utilizza la funzione

> scale(X, center = TRUE, scale = TRUE)

dove

- X rappresenta una matrice numerica o un data frame;

- center è posta uguale a TRUE se dagli elementi di ogni colonna della matrice X si sottrae il valore medio della corrispondente colonna (di default è TRUE);

- scale è posta uguale a TRUE se si dividono gli elementi centrati di ogni colonna della matrice X per la deviazione standard della corrispondente colonna (di default è TRUE).

Ad esempio, riferendosi alla matrice X in (6.5) e applicando la funzione scale(X) si ottiene:

```
> scale(X)
         peso altezza
I1 -1.0596259      -1
I2  0.9271726       0
I3  0.1324532       1
attr(,"scaled:center")
     peso   altezza
 62.66667 165.00000
attr(,"scaled:scale")
    peso  altezza
2.516611 5.000000
```

dove i valori indicati nei due attributi corrispondono alle medie campionarie e alle deviazioni standard campionarie delle due colonne del data frame originario dei dati. Infatti, utilizzando le funzione apply(X, 2, mean), apply(X, 2, var) e apply(X, 2, sd) è possibile calcolare la media campionaria, la varianza campionaria e la deviazione standard campionaria delle colonne del data frame originario X. Applicando tali funzioni direttamente al data frame Xi si ha:

```
> apply(X,2,mean)
     peso   altezza
 62.66667 165.00000
> apply(X,2,sd)
    peso  altezza
2.516611 5.000000
```

Considerando ora il data frame Y in (6.6) e applicando la funzione scale(Y) si ha:

```
> scale(Y)
         peso altezza
I1 -1.0596259      -1
I2  0.9271726       0
I3  0.1324532       1
attr(,"scaled:center")
    peso  altezza
62.66667  1.65000
attr(,"scaled:scale")
    peso  altezza
2.516611 0.050000
```

dove i valori indicati nei due attributi corrispondono alle medie campionarie e alle deviazioni campionarie delle due colonne del data frame originario dei dati, ottenibili anche nel seguente modo:

```
> apply(Y,2,mean)
    peso  altezza
62.66667  1.65000
```

```
> apply(Y,2,sd)
    peso  altezza
2.516611 0.050000
```

Si nota che sia partendo dal data frame X sia a partire dal data frame Y, dopo lo scalamento e la standardizzazione si è ottenuto lo stesso data frame. Calcoliamo ora la funzione distanza per la matrice X (o equivalentemente per la matrice Y) dopo aver effettuato lo scalamento:

```
> Z<-scale(X)
> dist(Z,method="euclidean",diag=TRUE,upper=TRUE)
         I1       I2       I3
I1 0.000000 2.224268 2.328315
I2 2.224268 0.000000 1.277333
I3 2.328315 1.277333 0.000000
```

che mostra che l'individuo I_1 è più vicino all'individuo I_2 (così come accadeva per la matrice originaria X in cui l'altezza era misurata in centimetri).

6.3.2 Altre metriche

La metrica Euclidea è molto usata nelle tecniche di clustering, ma esistono molte altre possibili metriche. Alcune metriche spesso utilizzate sono la *metrica del valore assoluto* (*metrica di Manhattan*), detta *metrica* L_1, così definita

$$d_1(X_i, X_j) = \sum_{k=1}^{p} |x_{ik} - x_{jk}| \tag{6.8}$$

e la *metrica del massimo* (*metrica di Chebycev*), così definita:

$$d_\infty(X_i, X_j) = \max_{k=1,2\ldots,p} |x_{ik} - x_{jk}|. \tag{6.9}$$

Se si considerano due caratteristiche, ossia $p = 2$, la metrica del valore assoluto corrisponde alla somma delle misure dei due cateti di un triangolo rettangolo. Il suo nome "Manhattan" deriva proprio dal fatto che essa corrisponde alla lunghezza che si deve percorrere qualora sia consentito di muoversi solo nelle direzioni parallele agli assi, come avviene in una città con una griglia regolare di strade che si intersecano ad angolo retto.

Utilizzando R applichiamo la *metrica del valore assoluto* (*metrica di Manhattan*) e la *metrica del massimo* al data frame scalato Z ottenendo:

```
> dist(Z,method="manhattan",diag=TRUE,upper=TRUE)
         I1       I2       I3
I1 0.000000 2.986799 3.192079
I2 2.986799 0.000000 1.794719
I3 3.192079 1.794719 0.000000
>
> dist(Z,method="maximum",diag=TRUE,upper=TRUE)
         I1       I2       I3
I1 0.000000 1.986799 2.000000
I2 1.986799 0.000000 1.000000
I3 2.000000 1.000000 0.000000
```

Utilizzando entrambe tali metriche si ha che l'individuo I_1 è più vicino all'individuo I_2 come accade per la metrica euclidea.

Entrambe le metriche del valore assoluto e del massimo sono computazionalmente semplici da calcolare con l'unica differenza che *la metrica di Chebycev coinvolge anche una procedura di ordinamento*.

Una misura di distanza che include come caso particolare la distanza Euclidea, la metrica del valore assoluto e la metrica di Chebycev risulta essere la *metrica di Minkowski*, detta anche *metrica* L_r, così definita

$$d_r(X_i, X_j) = \left[\sum_{k=1}^{p} |x_{ik} - x_{jk}|^r\right]^{1/r}, \tag{6.10}$$

dove $r \geq 1$. Se $r = 2$ si ottiene la *metrica Euclidea*, se $r = 1$ si ottiene la *metrica del valore assoluto* ed infine se $r = \infty$ si ottiene la *metrica di Chebycev*. Per ogni coppia di vettori X_i e X_j in E_p e per ogni due interi positivi r e k tali che $r \geq k$ vale la disuguaglianza $d_r(X_i, X_j) \leq d_k(X_i, X_j)$, che implica che

$$d_\infty(X_i, X_j) \leq d_2(X_i, X_j) \leq d_1(X_i, X_j).$$

La distanza di Minkowski (o city-block) trova applicazioni nell'urbanistica ed in particolare nella costruzione di strade in una città in cui sono già presenti degli edifici.

In R, applicando la metrica di Minkowski con $r = 4$ al data frame scalato Z, si ottiene:

```
> dist(Z,method="minkowski",4,diag=TRUE,upper=TRUE)
          I1        I2        I3
I1 0.000000 2.017936 2.060322
I2 2.017936 0.000000 1.087542
I3 2.060322 1.087542 0.000000
```

Anche in questo caso l'individuo I_1 è più vicino all'individuo I_2 come accadeva con le precedenti metriche.

Un'altra possibile metrica è la *metrica di Canberra*, proposta da Lance & Williams nel 1966, così definita:

$$d_c(X_i, X_j) = \sum_{k=1}^{p} \frac{|x_{ik} - x_{jk}|}{|x_{ik}| + |x_{jk}|}, \tag{6.11}$$

in cui sono omessi i valori aventi zero al numeratore o al denominatore. La metrica di Canberra è definita per *variabili non negative* ed ha la caratteristica di essere sensibile alle differenze relative piuttosto che a quelle assolute. Questa distanza assegna alla differenza fra i valori relativi ai vettori X_i e X_j un peso inversamente proporzionale alla somma dei valori stessi. Uno dei problemi di questa distanza è che, se uno dei due valori x_{ik} o x_{jk} è uguale a zero, allora il contributo $|x_{ik} - x_{jk}|/|x_{ik} + x_{jk}|$ alla distanza totale è uguale a 1, ossia il massimo possibile. Se si utilizza tale metrica non è necessario scalare la matrice dei dati, poiché i contributi alla somma sono adimensionali. Inoltre, la metrica

di Canberra è poco sensibile all'asimmetria delle distribuzioni delle variabili (caratteristiche) e alla presenza di eventuali valori anomali (outlier).

In R, applicando la metrica di Canberra al data frame X, in cui il peso è misurato in chilogrammi e l'altezza in centimetri, si ottiene:

```
> dist(X,method="canberra",diag=TRUE,upper=TRUE)
           I1         I2         I3
I1 0.00000000 0.05538462 0.05469327
I2 0.05538462 0.00000000 0.03055037
I3 0.05469327 0.03055037 0.00000000
```

mentre, applicando la metrica di Canberra al data frame Y, in cui il peso è misurato in chilogrammi e l'altezza in metri, si ottiene:

```
> dist(Y,method="canberra",diag=TRUE,upper=TRUE)
           I1         I2         I3
I1 0.00000000 0.05538462 0.05469327
I2 0.05538462 0.00000000 0.03055037
I3 0.05469327 0.03055037 0.00000000
```

A partire dal data frame X e dal data frame Y si è quindi ottenuta la stessa matrice delle distanze. Si nota che l'individuo I_1 è più vicino all'individuo I_3.

Un'altra funzione distanza è la *distanza di Jaccard* così definita:

$$d(X_i, X_j) = 1 - \frac{\sum_{k=1}^{p} \min(x_{ik}, x_{jk})}{\sum_{k=1}^{p} \max(x_{ik}, x_{jk})} \tag{6.12}$$

In R è disponibile la distanza di Jaccard soltanto per *vettori binari*. Infatti, se si denota n_{00} il numero di caratteristiche che valgono 0 in entrambi i vettori considerati, n_{11} il numero di caratteristiche che valgono 1 in entrambi i vettori considerati, n_{01} il numero di caratteristiche che valgono 0 in X_i e 1 in X_j e con n_{10} il numero di caratteristiche che valgono 1 in X_i e 0 in X_j, la (6.12) diventa:

$$d(X_i, X_j) = 1 - \frac{n_{11}}{n_{11} + n_{01} + n_{10}} = \frac{n_{01} + n_{10}}{n_{11} + n_{01} + n_{10}}. \tag{6.13}$$

che fornisce la proporzione di bits nei vettori X_i e X_j in cui solo uno vale 1 rispetto ai bits in cui almeno uno vale 1.

Consideriamo, ad esempio, la matrice dei dati

$$U = \begin{array}{c} \\ I_1 \\ I_2 \\ I_3 \end{array} \begin{array}{c} \begin{array}{cccccccccc} C_1 & C_2 & C_3 & C_4 & C_5 & C_6 & C_7 & C_8 & C_9 & C_{10} \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \end{pmatrix} \end{array}$$

e inseriamo i valori n_{ij} relativi ai vettori (X_1, X_2), (X_1, X_3) e (X_2, X_3) nelle rispettive matrici $n(X_1, X_2)$, $n(X_1, X_3)$ e $n(X_2, X_3)$ ottenendo:

$$n(X_1, X_2) = \begin{array}{c} \\ 0 \\ 1 \end{array} \begin{array}{c} \begin{array}{cc} 0 & 1 \end{array} \\ \begin{pmatrix} 5 & 1 \\ 2 & 2 \end{pmatrix} \end{array} \quad n(X_1, X_3) = \begin{array}{c} \\ 0 \\ 1 \end{array} \begin{array}{c} \begin{array}{cc} 0 & 1 \end{array} \\ \begin{pmatrix} 1 & 5 \\ 2 & 2 \end{pmatrix} \end{array} \quad n(X_2, X_3) = \begin{array}{c} \\ 0 \\ 1 \end{array} \begin{array}{c} \begin{array}{cc} 0 & 1 \end{array} \\ \begin{pmatrix} 0 & 7 \\ 3 & 0 \end{pmatrix} \end{array}$$

Utilizzando poi la (6.13), si può ottenere la matrice delle distanze, ossia

$$D = \begin{pmatrix} 0 & 3/5 & 7/9 \\ 3/5 & 0 & 1 \\ 7/9 & 1 & 0 \end{pmatrix},$$

dove $3/5 = 0.6$ e $7/9 = 0.777777844$. Calcoliamo nuovamente la matrice delle distanze D di Jaccard relativa alla matrice U dei dati mediante R:

```
> U<-data.frame(c1=c(1,1,0),c2=c(0,0,1),c3=c(0,0,1),c4=c(1,0,1),
+ c5=c(0,1,0),c6=c(0,0,1),c7=c(1,0,1),c8=c(0,0,1),c9=c(0,0,1),
+ c10=c(1,1,0))
> row.names(U)<-c("I1","I2","I3")
> U # visualizza U
   c1 c2 c3 c4 c5 c6 c7 c8 c9 c10
I1  1  0  0  1  0  0  1  0  0   1
I2  1  0  0  0  1  0  0  0  0   1
I3  0  1  1  1  0  1  1  1  1   0
>
> dist(U,method="binary",diag=TRUE,upper=TRUE)
          I1        I2        I3
I1 0.0000000 0.6000000 0.7777778
I2 0.6000000 0.0000000 1.0000000
I3 0.7777778 1.0000000 0.0000000
```

Si nota che gli individui I_1 e I_2 hanno la minima distanza di Jaccard.

6.3.3 Misure di similarità

Nella maggior parte delle tecniche di clustering occorre inizialmente calcolare la matrice D delle distanze oppure una matrice S delle similarità. Una *misura di similarità* fornisce un valore numerico compreso tra 0 e 1 e permette di definire in modo quantitativo la somiglianza o differenza tra due individui I_i e I_j, intendendo ovviamente con 0 l'assoluta assenza e con 1 la massima presenza di somiglianza. Occorre quindi definire tale funzione.

Definizione 6.2 *Una funzione a valori reali $s_{ij} = s(X_i, X_j)$ è detta misura di similarità se e soltanto se essa soddisfa le seguenti condizioni:*

(i) $s(X_i, X_i) = 1$*;*

(ii) $0 \leq s(X_i, X_j) \leq 1$*;*

(iii) $s(X_i, X_j) = s(X_j, X_i)$ *per ogni* X_i *e* X_j*.*

La proprietà *(i)* implica che la misura di similarità è unitaria se i due punti sono identici. La proprietà *(ii)* afferma che la misura di similarità è compresa tra 0 e 1. La proprietà *(iii)* impone la simmetria richiedendo che la misura di similarità tra X_i e X_j deve essere la stessa della misura di similarità tra X_j e X_i. La quantità s_{ij} è chiamata semplicemente *coefficiente di similarità* e risulta essere l'elemento nella riga i-esima e colonna j-esima della matrice di similarità

S, così definita:

$$S = \begin{pmatrix} 1 & s_{12} & \dots & s_{1n} \\ s_{21} & 1 & \dots & s_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ s_{n1} & s_{n2} & \dots & 1 \end{pmatrix}. \tag{6.14}$$

Un esempio di misura di similarità per vettori binari è il complemento a 1 della distanza di Jaccard, detto *coefficiente di similarità di Jaccard*, ossia

$$s(X_i, X_j) = \frac{n_{11}}{n_{11} + n_{01} + n_{10}}.$$

Infatti si nota che le condizioni *(ii)* e *(iii)* sono soddisfatte; inoltre, se $X_i = X_j$ risulta che $n_{01} = n_{10} = 0$ e quindi anche la prima condizione è valida.

La più ovvia differenza tra una misura di similarità e una misura di distanza è che la prima misura assume valori tra 0 e 1, mentre la seconda misura assume valori non negativi. È importante comunque sottolineare che è sempre possibile trasformare una misura di distanza in una misura di similarità. Ad esempio, se d_{ij} è la distanza tra X_i e X_j, allora mediante la trasformazione

$$s_{ij} = \frac{1}{1 + d_{ij}} \qquad (i, j = 1, 2, \dots, n)$$

si ottiene una misura di similarità tra X_i e X_j. Invece, passare da una misura di similarità ad una misura di distanza non è sempre possibile poiché la funzione distanza deve soddisfare anche la disuguaglianza triangolare.

6.4 Misura di non omogeneità totale

Riconsideriamo un insieme $I = \{I_1, I_2, \dots, I_n\}$ di n individui e assumiamo che esista un *insieme di caratteristiche* $C = \{C_1, C_2, \dots, C_p\}$ che sono osservabili e sono possedute da ogni individuo in I. Sia

$$X = \begin{pmatrix} x_{11} & x_{12} & \dots & x_{1p} \\ x_{21} & x_{22} & \dots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \dots & x_{np} \end{pmatrix} = \begin{pmatrix} X_1 \\ X_2 \\ \vdots \\ X_n \end{pmatrix} \tag{6.15}$$

la matrice delle misure (supponiamo di effettuare lo scalamento se le colonne hanno unità di misura differenti) e sia

$$D = \begin{pmatrix} 0 & d_{12} & \dots & d_{1n} \\ d_{21} & 0 & \dots & d_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ d_{n1} & d_{n2} & \dots & 0 \end{pmatrix} \tag{6.16}$$

la matrice delle distanze di cardinalità $n \times n$, dove $d_{ij} = d(X_i, X_j) \quad (i, j = 1, 2, \dots, n)$.

Alla matrice X si può associare una matrice W_X di cardinalità $p \times p$, detta *matrice delle varianze e covarianze* così definita:

$$W_X = \begin{pmatrix} w_{11} & w_{12} & \dots & w_{1p} \\ w_{21} & w_{22} & \dots & w_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ w_{p1} & w_{p2} & \dots & w_{pp} \end{pmatrix}, \tag{6.17}$$

dove il generico elemento $w_{r\ell}$ è dato da:

$$w_{r\ell} = \frac{1}{n-1} \sum_{i=1}^{n} (x_{ir} - \overline{x}_r)\,(x_{i\ell} - \overline{x}_\ell) \qquad (r, \ell = 1, 2, \dots, p) \tag{6.18}$$

Si nota che se $r = \ell$ l'elemento $w_{r\ell}$ è la *varianza campionaria* relativa alla caratteristica r-esima effettuata su tutti gli n individui, mentre se $r \neq \ell$ l'elemento $w_{r\ell}$ è la *covarianza campionaria* tra la caratteristica r-esima e la caratteristica ℓ-esima effettuata su tutti gli n individui.

In R utilizzando le funzione apply(X, 2, mean), apply(X, 2, var) e apply(X, 2, sd) è possibile calcolare la *media campionaria*, la *varianza campionaria* e la *deviazione standard campionaria* delle colonne di una matrice X. Inoltre applicando la funzione cov(X) è possibile ottenere la matrice W_X delle *varianze e covarianze campionarie tra le varie caratteristiche*.

Esempio 6.1 Consideriamo la seguente matrice dei dati che si riferisce a due caratteristiche C_1 e C_2 osservate per cinque differenti individui I_1, I_2, I_3, I_4, I_5, espresse nelle stesse unità di misura:

$$X = \begin{array}{c} \\ I_1 \\ I_2 \\ I_3 \\ I_4 \\ I_5 \end{array} \begin{array}{c} \begin{array}{cc} C_1 & C_2 \end{array} \\ \begin{pmatrix} 1 & 1 \\ 1 & 2 \\ 6 & 3 \\ 8 & 2 \\ 8 & 0 \end{pmatrix} \end{array}$$

Le seguenti linee di codice

```
> X<-data.frame(c1=c(1,1,6,8,8),c2=c(1,2,3,2,0))
> row.names(X)<-c("I1","I2","I3","I4","I5")
> X # visualizza il data frame X
   c1 c2
I1  1  1
I2  1  2
I3  6  3
I4  8  2
I5  8  0
>
> apply(X,2,mean)
 c1  c2
4.8 1.6
>
```

```
> apply(X,2,var)
  c1   c2
12.7  1.3
>
> WI<-cov(X)
> WI # visualizza la matrice di covarianza
      c1    c2
c1 12.70 -0.35
c2 -0.35  1.30
```

Si nota che $s_1^2 = 12.70$, $s_2^2 = 1.30$ e $\text{Cov}(C_1, C_2) = -0.35$, che mostra che le due caratteristiche C_1 e C_2 sono *correlate negativamente*. Inoltre, possiamo rappresentare i cinque punti relativi agli individui I_1, I_2, I_3, I_4, I_5 in uno scatterplot. Le seguenti linee di codice

```
> plot(X$c1,X$c2,col="red",xlab="C1",
+ ylab="C2",ylim=c(0,3.5))
> text(X$c1,X$c2+0.1,c("I1","I2","I3","I4","I5"))
>
> abline(lm(X$c2~X$c1),lty=2,col="blue")
```

producono lo scatterplot di Figura 6.1 in cui è riportata anche la retta di regressione. ♢

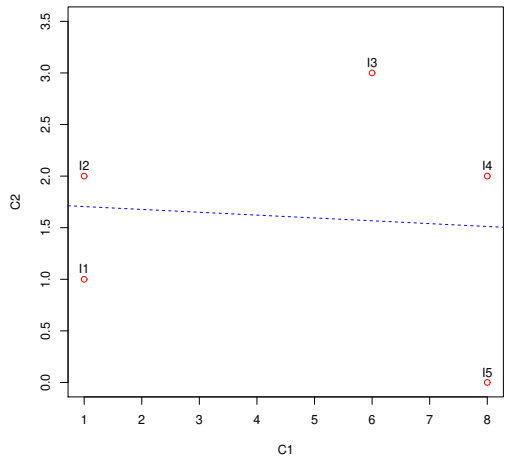

Figura 6.1: Scatterplot associato ai cinque individui.

Vogliamo ora definire la *matrice statistica di non omogeneità*.

Definizione 6.3 *La matrice statistica di non omogeneità (statistical scatter matrix) per l'insieme I di individui, di cardinalità $p \times p$, è così definita:*

$$H_I = (n-1)W_I = \begin{pmatrix} h_{11} & h_{12} & \dots & h_{1p} \\ h_{21} & h_{22} & \dots & h_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ h_{p1} & h_{p2} & \dots & h_{pp} \end{pmatrix}, \tag{6.19}$$

dove l'elemento generico $h_{r\ell}$ risulta essere

$$h_{r\ell} = \sum_{i=1}^{n} (x_{ir} - \overline{x}_r)\,(x_{i\ell} - \overline{x}_\ell) = (n-1)\, w_{r\ell} \qquad (r, \ell = 1, 2 \dots, p). \tag{6.20}$$

Quando $r = \ell$, si nota che h_{rr} corrisponde a $n-1$ volte la varianza campionaria della caratteristica r-esima effettuata su tutti gli n individui, ossia

$$h_{rr} = (n-1)\,\mathrm{Var}(C_r) = (n-1)\, s_r^2 \qquad (r = 1, 2 \dots, p). \tag{6.21}$$

Definizione 6.4 *Si definisce misura di non omogeneità statistica (statistical scatter) dell'insieme I di individui la traccia della matrice H_I:*

$$\mathrm{tr}H_I = \sum_{r=1}^{p} h_{rr} = (n-1) \sum_{r=1}^{p} s_r^2. \tag{6.22}$$

È possibile applicare tale definizione quando $n > 1$; quando $n = 1$, si suppone che la misura di non omogeneità statistica sia nulla.

La traccia di una matrice di non omogeneità di un insieme di individui fornisce una misura della dispersione dei dati intorno al valore medio dell'insieme dal quale è stata ricavata. È intuitivo pensare che più un insieme di dati è addensato più piccola è la traccia della matrice di non omogeneità.

La misura di non omogeneità statistica $\mathrm{tr}H_I$ è anche esprimibile in termine della somma dei quadrati delle distanze euclidee tra ogni vettore $X_1, X_2, \dots, X_n$ e il vettore $\overline{X}$ delle medie campionarie, come mostrato di seguito.

Proposizione 6.1 *La misura di non omogeneità statistica (6.22) dell'insieme I di individui si può anche scrivere:*

$$\mathrm{tr}H_I = \sum_{i=1}^{n} d_2^2(X_i, \overline{X}), \tag{6.23}$$

dove d_2 indica la distanza euclidea e dove $\overline{X} = (\overline{x}_1, \overline{x}_2, \dots, \overline{x}_p)$ è un vettore di cardinalità p il cui generico elemento $\overline{x}_j$ rappresenta la media campionaria relativa alla caratteristica j-esima effettuata sugli n individui.

Dimostrazione Poiché la traccia è la somma degli elementi sulla diagonale della matrice H_I, ricordando la (6.2) si ha:

$$\mathrm{tr}H_I = \sum_{k=1}^{p} \sum_{i=1}^{n} \big(x_{ik} - \overline{x}_k\big)^2 = \sum_{i=1}^{n} \Big[\sum_{k=1}^{p} \big(x_{ik} - \overline{x}_k\big)^2 \Big] = \sum_{i=1}^{n} d_2^2(X_i, \overline{X}).$$

ossia la (6.23). □

A partire dalla (6.23) si può dimostrare che risulta:

$$\mathrm{tr}H_I = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=i}^{n} d_2^2(X_i, X_j), \tag{6.24}$$

ossia la traccia della matrice di non omogeneità statistica corrisponde al rapporto tra la somma dei quadrati degli elementi al di sotto della diagonale principale della matrice delle distanze euclidee e il numero n di individui. Come si evince dalla (6.23) e (6.24) la distanza euclidea gioca un ruolo rilevante nel calcolo della misura di non omogeneità statistica. Inoltre, tale misura *dipende sia dall'omogeneità interna sia dalla numerosità del gruppo.*

Esempio 6.2 Riconsideriamo la matrice dei dati dell'Esempio 6.1. Vogliamo calcolare la matrice statistica di non omogeneità e la traccia di tale matrice dei dati con tre differenti metodi utilizzando R.

Il primo metodo calcola la misura di non omogeneità statistica dell'insieme I di individui utilizzando la definizione (6.22):

```
> # PRIMO METODO
> n<-nrow(X)
>  if(n>1)
> trHI<-(n-1)*sum(apply(X,2,var))
> else
>  trHI<-0
>
> trHI # visualizza la misura di non omogeneita' statistica
[1] 56
```

Il secondo metodo determina la matrice di non omogeneità statistica dell'insieme I di individui e somma gli elementi sulla diagonale.

```
> # SECONDO METODO (n>1)
> n<-nrow(X) # numero di righe del data frame
> WI<-cov(X)
> HI<-(n-1)*WI
> HI # visualizza la matrice di non omogeneita' statistica
     c1   c2
c1 50.8 -1.4
c2 -1.4  5.2
>
> trHI<-sum(diag(HI))
> trHI # visualizza la misura di non omogeneita' statistica
[1] 56
```

La matrice di non omogeneità statistica dell'insieme I di individui è

$$H_I = \begin{pmatrix} 50.8 & -1.4 \\ -1.4 & 5.2 \end{pmatrix},$$

e la traccia (misura di non omogeneità statistica) di questa matrice vale 56.

Il terzo metodo calcola la misura di non omogeneità statistica dell'insieme I di individui tramite la (6.24) utilizzando i quadrati delle distanze euclidee:

```
> # TERZO METODO (n>1)
> d<-dist(X,method="euclidean",diag=FALSE,upper=FALSE)
> d # visualizza la matrice delle distanze
         I1       I2       I3       I4
I2 1.000000
I3 5.385165 5.099020
I4 7.071068 7.000000 2.236068
I5 7.071068 7.280110 3.605551 2.000000
>
> tr<-sum(d^2)/n
> tr # visualizza la misura di non omogeneita' statistica
[1] 56
```

6.5 Misure di non omogeneità tra cluster

Nel Paragrafo 6.4 abbiamo considerato delle misure di non omogeneità relative all'insieme totale di individui della popolazione. Accanto a tali misure occorre anche definire delle misure di non omogeneità all'interno dei cluster e delle misure di non omogeneità (o disparità) tra cluster distinti. Al termine del procedimento di classificazione, gli individui appartenenti allo stesso cluster dovrebbero essere il più possibile omogenei tra di loro e il più possibile differenti da quelli appartenenti agli altri cluster individuati.

Misure di non omogeneità per due cluster

Siano $I = \{I_1, I_2, \ldots, I_{n_1}\}$ e $J = \{J_1, J_2, \ldots, J_{n_2}\}$ due cluster distinti di individui di una popolazione. Inoltre, sia $C = \{C_1, C_2, \ldots, C_p\}$ un insieme di caratteristiche che generano due insiemi di misure $\mathbf{X} = \{X_1, X_2, \ldots, X_{n_1}\}$ e $\mathbf{Y} = \{Y_1, Y_2, \ldots, Y_{n_2}\}$ che descrivono rispettivamente gli insiemi disgiunti I e J di individui. A tali insiemi sono associate due matrici di misure X e Y rispettivamente di cardinalità $n_1 \times p$ e $n_2 \times p$, ossia

$$X = \begin{pmatrix} x_{11} & x_{12} & \ldots & x_{1p} \\ x_{21} & x_{22} & \ldots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n_1 1} & x_{n_1 2} & \ldots & x_{n_1 p} \end{pmatrix} = \begin{pmatrix} X_1 \\ X_2 \\ \vdots \\ X_{n_1} \end{pmatrix}, \tag{6.25}$$

$$Y = \begin{pmatrix} y_{11} & y_{12} & \ldots & y_{1p} \\ y_{21} & y_{22} & \ldots & y_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ y_{n_2 1} & y_{n_2 2} & \ldots & y_{n_2 p} \end{pmatrix} = \begin{pmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_{n_2} \end{pmatrix}, \tag{6.26}$$

La media campionaria relativa alla caratteristica j-esima effettuata su tutti gli n_1 individui del primo cluster è:

$$\overline{x}_j = \frac{1}{n_1} \sum_{i=1}^{n_1} x_{ij} \qquad (i = 1, 2, \ldots, p), \tag{6.27}$$

mentre la media campionaria elative alla caratteristica j-esima effettuata su tutti gli n_2 individui del secondo cluster è:

$$\overline{y}_j = \frac{1}{n_2} \sum_{i=1}^{n_2} y_{ij} \qquad (i = 1, 2, \ldots, p). \tag{6.28}$$

Seguendo la Definizione 6.3 è possibile definire le matrici statistiche di non omogeneità (statistical scatter matrix) per gli insiemi I e J di individui, entrambe di cardinalità $p \times p$, così definite:

$$H_I = \begin{pmatrix} h_{11}^{(1)} & h_{12}^{(1)} & \ldots & h_{1p}^{(1)} \\ h_{21}^{(1)} & h_{22}^{(1)} & \ldots & h_{2p}^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ h_{p1}^{(1)} & h_{p2}^{(1)} & \ldots & h_{pp}^{(1)} \end{pmatrix}, \qquad H_J = \begin{pmatrix} h_{11}^{(2)} & h_{12}^{(2)} & \ldots & h_{1p}^{(2)} \\ h_{21}^{(2)} & h_{22}^{(2)} & \ldots & h_{2p}^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ h_{p1}^{(2)} & h_{k2}^{(2)} & \ldots & h_{pp}^{(2)} \end{pmatrix} \tag{6.29}$$

dove l'elemento generico $h_{r\ell}$ risulta essere

$$\begin{aligned} h_{r\ell}^{(1)} &= \sum_{i=1}^{n_1} (x_{ir} - \overline{x}_r)\,(x_{i\ell} - \overline{x}_\ell) \qquad (r, \ell = 1, 2 \ldots, p), \\ h_{r\ell}^{(2)} &= \sum_{i=1}^{n_2} (y_{ir} - \overline{y}_r)\,(y_{i\ell} - \overline{y}_\ell) \qquad (r, \ell = 1, 2 \ldots, p). \end{aligned} \tag{6.30}$$

Si nota che gli elementi della matrice H_I si ottengono come prodotto di $n_1 - 1$ per gli elementi della matrice della varianze e covarianze tra le caratteristiche della matrice X, mentre gli elementi della matrice H_J si ottengono come prodotto di $n_2 - 1$ per gli elementi della matrice della varianze e covarianze tra le caratteristiche della matrice Y.

Vogliamo ora definire le matrici di non omogeneità tra i due cluster $H_{I \cap J}$ (*statistical between scatter matrix*) e di non omogeneità dell'unione di due cluster $H_{I \cup J}$ e le relative misure di non omogeneità statistiche.

La matrice $H_{I \cup J}$ si può esprimere come la somma di tre matrici di cardinalità $p \times p$, ossia la matrice di non omogeneità statistica H_I relativa al cluster I, la matrice di non omogeneità statistica H_J relativa al cluster J e la matrice di non omogeneità statistica $H_{I \cap J}$ tra i due cluster I e J, ossia

$$H_{I \cup J} = H_I + H_J + H_{I \cap J}. \tag{6.31}$$

Dalla (6.31) si ottiene la *misura di non omogeneità statistica relativa all'unione dei cluster* I e J, ossia:

$$\operatorname{tr} H_{I \cup J} = \operatorname{tr} H_I + \operatorname{tr} H_J + \operatorname{tr} H_{I \cap J}. \tag{6.32}$$

Ovviamente la misura di non omogeneità statistica tra i cluster può essere semplicemente calcolata come

$$\operatorname{tr} H_{I \cap J} = \operatorname{tr} H_{I \cup J} - \operatorname{tr} H_I - \operatorname{tr} H_J.$$

Esempio 6.3 Riconsideriamo la seguente matrice dei dati che si riferisce a due caratteristiche C_1 e C_2 osservate per cinque differenti individui I_1, I_2, I_3, I_4, I_5:

$$X = \begin{array}{c} \\ I_1 \\ I_2 \\ I_3 \\ I_4 \\ I_5 \end{array} \begin{array}{c} \begin{array}{cc} C_1 & C_2 \end{array} \\ \begin{pmatrix} 1 & 1 \\ 1 & 2 \\ 6 & 3 \\ 8 & 2 \\ 8 & 0 \end{pmatrix} \end{array}$$

e consideriamo due cluster $G_1 = \{I_1, I_2\}$ e $G_2 = \{I_3, I_4, I_5\}$.

Per il primo gruppo consideriamo le seguenti linee di codice

```
> X1<-data.frame(c1=c(1,1),c2=c(1,2))
> row.names(X1)<-c("I1","I2")
> X1 # visualizza il primo data frame
   c1 c2
I1  1  1
I2  1  2
>
> apply(X1,2,mean)
 c1  c2
1.0 1.5
>
> apply(X1,2,var)
 c1  c2
0.0 0.5
>
> W1<-cov(X1)
> W1 # visualizza la matrice delle varianze covarianze
   c1  c2
c1  0 0.0
c2  0 0.5
> # SECONDO METODO (n>1)
> n1<-nrow(X1)
> H1<-(n1-1)*W1
> H1 # visualizza la matrice di non omogeneita' statistica
   c1  c2
c1  0 0.0
c2  0 0.5
>
> tr1<-sum(diag(H1))
> tr1 # visualizza la misura di non omogeneita' statistica
[1] 0.5
```

Si nota che per il primo gruppo si ha $s_1^2 = 0$, $s_2^2 = 0.5$, $\mathrm{Cov}(C_1, C_2) = 0$ e la traccia della matrice di non omogeneità statistica del primo gruppo è $\mathrm{tr}\, H_I = 0.5$.

Per il secondo gruppo consideriamo le seguenti linee di codice:

```
> X2<-data.frame(c1=c(6,8,8),c2=c(3,2,0))
> row.names(X2)<-c("I3","I4","I5")
> X2 # visualizza il secondo data frame
   c1 c2
I3  6  3
```

```
I4  8  2
I5  8  0
>
> apply(X2,2,mean)
       c1       c2
7.333333 1.666667
>
> apply(X2,2,var)
       c1       c2
1.333333 2.333333
>
> W2<-cov(X2)
> W2 # visualizza la matrice delle varianze covarianze
          c1        c2
c1  1.333333 -1.333333
c2 -1.333333  2.333333
> # SECONDO METODO (n>1)
> n2<-nrow(X2)
> H2<-(n2-1)*W2
> H2 # visualizza la matrice di non omogeneita' statistica
          c1        c2
c1  2.666667 -2.666667
c2 -2.666667  4.666667
>
> tr2<-sum(diag(H2))
> tr2 # visualizza la misura di non omogeneita' statistica
[1] 7.333333
```

Si nota che $s_1^2 = 1.333333$, $s_2^2 = 2.333333$, $\mathrm{Cov}(C_1, C_2) = -1.333333$ e la traccia della matrice di non omogeneità statistica del secondo gruppo è $\mathrm{tr}\, H_J = 7.333333$.

Le misure di non omogeneità statistiche dei cluster $G_1 = \{I_1, I_2\}$ e $G_2 = \{I_3, I_4, I_5\}$, la misura di non omogeneità interna ai cluster (within) e la misura di non omogeneità tra i cluster (between) si possono ricavare nel seguente modo:

```
> X<-data.frame(c1=c(1,1,6,8,8),c2=c(1,2,3,2,0))
>  row.names(X)<-c("I1","I2","I3","I4","I5")
> X # visualizza il data frame X
   c1 c2
I1  1  1
I2  1  2
I3  6  3
I4  8  2
I5  8  0
> # PRIMO METODO (n>1)
> n<-nrow(X)
>  trHI<-(n-1)*sum(apply(X,2,var))
> trHI # visualizza la misura di non omogeneita' statistica
[1] 56
>
> X1<-data.frame(c1=c(1,1),c2=c(1,2))
> row.names(X1)<-c("I1","I2")
> X1 # visualizza il primo data frame
   c1 c2
I1  1  1
I2  1  2
```

```
>  # PRIMO METODO (n1>1)
> n1<-nrow(X1)
> tr1<-(n1-1)*sum(apply(X1,2,var))
> tr1 # visualizza la misura di non omogeneita' di G1
[1] 0.5
>
> X2<-data.frame(c1=c(6,8,8),c2=c(3,2,0))
> row.names(X2)<-c("I3","I4","I5")
> X2 # visualizza il secondo data frame
   c1 c2
I3  6  3
I4  8  2
I5  8  0
> # PRIMO METODO (n2>1)
> n2<-nrow(X2)
> tr2<-(n2-1)*sum(apply(X2,2,var))
> tr2 # visualizza la misura di non omogeneita' di G2
[1] 7.333333
>
> trWithin<- tr1+tr2
> trWithin # visualizza la misura di non omogeneita' interna
[1] 7.833333
>
> trBetween <- trHI-trWithin
> trBetween # visualizza la misura di non omogeneita' tra i cluster
[1] 48.16667
```

In conclusione, le misure di non omogeneità statistica per il primo e per il secondo cluster sono $\text{tr}\, H_I = 0.5$ e $\text{tr}\, H_J = 7.333333$, la misura di non omogeneità statistica tra i cluster è $\text{tr}\, H_{I\cap J} = 48.16667$ e la misura di non omogeneità relativa all'unione dei due cluster è $\text{tr}\, H_{I\cup J} = 56$, che ovviamente coincide con la misura di non omogeneità totale ricavata nell'Esempio 6.2. Si nota che la misura di non omogeneità di ciascuno dei cluster è minore della misura di non omogeneità ottenuta unendo i due cluster, ossia di $\text{tr}\, H_{I\cup J} = 56$. Inoltre, la misura di non omogeneità all'interno (within) $\text{tr}\, H_I + \text{tr}\, H_J = 0.5 + 7.333333 = 7.833333$ è inferiore della misura di non omogeneità tra i cluster (between) $\text{tr}\, H_{I\cap J} = 48.16667$.

Misure di non omogeneità: Caso generale

Se desideriamo partizionare un insieme $I = \{I_1, I_2, \ldots, I_n\}$ di n individui in m particolari cluster è possibile considerare un'estensione della (6.31). Infatti, denotando con $G_1, G_2, \ldots, G_m$ una partizione degli n individui $I = \{I_1, I_2, \ldots, I_n\}$ in m cluster, le matrici di non omogeneità statistica soddisfano la relazione:

$$H_{G_1\cup\ldots\cup G_m} = \sum_{\ell=1}^{m} H_{G_\ell} + \sum_{i<j} H_{G_i\cap G_j} + \sum_{i<j<k} H_{G_i\cap G_j\cap G_k} + \ldots + H_{G_1\cap\ldots\cap G_m} \tag{6.33}$$

Se denotiamo con

$$T = H_{G_1\cup\ldots\cup G_m} = H_I$$

la matrice di non omogeneità statistica relativa all'insieme totale $I = \{I_1, I_2, \ldots, I_n\}$ degli n individui, con

$$S = H_{G_1} + H_{G_2} + \ldots + H_{G_m}$$

la somma delle matrici di non omogeneità statistica relative ai singoli m cluster (*within*) e con

$$B = \sum_{i<j} H_{G_i \cap G_j} + \sum_{i<j<k} H_{G_i \cap G_j \cap G_k} + \ldots + H_{G_1 \cap \ldots \cap G_m}$$

la matrice di non omogeneità statistica tra i vari cluster considerati (*between*), l'equazione matriciale (6.33) si può scrivere

$$T = S + B \tag{6.34}$$

Le matrici T, S e B hanno cardinalità $p \times p$. Per ogni fissata matrice X dei dati (di cardinalità $n \times p$ corrispondente all'insieme $I = \{I_1, I_2, \ldots, I_n\}$ degli n individui) si ha che la matrice T è fissata. Invece, le matrici S e B dipendono strettamente dalla partizione in cluster dell'insieme I di individui considerata. Per ogni partizione dell'insieme I degli n individui in m fissati cluster, otteniamo un'equazione matriciale del tipo (6.34) da cui segue

$$\text{tr } T = \text{tr } S + \text{tr } B,$$

o equivalentemente

$$1 = \frac{\text{tr } S}{\text{tr } T} + \frac{\text{tr } B}{\text{tr } T}.$$

Poiché tr T è univocamente determinata per ogni matrice X di cardinalità $n \times p$, fissato il numero m di suddivisioni, i cluster dovrebbero essere individuati in modo da *minimizzare la misura di non omogeneità statistica all'interno dei cluster (within) e massimizzare la misura di non omogeneità statistica tra i gruppi (between).*

Una volta scelta la misura di distanza (o di similarità) si pone il problema di procedere alla scelta di un idoneo algoritmo di raggruppamento delle unità osservate. I metodi di raggruppamento si distinguono in tre tipi:

- metodi di enumerazione completa;

- metodi gerarchici;

- metodi non gerarchici.

Le misure di non omogeneità statistiche sono utilizzate per valutare, fissato il numero di cluster, la bontà della suddivisione in cluster ottenuta con i vari metodi (di enumerazione completa, gerarchici, non gerarchici).

6.6 Metodi di enumerazione completa

Supponiamo di considerare un insieme $\{I_1, I_2, \ldots, I_n\}$ di n individui e sia m il numero di cluster. Il numero di partizioni di un insieme di n individui in m cluster non vuoti può essere valutato calcolando *il numero di modi in cui è possibile sistemare n biglie distinte in m urne identiche con nessuna urna vuota*. Un importante risultato relativo al calcolo combinatorio è il seguente.

Proposizione 6.2 *Il numero di modi in cui è possibile sistemare n biglie distinte in m urne distinte tali che nessuna delle m urne sia vuota è*

$$R(n,m) = \sum_{k=0}^{m} \binom{m}{k} (-1)^k (m-k)^n, \tag{6.35}$$

quando l'ordine delle biglie in ciascuna urna è irrilevante.

Nella Proposizione 6.2 le m urne sono supposte distinte. Invece, in una partizione di n individui in m cluster di cui nessuno è vuoto, l'ordine degli m cluster è irrilevante. Pertanto, sussiste il risultato:

Proposizione 6.3 *Il numero totale di modi di partizionare n individui in un fissato numero m di cluster è dato da*

$$S(n,m) = \frac{1}{m!} \sum_{k=0}^{m} \binom{m}{k} (-1)^k (m-k)^n, \tag{6.36}$$

detto numero di Stirling del secondo tipo.

In particolare, se $m = 2$ dalla (6.36) segue che

$$S(n,2) = \frac{1}{2} \sum_{k=0}^{2} \binom{2}{k} (-1)^k (2-k)^n = \frac{1}{2} (2^n - 2) = 2^{n-1} - 1,$$

che rappresenta il *numero di partizioni di n individui in due cluster*.

Definiamo una funzione che calcola i numeri di Stirling del secondo tipo:

```
> stirling2<-function(n,m){
+       s<-0
+       if((m>=1)&(m<=n)){
+       for(k in seq(0,m)){
+               s<-s+(choose(m,k)*(-1)^k*(m-k)^n/factorial(m))}
+               return(c(s))
+               }}
>
> stirling2(6,1)
[1] 1
>
> stirling2(6,3)
[1] 90
>
> stirling2(8,4)
[1] 1701
```

Se $m = 1$ esiste un unico modo per sistemare n individui in un unico cluster. Se $n = 6$ e $m = 3$ esistono 90 modi di partizionare l'insieme costituito da 6 individui in 3 sottoinsiemi distinti

$$S(6,3) = \frac{1}{3!}\sum_{k=0}^{3}\binom{3}{k}(-1)^k(3-k)^6 = \frac{1}{6}[3^6 - 3\,2^6 + 3\,1^6] = 90.$$

Se, invece, $n = 8$ ed $m = 4$ esistono 1701 modi di partizionare l'insieme costituito da 8 individui in 4 sottoinsiemi distinti. Si nota che il numero di partizioni cresce in maniera esponenziale sia con n che con m.

I numeri di Stirling possono anche essere calcolati tramite la relazione di ricorrenza:

$$\begin{aligned} &S(n,1) = 1 \\ &S(n,n) = 1 \\ &S(n,m) = S(n-1,m-1) + m\,S(n-1,m) \qquad (2 \leq m \leq n-1) \end{aligned}$$

Un modo per risolvere il problema di clustering consiste nel determinare una partizione che soddisfa alcuni criteri di ottimalità. Questo criterio può essere dato in termini di una relazione funzionale che rifletta il livello di desiderabilità dei vari raggruppamenti. Tale relazione funzionale è spesso chiamata *funzione obiettivo*. Assegnata una funzione obiettivo (ad esempio, la misura di non omogeneità statistica interna ai cluster) la soluzione ottima per il problema di clustering può essere ottenuta, almeno concettualmente, valutando la funzione obiettivo per ogni possibile partizione dell'insieme degli n individui in cluster, scegliendo quella partizione che fornisce il valore ottimo della funzione obiettivo (il minimo valore per la misura di non omogeneità statistica interna ai cluster). Questa procedura è chiamata *clustering con enumerazione completa*; si nota che essa è impraticabile a meno che n (il numero degli individui) e m (il numero di cluster) non siano piccoli.

Se il numero m dei cluster non è fissato a priori, il numero totale di partizioni di n individui è:

$$B_n = \sum_{m=1}^{n} S(n,m). \tag{6.37}$$

I numeri $B_1, B_2, \ldots$ sono detti *numeri di Bell*. Definiamo una funzione che calcola il numero di Bell B_n:

```
> sumstirling2<-function(n){
+       s<-0
+       for(k in seq(1,n))
+       s<-s+stirling2(n,k)
+       return(c(s))
+       }
>
> sumstirling2(6)
[1] 203
```

Se, ad esempio, $n = 6$, allora

$$B_6 = \sum_{m=1}^{6} S(6, m) = 1 + \frac{62}{2} + \frac{540}{6} + \frac{1560}{24} + \frac{1800}{120} + 1 = 203,$$

che rappresenta il *numero totale di partizioni di 6 individui senza fissare a priori il numero di cluster*.

Nel metodo di enumerazione completa con fissato il numero di cluster, è possibile utilizzare le misure di non omogeneità statistiche per ottenere la migliore partizione in cluster. Infatti, fissato il numero m di suddivisioni, occorre individuare quella partizione in cluster che *minimizzi la misura di non omogeneità statistica all'interno dei cluster (within) e massimizzi la misura di non omogeneità statistica tra i gruppi (between)*.

Occorre sottolineare che le tecniche di enumerazione completa sono computazionalmente onerose poiché prevedono il calcolo delle funzioni di non omogeneità per ogni possibile partizione dell'insieme totale di n individui in m cluster. Pertanto, nella pratica si adottano *metodi di raggruppamento gerarchici e non gerarchici* che operano su una sottoclasse delle partizioni degli n individui in cluster. Tali metodi saranno esaminati nel prossimo capitolo.

Capitolo 7

Analisi dei cluster: parte 2

7.1 Introduzione

Il problema principale che si presenta utilizzando le tecniche di enumerazione completa è che esse sono computazionalmente onerose poiché prevedono il calcolo della funzione obiettivo per ogni possibile partizione dell'insieme totale di n individui in m cluster. Essendo le tecniche di enumerazione completa computazionalmente impraticabili (a meno che n ed m non siano piccoli), spesso si adottano i metodi di raggruppamento gerarchici e non gerarchici che operano su una sottoclasse delle partizioni degli n individui in cluster.

I *metodi gerarchici di clustering* eseguono una sequenza ordinata di operazioni della stessa natura. Tali metodi sono distinti *in agglomerativi* che procedono per aggregazioni successive delle unità partendo da n gruppi formati da un singolo individuo e *divisivi* che partono da un solo gruppo formato da tutte le unità e procedono a divisioni successive fino a giungere a gruppi formati da una sola unità. I metodi gerarchici hanno due vantaggi: quello di fornire una visione completa dell'insieme in termini di distanza (o di coefficienti di similarità) seppure condizionata dalla scelta del metodo seguito e quello di non comportare la scelta a priori del numero di cluster oppure la scelta a priori di parametri per la determinazione automatica del loro numero. Invece, uno svantaggio di tali metodi è che essi non consentono di riallocare gli individui che sono stati già classificati ad un livello precedente dell'analisi.

I *metodi non gerarchici di clustering* consentono, a differenza delle tecniche di tipo gerarchico, di riallocare gli individui già classificati ad un livello precedente dell'analisi. A differenza dei metodi gerarchici, l'obiettivo finale dei metodi non gerarchici è quello di ottenere un'unica partizione degli n individui di partenza in cluster. In molti metodi non gerarchici di clustering si assume che il numero di cluster in cui suddividere l'insieme totale degli n individui sia fissato a priori dal ricercatore, mentre in altri tale numero è determinato nel corso dell'analisi.

7.2 Metodi gerarchici

I metodi di clustering gerarchico possono essere di due tipi: *agglomerativi* e *divisivi*. I *metodi gerarchici di tipo agglomerativo* partono da una situazione in cui si hanno n cluster distinti ognuno contenente un solo individuo per giungere, attraverso successive unioni dei cluster meno distanti tra loro, ad una situazione in cui si ha un solo cluster che contiene tutti gli n individui. Invece, i *metodi gerarchici di tipo divisivo* partono da una situazione in cui si ha un solo cluster che contiene tutti gli n individui per giungere, attraverso successive divisioni dei cluster più distanti tra loro, ad una situazione in cui si hanno n cluster distinti ognuno contenente un solo individuo. Quindi, i metodi gerarchici di tipo agglomerativo procedono con una sequenza di successive unioni degli n individui iniziali in gruppi, mentre i metodi gerarchici di tipo divisivo procedono con una sequenza di successive divisioni dell'insieme degli n individui in partizioni sempre più fini.

L'obiettivo finale dei metodi gerarchici non è quello di ottenere una singola partizione degli n individui di partenza, ma di ottenere una sequenza di partizioni che possono essere rappresentate graficamente mediante una struttura ad albero, detta *dendrogramma*, nella quale sull'insieme delle ordinate sono riportati i livelli di distanza mentre sull'asse delle ascisse sono riportati i singoli individui. Ad ogni livello di distanza corrisponde una partizione, mentre ad ogni partizione corrispondono infiniti livelli di distanza compresi tra quelli che individuano due successive unioni o divisioni.

Il dendrogramma fornisce un quadro completo della struttura dell'insieme in termini delle misure di distanza tra gli individui. Fissando un opportuno livello della funzione distanza e analizzando tale struttura, il ricercatore può indirettamente stabilire a quale stadio dell'analisi gerarchica occorre fermarsi ottenendo la partizione dell'insieme totale di individui in cluster. Occorre osservare che è possibile effettuare a priori l'opportuna scelta del livello della funzione distanza se l'analisi gerarchica è effettuata per finalità comparative o se si hanno precedenti conoscenze del fenomeno in osservazione. In generale, comunque, tale scelta può anche essere effettuata a posteriori con il grosso vantaggio di avere una visione globale della struttura dell'insieme totale di individui in termini di distanza.

Le tecniche gerarchiche di tipo agglomerativo si sono rivelate particolarmente utili in biologia e zoologia per raggruppare piante e animali rispetto a caratteristiche di tipo genetico. Ad esempio, in tassonomia biologica gli individui sono raggruppati in specie, le specie in generi, i generi in famiglie e così via.

Nel seguito di questo paragrafo analizzeremo approfonditamente vari metodi gerarchici di tipo agglomerativo.

Sia $I = \{I_1, I_2, \ldots, I_n\}$ un insieme di n individui o entità appartenenti ad una popolazione. Nei metodi gerarchici di clustering di tipo agglomerativo si valuta inizialmente la matrice delle distanza D tra gli individui. La funzione distanza più comunemente utilizzata è la *metrica euclidea*. A partire da una di tali matrici, si considera inizialmente un insieme di n cluster $\{I_1\}$, $\{I_2\}$, ..., $\{I_n\}$. I due cluster più vicini, ad esempio I_i e I_j, sono uniti in un singolo

cluster e quindi il nuovo insieme è costituito da $n-1$ cluster, ossia $\{I_1\}$, $\{I_2\}$, ..., $\{I_i, I_j\}$, ..., $\{I_n\}$. Ripetendo tale procedura sequenzialmente si ottengono insiemi di $n-2$ cluster, $n-3$ cluster e così via fino a che si ottiene un unico cluster di n individui, ossia l'insieme originario di tutti gli individui I.

Molti metodi di analisi gerarchica sono caratterizzati da una struttura comune che si riflette in un algoritmo generale che può essere così esplicitato:

◇ **Algoritmo**

- **Step 1:** A partire dalla matrice X originaria dei dati o dalla matrice scalata, considerare la matrice delle distanze D (o la matrice di similarità S) tra gli individui (considerati come singoli cluster contenenti un solo individuo);

- **Step 2:** individuare la coppia di cluster meno distanti (o più somiglianti) e raggruppare in un unico cluster i due cluster meno distanti (o più somiglianti); inoltre, calcolare la distanza (o similarità) di questo nuovo cluster, originato dall'agglomerazione, da tutti gli altri gruppi già esistenti;

- **Step 3:** costruire una nuova matrice di distanza (o di similarità) che risulterà ridotta di una riga e di una colonna rispetto a quella che la precede; infatti, le due righe e due colonne dei gruppi agglomerati sono sostituite con una singola riga e una singola colonna contenenti le nuove distanze;

- **Step 4:** operare sulla matrice così ottenuta a partire dal passo 2 fino ad esaurire tutte le possibilità di raggruppamento, raggiungendo alla fine una matrice di cardinalità 2×2. La procedura richiede $n-1$ iterazioni.

- **Step 5:** rappresentare graficamente il processo di agglomerazione attraverso un dendrogramma che riporta sull'asse verticale il livello di distanza a cui avviene l'agglomerazione e sull'asse orizzontale riporta gli individui. Ad ogni livello di distanza corrisponde una partizione.

I cinque passi sopra delineati costituiscono lo scheletro comune di molti metodi gerarchici. Le uniche differenze esistenti si riscontreranno nel passo 1 e nel passo 2. Per quanto riguarda il passo 1, la *scelta della misura di distanza* (o di misura di similarità) influenza il metodo richiedendo più o meno forti proprietà. Il passo 2 è quello che caratterizza ciascun metodo. Infatti, ciascun metodo si differenzia dagli altri per il modo in cui si individuano i due cluster meno distanti (o più somiglianti) e per *il modo in cui si determina la distanza (o similarità) che intercorre tra il nuovo cluster ottenuto e i rimanenti*. Il procedimento indicato nel passo 2 è specifico per ciascun metodo e influenza la sua denominazione.

L'analisi gerarchica di tipo agglomerativo viene effettuata in R attraverso la funzione

```
> hclust(d, method = "complete")
```

dove

- d rappresenta un oggetto (che individua una struttura di similarità o distanza) creato tramite la funzione dist();

- method seleziona il metodo gerarchico agglomerativo (di default è complete).

Alcune delle opzioni disponibili per method sono:

(1) metodo del legame singolo (single);

(2) metodo del legame completo (complete);

(3) metodo del legame medio (average);

(4) metodo del centroide (centroid);

(5) metodo della mediana (median).

La funzione hclust() *produce come output una lista*, i cui elementi sono:

- la sequenza del processo di agglomerazione ($merge);

- un vettore, la cui lunghezza corrisponde al numero di iterazioni, che indica il livello di distanza alla quale è avvenuta l'unione tra due cluster ($height);

- un'opportuna permutazione delle unità (individui) finalizzata alla costruzione del dendrogramma ($order);

- un vettore delle etichette che contrassegnano le varie unità ($labels).

Per ottenere il dendrogramma si impiega la funzione:

$> \text{plot}(z, \text{labels} = \text{NULL}, \text{hang} = -1, \text{main} = \text{"Dendrogramma"}, \text{sub} = \text{NULL}, \text{xlab} = \text{NULL})$

dove

- z è l'oggetto creato (output) dalla funzione hclust();

- labels è un vettore di etichette per i rami del dendrogramma (di default impiega i nomi delle righe del data frame);

- hang determina l'altezza alla quale le etichette vengono visualizzate al di sotto del dendrogramma (un valore negativo pone le etichette al di sotto dell'ordinata nulla);

- main, sub, xlab sono comandi per la finestra grafica.

Descriviamo ora come operano i vari metodi gerarchici agglomerativi precedentemente introdotti.

(1) **Metodo del legame singolo** (Nearest neighbour method)
In questo metodo la distanza tra i gruppi G_1 (contenente n_1 individui) e G_2 (contenente n_2 individui) è definita come la *minima* tra tutte le $n_1 n_2$ distanze che si possono calcolare tra ogni individuo di G_1 e ogni individuo di G_2, così come illustrato nella Figura 7.1.

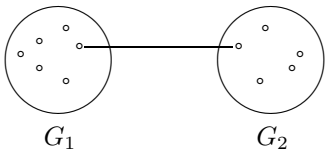

Figura 7.1: Metodo del legame singolo.

Nella procedura gerarchica si considera inizialmente, ossia al livello 0, un insieme di n cluster $\{I_1\}$, $\{I_2\}$, ..., $\{I_n\}$. Al passo successivo si cerca nella matrice D delle distanze il coefficiente di distanza minima e si raggruppano nello stesso cluster G_{ij} i due individui I_i e I_j associati secondo tale coefficiente. Nel caso i coefficienti di distanza minima siano più di uno, si attua una scelta arbitraria tra di essi. Al livello 1 quindi si modifica la matrice delle distanze valutando le distanze di G_{ij} da ogni altro individuo I_k non appartenente a G_{ij} mediante la seguente relazione:

$$d_{(ij),k} = \min(d_{ik}, d_{jk}). \tag{7.1}$$

In altre parole, la distanza dell'individuo I_k dal cluster G_{ij} si ottiene scegliendo la più piccola tra le due distanze d_{ik} e d_{jk}. Quindi, al livello 1 si costruisce una nuova matrice D_1 di cardinalità $(n-1) \times (n-1)$ costituita da G_{ij}, considerato come un unico elemento, e dagli $n-2$ individui esterni a G_{ij}. Ad ogni passo successivo, dopo che i cluster G_u e G_v sono stati uniti scegliendo dalla precedente matrice delle distanze i due cluster più vicini, la distanza tra il nuovo cluster, denotato con G_{uv}, e un altro cluster G_z è così definita

$$d_{(uv),z} = \min(d_{uz}, d_{vz}). \tag{7.2}$$

La quantità $d_{(uv),z}$ rappresenta la misura di distanza tra gli elementi meno distanti dei cluster G_{uv} e G_z. La procedura si ripete fino ad ottenere un unico cluster formato da tutti gli individui.

Osservazioni Un vantaggio del metodo del legame singolo è di consentire di *individuare gruppi di qualsiasi forma* (anche non ellissoidali, ossia anche con forme allungate e scarsamente omogenei al loro interno) e di mettere in luce eventuali valori anomali meglio di altre tecniche gerarchiche. Uno svantaggio di tale metodo è che invece esso può dare origine alla *formazione di una catena* tra gli individui. Infatti, tale metodo ha il difetto di basare l'unione di due cluster G_1 e G_2, di numerosità n_1 e n_2, su un solo legame, quello corrispondente alla distanza più piccola tra gli $n_1 n_2$ individui esistenti. Quindi, ad un certo livello di distanza d, si può verificare che si vengano a trovare nello stesso cluster individui piuttosto dissimili. La formazione di una catena risulta essere particolarmente grave quando esistono cluster ben delineati, ma non ben separati. In questo caso, a causa dell'effetto della catena, il metodo del legame singolo può fallire nell'individuazione dei cluster distinti pessendo presenti un piccolo numero di punti intermedi tra tali cluster.

Esempio 7.1 Nella seguente matrice delle misure sono riportate due caratteristiche C_1 e C_2 osservate per cinque differenti individui I_1, I_2, I_3, I_4, I_5:

$$X = \begin{array}{c} \\ I_1 \\ I_2 \\ I_3 \\ I_4 \\ I_5 \end{array} \begin{array}{c} \begin{array}{cc} C_1 & C_2 \end{array} \\ \begin{pmatrix} 1 & 1 \\ 1 & 2 \\ 6 & 3 \\ 8 & 2 \\ 8 & 0 \end{pmatrix} \end{array}$$

Utilizzando R definiamo e scaliamo tale matrice dei dati:

```
> X<-data.frame(c1=c(1,1,6,8,8),c2=c(1,2,3,2,0))
> row.names(X)<-c("I1","I2","I3","I4","I5")
> X # visualizza il data frame X
   c1 c2
I1  1  1
I2  1  2
I3  6  3
I4  8  2
I5  8  0
>
> Z<-scale(X)
> Z # visualizza la matrice scalata
           c1         c2
I1 -1.0663057 -0.5262348
I2 -1.0663057  0.3508232
I3  0.3367281  1.2278812
I4  0.8979417  0.3508232
I5  0.8979417 -1.4032928
attr(,"scaled:center")
 c1  c2
4.8 1.6
attr(,"scaled:scale")
      c1       c2
3.563706 1.140175
```

Si nota che center fornisce le medie campionarie relative alle due caratteristiche della matrice iniziale dei dati, mentre scale le deviazioni standard campionarie.

Calcoliamo ora la matrice delle distanze utilizzando la metrica euclidea e applichiamo il metodo gerarchico del legame singolo.

```
> d<-dist(Z,method="euclidean",diag=TRUE,upper=TRUE)
> d # visualizza la matrice delle distanze
         I1       I2       I3       I4       I5
I1 0.000000 0.877058 2.246203 2.151162 2.151162
I2 0.877058 0.000000 1.654610 1.964247 2.633475
I3 2.246203 1.654610 0.000000 1.041245 2.690360
I4 2.151162 1.964247 1.041245 0.000000 1.754116
I5 2.151162 2.633475 2.690360 1.754116 0.000000
>
> hls<-hclust(d,method="single")
>
> str(hls) # visualizza informazioni sull'oggetto cluster
List of 7
```

```
$ merge       : int [1:4, 1:2] -1 -3 1 -5 -2 -4 2 3
$ height      : num [1:4] 0.877 1.041 1.655 1.754
$ order       : int [1:5] 5 1 2 3 4
$ labels      : chr [1:5] "I1" "I2" "I3" "I4" ...
$ method      : chr "single"
$ call        : language hclust(d = d, method = "single")
$ dist.method: chr "euclidean"
- attr(*, "class")= chr "hclust"
```

In Tabella 7.1 è analizzato l'output ottenuto con il metodo del legame singolo. I risultati di $merge sono stati disposti su due colonne: *i numeri con il segno negativo indicano i singoli individui*, mentre *i numeri positivi indicano i cluster che si formano*. Inoltre, $height indica la distanza in cui è avvenuta l'agglomerazione tra i cluster.

Tabella 7.1: Analisi dell'output del metodo gerarchico del legame singolo

		Agglomerazione	Distanza
-1	-2	Al livello 1 si uniscono gli individui I_1 e I_2	0.877
-3	-4	Al livello 2 si uniscono gli individui I_3 e I_4	1.041
1	2	Al livello 3 si uniscono il primo cluster (formato dagli individui I_1 e I_2) con il secondo cluster (formato dagli individui I_3 e I_4)	1.655
-5	3	Al livello 4 si unisce il terzo cluster (formato dagli individui I_1, I_2, I_3, I_4) con l'individuo I_5	1.754

Costruiamo ora il dendrogramma utilizzando R. Le seguenti linee di codice

```
> plot(hls,hang=-1,xlab="Metodo gerarchico agglomerativo",
+ sub="del legame singolo")
> axis(side=4,at=round(c(0,hls$height),2))
```

producono il grafico di Figura 7.2
L'istruzione axis(side = 4, at = round(c(0, hls$height), 2)) permette di costruire l'asse delle altezze alla destra del grafico arrotondando i numeri alla seconda cifra decimale.

Per visualizzare il taglio del dendrogramma in corrispondenza di un salto nelle distanze si utilizza la funzione

abline(h = NULL, lty = NULL)

dopo il comando plot() dove

- h è l'altezza alla quale si inserisce il taglio;

- lty definisce lo stile della linea del taglio (1 per una linea continua, 2 per una linea tratteggiata, ...)

Ad esempio, se nelle precedenti linee di codice aggiungiamo l'istruzione

```
> abline(h=1.2,lty=2,col="red")
```

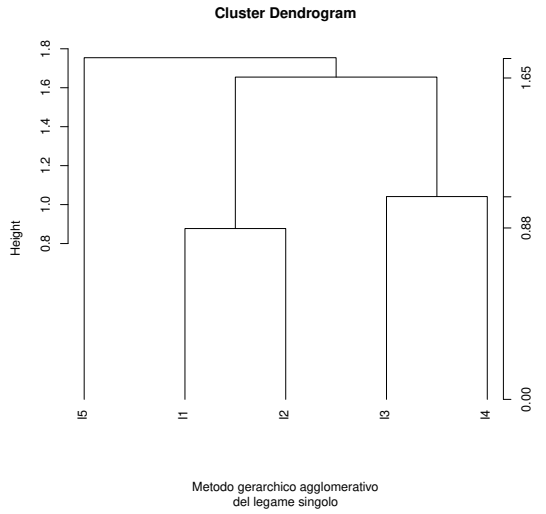

Figura 7.2: Dendrogramma ottenuto con il metodo del legame singolo.

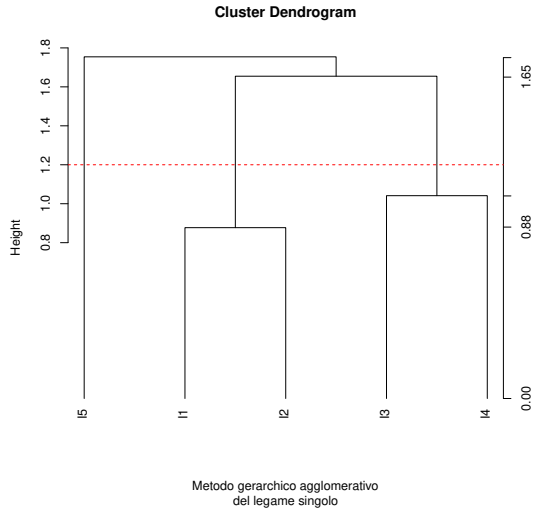

Figura 7.3: Livello di distanza pari a 1.2 nel dendrogramma di Figura 7.2.

è possibile visualizzare nel grafico precedente un livello di distanza pari a 1.2 mediante una linea tratteggiata, come mostrato nel grafico in Figura 7.3.

Si nota che se si fissa un livello di distanza pari a 1.2 l'insieme di individui resta suddiviso nei tre cluster $\{I_1, I_2\}$, $\{I_3, I_4\}$ e $\{I_5\}$.

Spieghiamo ora gradualmente (il che è possibile poiché sono stati considerati soltanto cinque individui) come è avvenuto il processo di agglomerazione con il metodo del legame singolo. Riconsideriamo la matrice delle distanze ottenuta con R, ossia

$$D = \begin{array}{c} \\ I_1 \\ I_2 \\ I_3 \\ I_4 \\ I_5 \end{array} \begin{array}{c} \begin{array}{ccccc} I_1 & I_2 & I_3 & I_4 & I_5 \end{array} \\ \begin{pmatrix} 0.000000 & \boxed{0.877058} & 2.246203 & 2.151162 & 2.151162 \\ \boxed{0.877058} & 0.000000 & 1.654610 & 1.964247 & 2.633475 \\ 2.246203 & 1.654610 & 0.000000 & 1.041245 & 2.690360 \\ 2.151162 & 1.964247 & 1.041245 & 0.000000 & 1.754116 \\ 2.151162 & 2.633475 & 2.690360 & 1.754116 & 0.000000 \end{pmatrix} \end{array}. \tag{7.3}$$

Livello 1: Essendo $d_{12} = 0.877058$ il più piccolo valore della matrice delle distanze, gli individui I_1 e I_2 sono uniti formando un unico gruppo. Le distanze tra questo gruppo e i tre rimanenti gruppi di singoli individui $\{I_3\}$, $\{I_4\}$, $\{I_5\}$ si ottengono dalla matrice delle distanze d tramite la (7.2):

$$\begin{aligned} d_{(1,2),3} &= \min(d_{13}, d_{23}) = \min(2.246203, 1.654610) = 1.654610 \\ d_{(1,2),4} &= \min(d_{14}, d_{24}) = \min(2.151162, 1.964247) = 1.964247 \\ d_{(1,2),5} &= \min(d_{15}, d_{25}) = \min(2.151162, 2.633475) = 2.151162. \end{aligned}$$

È quindi possibile costruire una nuova matrice delle distanze D_1 di ordine 4:

$$D_1 = \begin{array}{c} \\ I_{1,2} \\ I_3 \\ I_4 \\ I_5 \end{array} \begin{array}{c} \begin{array}{cccc} I_{1,2} & I_3 & I_4 & I_5 \end{array} \\ \begin{pmatrix} 0.000000 & 1.654610 & 1.964247 & 2.151162 \\ 1.654610 & 0.000000 & \boxed{1.041245} & 2.690360 \\ 1.964247 & \boxed{1.041245} & 0.000000 & 1.754116 \\ 2.151162 & 2.690360 & 1.754116 & 0.000000 \end{pmatrix} \end{array}.$$

Livello 2: Il più piccolo valore della distanza nella matrice D_1 è $d_{34} = 1.041245$ e quindi gli individui I_3 e I_4 sono uniti formando un unico gruppo. Le distanze tra il gruppo $\{I_3, I_4\}$ e i due rimanenti gruppi di individui $\{I_1, I_2\}$ e I_5 si ottengono dalla matrice D_1 tramite la (7.2) come segue:

$$\begin{aligned} d_{(3,4),(1,2)} &= \min(d_{3,(1,2)}, d_{4,(12)}) = \min(1.654610, 1.964247) = 1.654610 \\ d_{(3,4),5} &= \min(d_{35}, d_{45}) = \min(2.690360, 1.754116) = 1.754116. \end{aligned}$$

È quindi possibile costruire una nuova matrice delle distanze D_2 di ordine 3:

$$D_2 = \begin{array}{c} \\ I_{1,2} \\ I_{3,4} \\ I_5 \end{array} \begin{array}{c} \begin{array}{ccc} I_{1,2} & I_{3,4} & I_5 \end{array} \\ \begin{pmatrix} 0.000000 & \boxed{1.654610} & 2.151162 \\ \boxed{1.654610} & 0.000000 & 1.754116 \\ 2.151162 & 1.754116 & 0.000000 \end{pmatrix} \end{array}.$$

Livello 3: Il più piccolo valore della distanza nella matrice D_2 è $d_{(12),(34)} = 1.654610$ e quindi i gruppi $\{I_1, I_2\}$ e $\{I_3, I_4\}$ sono uniti formando un unico gruppo. Le distanze tra il gruppo $\{I_1, I_2, I_3, I_4\}$ e il rimanente gruppo $\{I_5\}$ si ricava dalla matrice D_2 tramite la (7.2)

$$d_{(1,2,3,4),5} = \min(d_{(1,2),5}, d_{(3,4),5}) = \min(2.151162, 1.754116) = 1.754116$$

e quindi la nuova matrice delle distanze di ordine 2 è:

$$D_3 = \begin{array}{c} \\ I_{1,2,3,4} \\ I_5 \end{array} \begin{array}{c} \begin{array}{cc} I_{1,2,3,4} & I_5 \end{array} \\ \begin{pmatrix} 0.00000 & 1.754116 \\ 1.754116 & 0.000000 \end{pmatrix} \end{array}.$$

Livello 4: Unendo i gruppi $\{I_1, I_2, I_3, I_4\}$ e $\{I_5\}$ si ottiene un unico cluster contenente tutti e cinque gli individui.

La sequenza delle agglomerazioni del metodo del legame singolo è pertanto rappresentata nella Tabella 7.2 in cui sono anche indicati i corrispondenti livelli di distanza.

Tabella 7.2: Metodo gerarchico del legame singolo

Numero di cluster	Cluster	Livello di distanza
5	$\{I_1\}, \{I_2\}, \{I_3\}, \{I_4\}, \{I_5\}$	
4	$\{I_1, I_2\}, \{I_3\}, \{I_4\}, \{I_5\}$	0.877058
3	$\{I_1, I_2\}, \{I_3, I_4\}, \{I_5\}$	1.041245
2	$\{I_1, I_2, I_3, I_4\}, \{I_5\}$	1.654610
1	$\{I_1, I_2, I_3, I_4, I_5\}$	1.754116

Con il metodo del legame singolo applicato alla matrice scalata Z, una suddivisione in 2 cluster conduce alla partizione $C_1 = \{I_5\}$ e $C_2 = \{I_1, I_2, I_3, I_4\}$. La misura di non omogeneità totale relativa alla matrice scalata Z è tr $T = 8$. Poiché il primo cluster ha un unico elemento, la misura di non omogeneità interna è nulla; inoltre, la misura di non omogeneità relativa al secondo cluster è 4.530588. La misura di non omogeneità interna ai cluster (within) per la matrice scalata è tr $S = 4.530588$ e la misura di non omogeneità tra i cluster (between) è tr $B =$ tr $T -$ tr $S = 8 - 4.530588 = 3.469412$. Quindi, la suddivisione in due cluster, ottenuta applicando il metodo del legame singolo alla matrice scalata, non è soddisfacente.

(2) **Metodo del legame completo** (Furthest neighbour method)
In questo metodo la distanza tra i gruppi G_1 (contenente n_1 individui) e G_2 (contenente n_2 individui) è definita come la massima tra tutte le $n_1 n_2$ distanze che si possono calcolare tra ogni individuo di G_1 e ogni individuo di G_2, così come illustrato nella Figura 7.4.

La *massima distanza* esistente tra gli individui dei due cluster rappresenta il *diametro della sfera che contiene tutti i punti appartenenti ai due gruppi.*

Nella procedura si considera inizialmente, ossia al livello 0, un insieme di n cluster $\{I_1\}$, $\{I_2\}$, ..., $\{I_n\}$. Al passo successivo si cerca nella matrice D

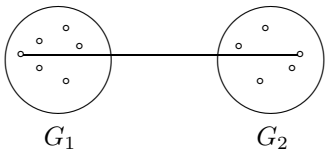

Figura 7.4: Metodo del legame completo.

delle distanze il coefficiente di distanza minima e si raggruppano nello stesso cluster G_{ij} i due individui I_i e I_j associati secondo tale coefficiente. Nel caso i coefficienti di distanza minima siano più di uno, si attua una scelta arbitraria tra di essi. Al livello 1 quindi si modifica la matrice delle distanze valutando le distanze di G_{ij} da ogni altro individuo I_k non appartenente a G_{ij} mediante la seguente relazione

$$d_{(ij),k} = \max(d_{ik}, d_{jk}). \tag{7.4}$$

In altre parole, la distanza dell'individuo I_k dal cluster G_{ij} si ottiene scegliendo la più grande tra le due distanze d_{ik} e d_{jk}. Quindi, al livello 1 si costruisce una nuova matrice di cardinalità $(n-1)\times(n-1)$ costituita da G_{ij}, considerato come un unico elemento, e dagli $n-2$ individui esterni a G_{ij}. Ad ogni passo successivo, dopo che i cluster G_u e G_v sono stati uniti scegliendo dalla precedente matrice delle distanze i due cluster più vicini, la distanza tra il nuovo cluster, denotato con G_{uv}, e un altro cluster G_z è così definita

$$d_{(uv),z} = \max(d_{uz}, d_{vz}) \tag{7.5}$$

La quantità $d_{(uv),z}$ rappresenta la misura di distanza tra i più lontani elementi dei cluster G_{uv} e G_z. La procedura si ripete fino ad ottenere un unico cluster formato da tutti gli individui.

Osservazioni Il metodo del legame completo identifica soprattutto gruppi di *forma ellissoidale*, ossia una serie di punti che si addensano intorno ad un nucleo centrale. Questo algoritmo privilegia l'omogeneità tra gli elementi del gruppo a scapito della differenziazione tra i gruppi. Il dendrogramma costruito con questo metodo ha i rami molto più lunghi rispetto al dendrogramma ottenuto con il metodo del legame singolo poiché i gruppi si formano a livelli di distanza maggiori.

Esempio 7.2 Alla matrice `d` delle distanze dell'Esempio 7.1 applichiamo ora il metodo del legame completo applicando R

```
> hlc<-hclust(d,method="complete")
>
> str(hlc) # visualizza informazioni sull'oggetto cluster
List of 7
 $ merge       : int [1:4, 1:2] -1 -3 1 -5 -2 -4 2 3
 $ height      : num [1:4] 0.877 1.041 2.246 2.69
 $ order       : int [1:5] 5 1 2 3 4
 $ labels      : chr [1:5] "I1" "I2" "I3" "I4" ...
```

```
$ method      : chr "complete"
$ call        : language hclust(d = d, method = "complete")
$ dist.method: chr "euclidean"
- attr(*, "class")= chr "hclust"
```

In Tabella 7.3 è analizzato l'output ottenuto con il metodo del legame completo.

Tabella 7.3: Analisi dell'output del metodo gerarchico del legame completo

		Agglomerazione	Distanza
-1	-2	Al livello 1 si uniscono gli individui I_1 e I_2	0.877
-3	-4	Al livello 2 si uniscono gli individui I_3 e I_4	1.041
1	2	Al livello 3 si uniscono il primo cluster (formato dagli individui I_1 e I_2) con il secondo cluster (formato dagli individui I_3 e I_4)	2.246
-5	3	Al livello 4 si unisce il terzo cluster (formato dagli individui I_1, I_2, I_3, I_4) con l'individuo I_5	2.690

Costruiamo ora il dendrogramma utilizzando R. Le seguenti linee di codice

```
> plot(hlc,hang=-1,xlab="Metodo gerarchico agglomerativo",
+ sub="del legame completo")
> axis(side=4,at=round(c(0,hlc$height),2))
```

producono il grafico di Figura 7.5.

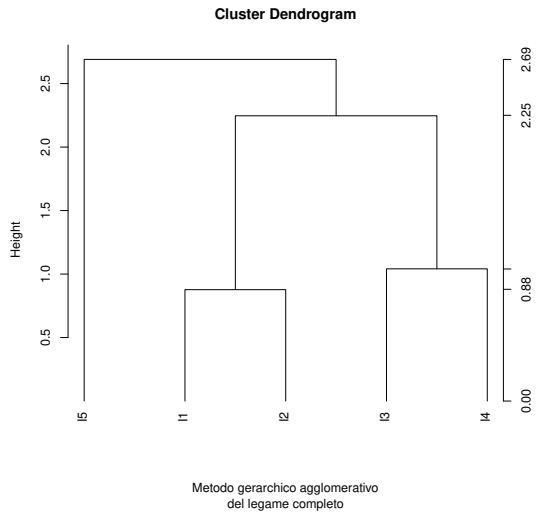

Figura 7.5: Dendrogramma ottenuto con il metodo del legame completo.

Si nota che con il metodo del legame completo la suddivisione in cluster non cambia rispetto al caso del legame singolo, ma variano alcuni livelli di

distanza relativi alle aggregazioni; in particolare, confrontando i dendrogrammi di Figura 7.2 e di Figura 7.5 si nota che risulta più accentuato il salto tra i livelli di distanza corrispondenti alle ultime due aggregazioni.

Spieghiamo ora gradualmente come è avvenuto il processo di agglomerazione con il metodo del legame completo. Riconsideriamo la matrice delle distanze (7.3) ottenuta in R.

Livello 1 Essendo $d_{12} = 0.877058$ il più piccolo valore della matrice delle distanze, gli individui I_1 e I_2 sono uniti formando un unico gruppo. Le distanze tra questo gruppo e i tre rimanenti gruppi di singoli individui $\{I_3\}$, $\{I_4\}$, $\{I_5\}$ si ottengono dalla matrice delle distanze `d` tramite la (7.5):

$$d_{(1,2),3} = \max(d_{13}, d_{23}) = \max(2.246203, 1.654610) = 2.246203$$
$$d_{(1,2),4} = \max(d_{14}, d_{24}) = \max(2.151162, 1.964247) = 2.151162$$
$$d_{(1,2),5} = \max(d_{15}, d_{25}) = \max(2.151162, 2.633475) = 2.633475$$

È quindi possibile costruire una nuova matrice delle distanze D_1 di ordine 4:

$$D_1 = \begin{array}{c} \\ I_{1,2} \\ I_3 \\ I_4 \\ I_5 \end{array} \begin{array}{c} \begin{array}{cccc} I_{1,2} & I_3 & I_4 & I_5 \end{array} \\ \begin{pmatrix} 0.000000 & 2.246203 & 2.151162 & 2.633475 \\ 2.246203 & 0.000000 & 1.041245 & 2.690360 \\ 2.151162 & 1.041245 & 0.000000 & 1.754116 \\ 2.633475 & 2.690360 & 1.754116 & 0.000000 \end{pmatrix} \end{array}.$$

Livello 2 Il più piccolo valore della distanza nella matrice D_1 è $d_{34} = 1.041245$ e quindi gli individui I_3 e I_4 sono uniti formando un unico gruppo. Le distanze tra il gruppo $\{I_3, I_4\}$ e i due rimanenti gruppi di individui $\{I_1, I_2\}$ e I_5 si ottengono dalla matrice D_1 tramite la (7.5) come segue:

$$d_{(3,4),(1,2)} = \max(d_{3,(1,2)}, d_{4,(12)}) = \max(2.246203, 2.151162) = 2.246203$$
$$d_{(3,4),5} = \max(d_{35}, d_{45}) = \max(2.690360, 1.754116) = 2.690360$$

È quindi possibile costruire una nuova matrice delle distanze D_2 di ordine 3:

$$D_2 = \begin{array}{c} \\ I_{1,2} \\ I_{3,4} \\ I_5 \end{array} \begin{array}{c} \begin{array}{ccc} I_{1,2} & I_{3,4} & I_5 \end{array} \\ \begin{pmatrix} 0.000000 & 2.246203 & 2.633475 \\ 2.246203 & 0.000000 & 2.690360 \\ 2.633475 & 2.690360 & 0.000000 \end{pmatrix} \end{array}.$$

Livello 3 Il più piccolo valore della distanza nella matrice D_2 è $d_{(12),(34)} = 2.246203$ e quindi i gruppi $\{I_1, I_2\}$ e $\{I_3, I_4\}$ sono uniti formando un unico gruppo. Le distanze tra il gruppo $\{I_1, I_2, I_3, I_4\}$ e il rimanente gruppo $\{I_5\}$ si ricava dalla matrice D_2 tramite la (7.5)

$$d_{(1,2,3,4),5} = \max(d_{(1,2),5}, d_{(3,4),5}) = \max(2.633475, 2.690360) = 2.690360$$

e quindi la nuova matrice delle distanze di ordine 2 è:

$$D_3 = \begin{array}{c} \\ I_{1,2,3,4} \\ I_5 \end{array} \begin{array}{c} \begin{array}{cc} I_{1,2,3,4} & I_5 \end{array} \\ \begin{pmatrix} 0.00000 & 2.690360 \\ 2.690360 & 0.000000 \end{pmatrix} \end{array}.$$

Livello 4 Unendo i gruppi $\{I_1, I_2, I_3, I_4\}$ e $\{I_5\}$ si ottiene un unico cluster contenente tutti e cinque gli individui.

Tabella 7.4: Metodo gerarchico del legame completo

Numero di cluster	Cluster	Livello di distanza
5	$\{I_1\}, \{I_2\}, \{I_3\}, \{I_4\}, \{I_5\}$	
4	$\{I_1, I_2\}, \{I_3\}, \{I_4\}, \{I_5\}$	0.877058
3	$\{I_1, I_2\}, \{I_3, I_4\}, \{I_5\}$	1.041245
2	$\{I_1, I_2, I_3, I_4\}, \{I_5\}$	2.246203
1	$\{I_1, I_2, I_3, I_4, I_5\}$	2.690360

La sequenza delle agglomerazioni del metodo del legame completo è pertanto rappresentata nella Tabella 7.4 in cui sono anche indicati i corrispondenti livelli di distanza.

Con il metodo del legame completo applicato alla matrice scalata Z, una suddivisione in 2 cluster conduce alla stessa partizione $C_1 = \{I_5\}$ e $C_2 = \{I_1, I_2, I_3, I_4\}$ del metodo del legame singolo.

(3) **Metodo del legame medio** (Average linkage method)
In questo metodo la distanza tra i gruppi G_1 e G_2 è definita come la media aritmetica delle distanze tra tutte le coppie di unità che compongono i due gruppi, così come illustrato nella Figura 7.6.

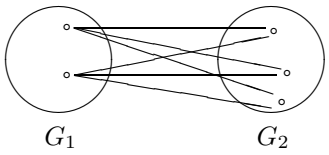

Figura 7.6: Metodo del legame medio.

Nella procedura si considera inizialmente, ossia al livello 0, un insieme di n cluster $\{I_1\}$, $\{I_2\}$, ..., $\{I_n\}$. Al passo successivo si cerca nella matrice D delle distanze il coefficiente di distanza minima e si raggruppano nello stesso cluster G_{ij} i due individui I_i e I_j associati secondo tale coefficiente. Nel caso i coefficienti di distanza minima siano più di uno, si attua una scelta arbitraria tra di essi. Al livello 1 quindi si modifica la matrice delle distanze valutando le distanze di G_{ij} da ogni altro individuo I_k non appartenente a G_{ij} mediante la seguente relazione

$$d_{(i,j),k} = \frac{1}{2}\left(d_{i,k} + d_{j,k}\right) \qquad (k = 1, 2, \ldots, n; k \neq i, j) \tag{7.6}$$

Quindi, al livello 1 si costruisce una nuova matrice di cardinalità $(n-1)\times(n-1)$ costituita da G_{ij}, considerato come un unico elemento, e dagli $n-2$ individui

esterni a G_{ij}. Ad ogni passo successivo, dopo che i cluster G_u e G_v sono stati uniti scegliendo dalla precedente matrice delle distanze i cluster più vicini, la distanza tra il nuovo cluster, denotato con G_{uv}, e un altro cluster G_z è così definita

$$\begin{aligned} d_{(uv),z} &= \frac{1}{(N_u + N_v)\, N_z} \sum_{\{i:I_i \in G_{uv}\}} \sum_{\{j:I_j \in G_z\}} d_{ij} \\ &= \frac{1}{(N_u + N_v)\, N_z} \sum_{\{i:I_i \in G_u\}} \sum_{\{j:I_j \in G_z\}} d_{ij} + \frac{1}{(N_u + N_v)\, N_z} \sum_{\{i:I_i \in G_v\}} \sum_{\{j:I_j \in G_z\}} d_{ij} \\ &= \frac{N_u}{N_u + N_v}\, d_{uz} + \frac{N_v}{N_u + N_v}\, d_{vz}, \end{aligned} \tag{7.7}$$

dove N_u, N_v e N_z sono rispettivamente il numero di individui del cluster G_u, del cluster G_v e del cluster G_z. La quantità $d_{(uv),z}$ rappresenta la misura di distanza media tra gli elementi dei cluster G_{uv} e G_z. La procedura si ripete fino ad ottenere un unico cluster formato da tutti gli individui.

Osservazioni Uno svantaggio del metodo del legame medio è che se le misure dei due cluster da unire sono molto differenti la distanza $d_{(uv),z}$ sarà molto vicina a quella del cluster più numeroso. Infatti, se risulta $N_u >> N_v$ dalla (7.7) segue che $d_{(u,v),z} \simeq d_{u,z}$.

Esempio 7.3 Alla matrice `d` delle distanze dell'Esempio 7.1 applichiamo ora il metodo del legame medio applicando R

```
> hlm<-hclust(d,method="average")
> str(hlm) # visualizza informazioni sull'oggetto cluster
List of 7
 $ merge      : int [1:4, 1:2] -1 -3 1 -5 -2 -4 2 3
 $ height     : num [1:4] 0.877 1.041 2.004 2.307
 $ order      : int [1:5] 5 1 2 3 4
 $ labels     : chr [1:5] "I1" "I2" "I3" "I4" ...
 $ method     : chr "average"
 $ call       : language hclust(d = d, method = "average")
 $ dist.method: chr "euclidean"
 - attr(*, "class")= chr "hclust"
```

In Tabella 7.5 è analizzato l'output ottenuto con il metodo del legame medio.

Tabella 7.5: Analisi dell'output del metodo gerarchico del legame medio

		Agglomerazione	Distanza
-1	-2	Al livello 1 si uniscono gli individui I_1 e I_2	0.877
-3	-4	Al livello 2 si uniscono gli individui I_3 e I_4	1.041
1	2	Al livello 3 si uniscono il primo cluster (formato dagli individui I_1 e I_2) con il secondo cluster (formato dagli individui I_3 e I_4)	2.004
-5	3	Al livello 4 si unisce il terzo cluster (formato dagli individui I_1, I_2, I_3, I_4) con l'individuo I_5	2.307

Costruiamo ora il dendrogramma utilizzando R. Le seguenti linee di codice

```
> plot(hlm,hang=-1,xlab="Metodo gerarchico agglomerativo",
+ sub="del legame medio")
> axis(side=4,at=round(c(0,hlm$height),2))
```

producono il grafico di Figura 7.7.

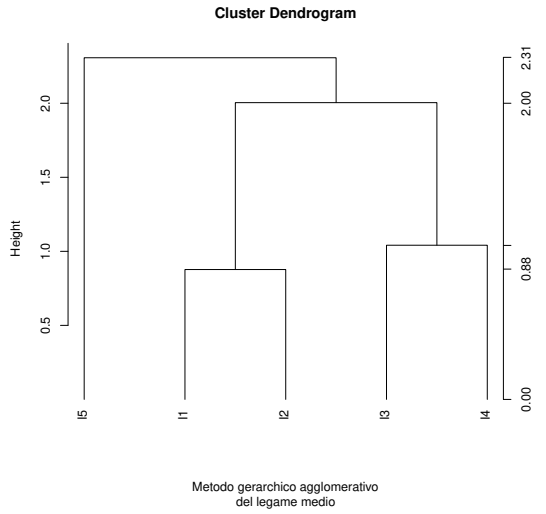

Figura 7.7: Dendrogramma della matrice d con il metodo del legame medio.

Si nota che con il metodo del legame medio la suddivisione in cluster non cambia rispetto al caso del legame singolo e del legame completo, ma variano alcuni livelli di distanza relativi alle aggregazioni; in particolare, il dendrogramma di Figura 7.7 è intermedio tra quello di Figura 7.2 e quello di Figura 7.5.

Spieghiamo ora gradualmente come è avvenuto il processo di agglomerazione con il metodo del legame medio. Riconsideriamo la matrice delle distanze (7.3) ottenuta in R.

Livello 1 Essendo $d_{12} = 0.877058$ il più piccolo valore della matrice delle distanze, gli individui I_1 e I_2 sono uniti formando un unico gruppo. Le distanze tra questo gruppo e i tre rimanenti gruppi di singoli individui $\{I_3\}$, $\{I_4\}$, $\{I_5\}$ si ottengono dalla matrice delle distanze d tramite la (7.7):

$$
\begin{aligned}
d_{(1,2),3} &= (d_{1,3} + d_{2,3})/2 = (2.246203 + 1.654610)/2 = 1.950406 \\
d_{(1,2),4} &= (d_{1,4} + d_{2,4})/2 = (2.151162 + 1.964247)/2 = 2.057704 \\
d_{(1,2),5} &= (d_{1,5} + d_{2,5})/2 = (2.151162 + 2.633475)/2 = 2.392319
\end{aligned}
$$

È quindi possibile costruire una nuova matrice delle distanze D_1 di ordine 4:

$$D_1 = \begin{array}{c} \\ I_{1,2} \\ I_3 \\ I_4 \\ I_5 \end{array} \begin{array}{c} \begin{array}{cccc} I_{1,2} & I_3 & I_4 & I_5 \end{array} \\ \begin{pmatrix} 0.000000 & 1.950406 & 2.057704 & 2.392319 \\ 1.950406 & 0.000000 & \boxed{1.041245} & 2.690360 \\ 2.057704 & \boxed{1.041245} & 0.000000 & 1.754116 \\ 2.392319 & 2.690360 & 1.754116 & 0.000000 \end{pmatrix} \end{array}.$$

Livello 2 Il più piccolo valore della distanza nella matrice D_1 è $d_{34} = 1.041245$ e quindi gli individui I_3 e I_4 sono uniti formando un unico gruppo. Le distanze tra il gruppo $\{I_3, I_4\}$ e i due rimanenti gruppi di individui $\{I_1, I_2\}$ e I_5 si ottengono dalla matrice D_1 tramite la (7.7) come segue:

$$d_{(3,4),(1,2)} = (d_{3,(1,2)} + d_{4,(12)})/2 = (1.950406 + 2.057704)/2 = 2.004055$$
$$d_{(3,4),5} = (d_{3,5} + d_{4,5})/2 = (2.690360 + 1.754116)/2 = 2.222238.$$

È quindi possibile costruire una nuova matrice delle distanze D_2 di ordine 3:

$$D_2 = \begin{array}{c} \\ I_{1,2} \\ I_{3,4} \\ I_5 \end{array} \begin{array}{c} \begin{array}{ccc} I_{1,2} & I_{3,4} & I_5 \end{array} \\ \begin{pmatrix} 0.000000 & \boxed{2.004055} & 2.392319 \\ \boxed{2.004055} & 0.000000 & 2.222238 \\ 2.392319 & 2.222238 & 0.000000 \end{pmatrix} \end{array}.$$

Livello 3 Il più piccolo valore della distanza nella matrice D_2 è $d_{(12),(34)} = 2.004055$ e quindi i gruppi $\{I_1, I_2\}$ e $\{I_3, I_4\}$ sono uniti formando un unico gruppo. Le distanze tra il gruppo $\{I_1, I_2, I_3, I_4\}$ e il rimanente gruppo $\{I_5\}$ si ricava dalla matrice D_2 tramite la (7.7)

$$d_{(1,2,3,4),5} = \frac{2}{4} d_{(1,2),5} + \frac{2}{4} d_{(3,4),5} = (2.392319 + 2.222238)/2 = 2.307279$$

e quindi la nuova matrice delle distanze di ordine 2 è:

$$D_3 = \begin{array}{c} \\ I_{1,2,3,4} \\ I_5 \end{array} \begin{array}{c} \begin{array}{cc} I_{1,2,3,4} & I_5 \end{array} \\ \begin{pmatrix} 0.00000 & \boxed{2.307279} \\ \boxed{2.307279} & 0.000000 \end{pmatrix} \end{array}.$$

Livello 4 Unendo i gruppi $\{I_1, I_2, I_3, I_4\}$ e $\{I_5\}$ si ottiene un unico cluster contenente tutti e cinque gli individui.

La sequenza delle agglomerazioni del metodo del legame medio è pertanto rappresentata nella Tabella 7.6 in cui sono anche indicati i corrispondenti livelli di distanza.

Con il metodo del legame medio applicato alla matrice scalata Z, una suddivisione in 2 cluster conduce alla stessa partizione $C_1 = \{I_5\}$ e $C_2 = \{I_1, I_2, I_3, I_4\}$ dei precedenti due metodi.

Nei metodi agglomerativi del *legame singolo*, del *legame completo* e del *legame medio* si può utilizzare una *qualsiasi misura di distanza*. Invece, nel *metodo*

Tabella 7.6: Metodo gerarchico del legame medio

Numero di cluster	Cluster	Livello di distanza
5	$\{I_1\}, \{I_2\}, \{I_3\}, \{I_4\}, \{I_5\}$	
4	$\{I_1, I_2\}, \{I_3\}, \{I_4\}, \{I_5\}$	0.877058
3	$\{I_1, I_2\}, \{I_3, I_4\}, \{I_5\}$	1.041245
2	$\{I_1, I_2, I_3, I_4\}, \{I_5\}$	2.004055
1	$\{I_1, I_2, I_3, I_4, I_5\}$	2.307279

del centroide e nel *metodo della mediana* si considera la distanza euclidea e si lavora con una *matrice* $D^{(2)}$ che contiene i *quadrati delle singole distanze euclidee.*

(4) **Metodo del centroide**

In questo metodo la distanza tra il gruppi G_1 e il gruppo G_2 è definita come la distanza tra i centroidi, ossia tra le medie campionarie calcolate sugli individui appartenenti ai due gruppi, così come illustrato nella Figura 7.8.

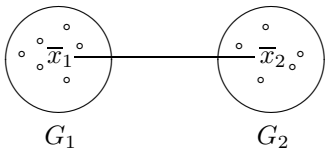

Figura 7.8: Metodo del centroide.

Nella procedura si considera inizialmente, ossia al livello 0, un insieme di n cluster $\{I_1\}$, $\{I_2\}$, ..., $\{I_n\}$. Al passo successivo si cerca nella matrice $D^{(2)}$, contenente i quadrati delle singole distanze euclidee, il coefficiente di distanza minima e si raggruppano nello stesso cluster G_{ij} i due individui I_i e I_j associati secondo tale coefficiente. Nel caso i coefficienti di distanza minima siano più di uno, si attua una scelta arbitraria tra di essi. Al livello 1 quindi si modifica la matrice deli quadrati delle distanze valutando i quadrati delle distanze di G_{ij} da ogni altro individuo I_k non appartenente a G_{ij} mediante la relazione:

$$d^2_{(ij),k} = \sum_{r=1}^{p} (\overline{x}_{(i,j),r} - \overline{x}_{k,r})^2 = \frac{1}{2}(d^2_{ik} + d^2_{jk}) - \frac{1}{4} d^2_{ij}, \qquad (k \neq i, j) \quad (7.8)$$

dove

$$\overline{x}_{(i,j),r} = \frac{1}{2}\,(x_{i,r} + x_{j,r}) \qquad \overline{x}_{k,r} = x_{k,r} \qquad (r = 1, 2, \ldots, p). \qquad (7.9)$$

Quindi, al livello 1 si modifica la matrice delle misure nel seguente modo:

$$X_1 = \begin{array}{c} \\ I_1 \\ I_2 \\ \vdots \\ I_{i,j} \\ \vdots \\ I_n \end{array} \begin{array}{c} \begin{array}{cccc} C_1 & C_2 & \dots & C_p \end{array} \\ \begin{pmatrix} x_{11} & x_{12} & \dots & x_{1p} \\ x_{21} & x_{22} & \dots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ \overline{x}_{(i,j),1} & \overline{x}_{(i,j),2} & \dots & \overline{x}_{(i,j),p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \dots & x_{np} \end{pmatrix} \end{array} \tag{7.10}$$

ottenendo una matrice di cardinalità $(n-1)\times p$. Ad ogni passo successivo, dopo che i cluster G_u e G_v sono stati uniti scegliendo dalla precedente matrice dei quadrati delle distanze euclidee i due cluster più vicini, la distanza tra il nuovo cluster, denotato con G_{uv}, e un altro cluster G_z è così definita

$$d^2_{(uv),z} = \sum_{k=1}^{p} (\overline{x}_{(u,v),k} - \overline{x}_{(z),k})^2 = \frac{N_u}{N_u+N_v}\, d^2_{uz} + \frac{N_v}{N_u+N_v}\, d^2_{vz} - \frac{N_u N_v}{(N_u+N_v)^2}\, d^2_{u,v}, \tag{7.11}$$

dove

$$\begin{aligned} \overline{x}_{(u,v),r} &= \frac{1}{N_u+N_v} \sum_{\{i:I_i\in G_{uv}\}} x_{i,r} \\ &= \frac{1}{N_u+N_v} \sum_{\{i:I_i\in G_u\}} x_{i,r} + \frac{1}{N_u+N_v} \sum_{\{i:I_i\in G_v\}} x_{i,r} \\ &= \frac{N_u}{N_u+N_v}\, \overline{x}_{(u),r} + \frac{N_v}{N_u+N_v}\, \overline{x}_{(v),r} \\ &\qquad\qquad (r = 1,2,\dots,p) \end{aligned} \tag{7.12}$$

$$\overline{x}_{(z),r} = \frac{1}{N_z} \sum_{k:I_k\in G_z} x_{kr}$$

e dove N_u, N_v e N_z denotano rispettivamente il numero di individui del cluster G_u, G_v e G_z. La procedura si ripete fino ad ottenere un unico cluster formato da tutti gli individui.

Osservazioni Il metodo del centroide può dare origine a fenomeni gravitazionali, per cui i gruppi grandi tendono ad attrarre al loro interno i piccoli gruppi. Inoltre, *le distanze in cui si verificano le successive agglomerazioni possono essere non crescenti*. Uno svantaggio del metodo del centroide è che se le misure dei due cluster da unire sono molto differenti il centroide del nuovo cluster sarà molto vicino a quello del cluster più numeroso. Infatti, se risulta $N_u >> N_v$ dalla prima delle (7.12) segue che $\overline{x}_{(u,v),r} \simeq \overline{x}_{(u),r}$.

Esempio 7.4 Nella seguente matrice delle misure sono riportate due caratteristiche C_1 e C_2, con le stesse unità di misura, osservate per cinque differenti

individui I_1, I_2, I_3, I_4, I_5:

$$X = \begin{array}{c} \\ I_1 \\ I_2 \\ I_3 \\ I_4 \\ I_5 \end{array} \begin{array}{c} \begin{array}{cc} C_1 & C_2 \end{array} \\ \begin{pmatrix} 36 & 20 \\ 35 & 25 \\ 40 & 21 \\ 37 & 28 \\ 33 & 24 \end{pmatrix} \end{array}$$

Utilizzando R definiamo la matrice dei dati:

```
> X<-data.frame(c1=c(36,35,40,37,33),c2=c(20,25,21,28,24))
> row.names(X)<-c("I1","I2","I3","I4","I5")
> X # visualizza il data frame X
   c1 c2
I1 36 20
I2 35 25
I3 40 21
I4 37 28
I5 33 24
```

Possiamo rappresentare i cinque punti relativi agli individui I_1, I_2, I_3, I_4, I_5 in uno scatterplot. Le seguenti linee di codice

```
> plot(X$c1,X$c2,col="red",xlab="C1",
+ ylab="C2",ylim=c(0,30))
> text(X$c1,X$c2+0.8,c("I1","I2","I3","I4","I5"))
> abline(lm(X$c2~X$c1),lty=2,col="blue")
```

producono lo scatterplot di Figura 7.9

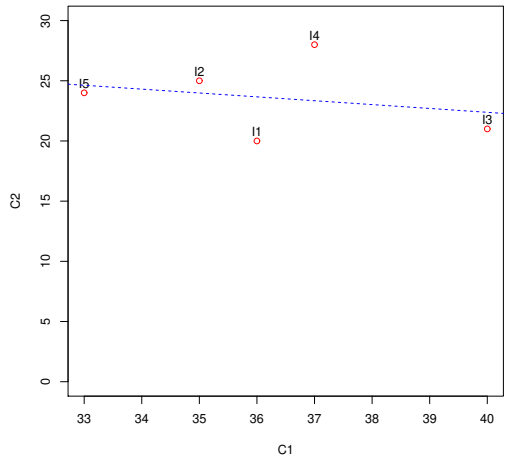

Figura 7.9: Scatterplot associato ai cinque individui.

Calcoliamo ora la matrice contenente i quadrati delle distanze euclidee:

```
> d<-dist(X,method="euclidean",diag=TRUE,upper=TRUE)
>
> d2<-d^2
> d2 # visualizza la matrice con i quadrati delle distanze euclidee
   I1 I2 I3 I4 I5
I1  0 26 17 65 25
I2 26  0 41 13  5
I3 17 41  0 58 58
I4 65 13 58  0 32
I5 25  5 58 32  0
```

Applichiamo ora il metodo gerarchico del centroide utilizzando i quadrati delle distanze euclidee:

```
> hc<-hclust(d2,method="centroid")
> str(hc) # visualizza informazioni sull'oggetto cluster
List of 7
 $ merge      : int [1:4, 1:2] -2 -1 -4 2 -5 -3 1 3
 $ height     : num [1:4] 5 17 21.3 35.7
 $ order      : int [1:5] 1 3 4 2 5
 $ labels     : chr [1:5] "I1" "I2" "I3" "I4" ...
 $ method     : chr "centroid"
 $ call       : language hclust(d = d2, method = "centroid")
 $ dist.method: chr "euclidean"
 - attr(*, "class")= chr "hclust"
```

In Tabella 7.7 è analizzato l'output ottenuto con il metodo del centroide.

Tabella 7.7: Analisi dell'output del metodo gerarchico del centroide

		Agglomerazione	Distanza
-2	-5	Al livello 1 si uniscono gli individui I_2 e I_5	5.0
-1	-3	Al livello 2 si uniscono gli individui I_1 e I_3	17.0
-4	1	Al livello 3 si uniscono l'individuo I_4 con il primo cluster (formato dagli individui I_2 e I_5)	21.3
2	3	Al livello 4 si unisce il secondo cluster (formato dagli individui I_1 e I_3) con il terzo cluster (formato dagli individui I_2,I_4 e I_5)	35.7

Costruiamo ora il dendrogramma utilizzando R. Le seguenti linee di codice

```
> plot(hc,hang=-1,xlab="Metodo gerarchico agglomerativo",
+ sub="del centroide")
> axis(side=4,at=round(c(0,hc$height),2))
```

producono il grafico di Figura 7.10.

Spieghiamo ora gradualmente come è avvenuto il processo di agglomerazione con il metodo del centroide. Analizzando la matrice dei quadrati delle distanze euclidee si nota che occorre unire gli individui I_2 e I_5 ad una distanza $d^2 = 5$.

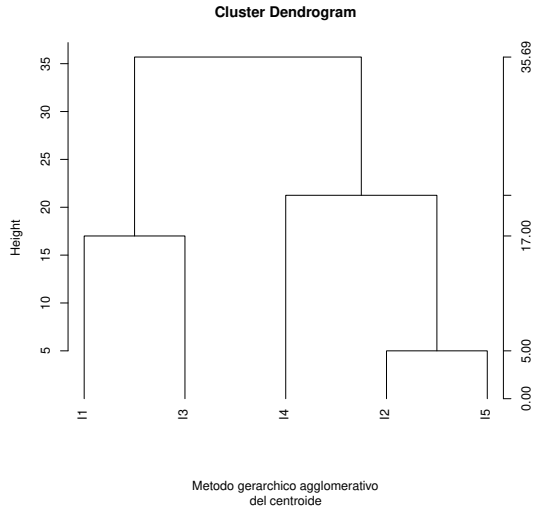

Figura 7.10: Dendrogramma ottenuto con il metodo del centroide.

Otteniamo una nuova matrice delle misure

$$X_1 = \begin{array}{c} \\ I_1 \\ I_{2,5} \\ I_3 \\ I_4 \end{array} \begin{array}{c} \begin{array}{cc} C_1 & C_2 \end{array} \\ \begin{pmatrix} 36 & 20 \\ 34 & 24.5 \\ 40 & 21 \\ 37 & 28 \end{pmatrix} \end{array}$$

essendo $x_{(2,5),1} = (35 + 33)/2 = 34$ e $x_{(2,5),2} = (25 + 24)/2 = 24.5$. Per questa nuova matrice calcoliamo la matrice dei quadrati delle distanze euclidee ottenendo:

```
       I1    I25     I3     I4
I1   0.00 24.25 17.00 65.00
I25 24.25  0.00 48.25 21.25
I3  17.00 48.25  0.00 58.00
I4  65.00 21.25 58.00  0.00
```

Si nota che gli individui I_1 e I_3 si uniscono ad un livello di distanza $d^2 = 17.00$ e quindi otteniamo una nuova matrice delle misure:

$$X_2 = \begin{array}{c} \\ I_{1,3} \\ I_{2,5} \\ I_4 \end{array} \begin{array}{c} \begin{array}{cc} C_1 & C_2 \end{array} \\ \begin{pmatrix} 38 & 20.5 \\ 34 & 24.5 \\ 37 & 28 \end{pmatrix} \end{array}$$

essendo $x_{(1,3),1} = (36 + 40)/2 = 38$ e $x_{(1,3),2} = (20 + 21)/2 = 20.5$. Per questa nuova matrice calcoliamo la matrice dei quadrati delle distanze euclidee ottenendo:

```
      I13   I25    I4
I13  0.00 32.00 57.25
I25 32.00  0.00 21.25
I4  57.25 21.25  0.00
```

Si nota che gli individui I_2 I_4 e I_5 si uniscono ad un livello di distanza $d^2 = 21.25$. In questo passaggio *si differenziano il metodo del centroide e il metodo della mediana*. Il metodo del centroide calcola il centro di gravità del nuovo cluster e quindi otteniamo una nuova matrice delle misure:

$$X_3 = \begin{matrix} \\ I_{1,3} \\ I_{2,4,5} \end{matrix} \begin{matrix} \begin{matrix} C_1 & C_2 \end{matrix} \\ \begin{pmatrix} 38 & 20.5 \\ 105/3 & 77/3 \end{pmatrix} \end{matrix}$$

poiché applicando la (7.12) risulta

$$\overline{x}_{((2,5),4),1} = \frac{2}{3}\,34 + \frac{1}{3}\,37 = \frac{105}{3}, \qquad \overline{x}_{((2,5),4),2} = \frac{2}{3}\,24.5 + \frac{1}{3}\,28 = \frac{77}{3}.$$

Per questa nuova matrice calcoliamo la matrice dei quadrati delle distanze euclidee ottenendo:

```
          I13      245
I13   0.00000 35.69444
245  35.69444  0.00000
```

Quindi tutti individui si uniscono in un unico cluster ad un livello di distanza $d^2 = 35.69444$.

(5) **Metodo della mediana**
Il metodo della mediana è simile a quello del centroide, con la differenza che *la procedura è indipendente dalla numerosità dei cluster*. Infatti, quando due gruppi si aggregano, il nuovo centroide è calcolato come la semisomma dei due centroidi precedenti. Il livello 1 della procedura è lo stesso del metodo del centroide. Ad ogni passo successivo, dopo che i cluster G_u e G_v sono stati uniti scegliendo dalla precedente matrice dei quadrati delle distanze euclidee i due cluster più vicini, la distanza tra il nuovo cluster, denotato con G_{uv}, e un altro cluster G_z è così definita:

$$d^2_{(uv),z} = \sum_{k=1}^{p} (\overline{x}_{(u,v),k} - \overline{x}_{(z),k})^2 = \frac{1}{2}\,d^2_{u,z} + \frac{1}{2}\,d^2_{v,z} - \frac{1}{4}\,d^2_{u,v}, \tag{7.13}$$

dove

$$\overline{x}_{(uv),r} = \frac{1}{2}\left(\overline{x}_{(u),r} + \overline{x}_{(v),r}\right) \qquad (r = 1, 2, \ldots, p). \tag{7.14}$$

La procedura si ripete fino ad ottenere un unico cluster formato da tutti gli individui.
Osservazioni Il metodo della mediana, così come il metodo del legame singolo, può dare origine alla formazione di una catena tra gli individui.

Esempio 7.5 Alla matrice contenente i quadrati delle distanze euclidee dell'Esempio 7.4 applichiamo ora il metodo della mediana utilizzando R

```
> hmed<-hclust(d2,method="median")
>
> str(hmed) # visualizza informazioni sull'oggetto cluster
List of 7
 $ merge      : int [1:4, 1:2] -2 -1 -4 2 -5 -3 1 3
 $ height     : num [1:4] 5 17 21.3 39.3
 $ order      : int [1:5] 1 3 4 2 5
 $ labels     : chr [1:5] "I1" "I2" "I3" "I4" ...
 $ method     : chr "median"
 $ call       : language hclust(d = d2, method = "median")
 $ dist.method: chr "euclidean"
 - attr(*, "class")= chr "hclust"
```

In Tabella 7.8 è analizzato l'output ottenuto con il metodo della mediana.

Tabella 7.8: Analisi dell'output del metodo gerarchico della mediana

		Agglomerazione	Distanza
-2	-5	Al livello 1 si uniscono gli individui I_2 e I_3	5.0
-1	-3	Al livello 2 si uniscono gli individui I_1 e I_5	17.0
-4	1	Al livello 3 si uniscono l'individuo I_4 con il primo cluster (formato dagli individui I_2 e I_5)	21.3
2	3	Al livello 4 si unisce il secondo cluster (formato dagli individui I_1 e I_3) con il terzo cluster (formato dagli individui I_2,I_4 e I_5)	39.3

Costruiamo ora il dendrogramma utilizzando R. Le seguenti linee di codice

```
> plot(hmed,hang=-1,xlab="Metodo gerarchico agglomerativo",
+ sub="della mediana")
> axis(side=4,at=round(c(0,hmed$height),2))
```

producono il grafico di Figura 7.11.

Si nota che in questo caso con il metodo della mediana la suddivisione in cluster non cambia rispetto al caso del centroide, ma varia l'ultimo livello di distanza relativo alle aggregazioni. Infatti, a differenza dal centroide, nel metodo della mediana otteniamo una nuova matrice delle misure:

$$X_3 = \begin{array}{c} \\ I_{1,3} \\ I_{2,4,5} \end{array} \begin{array}{c} \begin{array}{cc} C_1 & C_2 \end{array} \\ \begin{pmatrix} 38 & 20.5 \\ 71/2 & 52.5/2 \end{pmatrix} \end{array}$$

poiché applicando la (7.14) risulta

$$\overline{x}_{((2,5),4),1} = \frac{34+37}{2} = \frac{71}{2}, \qquad \overline{x}_{((2,5),4),2} = \frac{24.5+28}{2} = \frac{52.5}{2}.$$

Per questa nuova matrice calcoliamo la matrice dei quadrati delle distanze euclidee ottenendo:

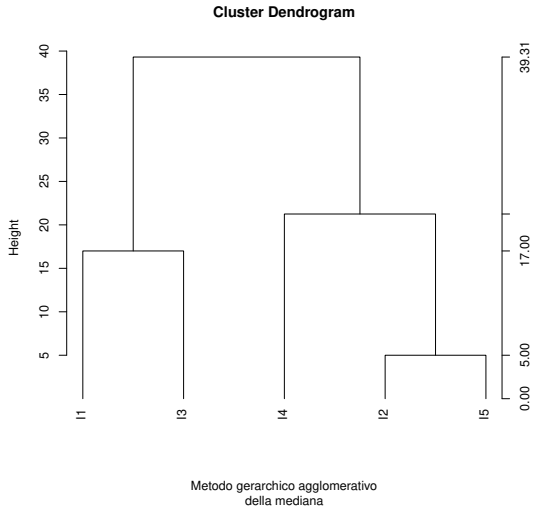

Figura 7.11: Dendrogramma ottenuto con il metodo della mediana.

```
         I13      245
I13   0.0000  39.3125
245  39.3125   0.0000
```

Quindi tutti individui si uniscono in un unico cluster ad un livello di distanza $d^2 = 39.3125$.

(6) **Metodo di Lance e Williams**

Lance e Williams (1967) hanno considerato uno schema ricorsivo in cui il calcolo della matrice dei quadrati delle distanze d^2_{k+1} al livello $(k+1)$-esimo dipende soltanto dai valori d^2_k della matrice delle distanze al livello k-esimo. Questo schema ricorsivo include, come vedremo, i metodi del *legame singolo*, del *legame completo*, del *legame medio*, del *centroide* e della *mediana*.

Nella procedura si considera inizialmente, ossia al livello 0, un insieme di n clusters $\{I_1\}$, $\{I_2\}$, ..., $\{I_n\}$. Al passo successivo si cerca nella matrice dei quadrati delle distanze Euclidee il coefficiente di distanza minima e si raggruppano nello stesso cluster G_{ij} i due individui I_i e I_j associati secondo tale coefficiente. Nel caso i coefficienti di distanza minima siano più di uno, si attua una scelta arbitraria tra di essi. Al livello $k+1$ $(k = 0, 1, 2, \ldots, n-2)$, dopo che i clusters G_u e G_v sono stati uniti scegliendo dalla precedente matrice delle distanze i clusters più vicini, la distanza tra il nuovo cluster, denotato con G_{uv}, e un altro cluster C_z è così definita

$$d^2_{(uv),z} = \alpha_u\, d^2_{uz} + \alpha_v\, d^2_{vz} + \beta\, d^2_{uv} + \gamma \left| d^2_{uz} - d^2_{vz} \right|, \tag{7.15}$$

dove α_u, α_v, β e γ sono dei parametri che dipendono dalla particolare procedura di clustering gerarchico scelta. Consideriamo ora le cinque procedure precedentemente descritte:

- *Metodo del legame singolo* La (7.2) mostra che occorre considerare la minima delle distanze, ossia

$$d^2_{(uv),z} = \frac{1}{2}\, d^2_{u,z} + \frac{1}{2}\, d^2_{v,z} - \frac{1}{2}\, |d^2_{u,z} - d^2_{v,z}|$$

e quindi confrontando tale espressione con la (7.15) si ha

$$\alpha_u = \frac{1}{2}, \quad \alpha_v = \frac{1}{2}, \quad \beta = 0, \quad \gamma = -\frac{1}{2}.$$

- *Metodo del legame completo* La (7.5) afferma che occorre considerare la massima delle distanze, ossia

$$d^2_{(uv),z} = \frac{1}{2}\, d^2_{u,z} + \frac{1}{2}\, d^2_{v,z} + \frac{1}{2}\, |d^2_{u,z} - d^2_{v,z}|$$

e quindi confrontando tale espressione con la (7.15) si ha:

$$\alpha_u = \frac{1}{2}, \quad \alpha_v = \frac{1}{2}, \quad \beta = 0, \quad \gamma = \frac{1}{2}.$$

- *Metodo del legame medio* Dalla (7.7) segue che

$$d^2_{(uv),z} = \frac{N_u}{N_u + N_v}\, d^2_{u,z} + \frac{N_v}{N_u + N_v}\, d^2_{v,z}$$

dove N_u, N_v e N_z sono rispettivamente il numero di individui del cluster G_u, del cluster G_v e del cluster G_z. Confrontando tale espressione con la (7.15) si ha:

$$\alpha_u = \frac{N_u}{N_u + N_v}, \quad \alpha_v = \frac{N_u}{N_u + N_v}, \quad \beta = 0, \quad \gamma = 0.$$

- *Metodo del centroide* Dalla (7.11) segue che

$$d^2_{(uv),z} = \frac{N_u}{N_u + N_v}\, d^2_{u,z} + \frac{N_v}{N_u + N_v}\, d^2_{v,z} - \frac{N_u\, N_v}{(N_u + N_v)^2}\, d^2_{u,v}$$

e quindi confrontando tale espressione con la (7.15) si ha:

$$\alpha_u = \frac{N_u}{N_u + N_v}, \quad \alpha_v = \frac{N_v}{N_u + N_v}, \quad \beta = -\frac{N_u N_v}{(N_u + N_v)^2}, \quad \gamma = 0.$$

- *Metodo della mediana* Dalla (7.13) segue che

$$d^2_{(uv),z} = \frac{1}{2}\, d^2_{u,z} + \frac{1}{2}\, d^2_{v,z} - \frac{1}{4}\, d^2_{u,v}$$

e quindi confrontando tale espressione con la (7.15) si ha

$$\alpha_u = \frac{1}{2}, \quad \alpha_v = \frac{1}{2}, \quad \beta = -\frac{1}{4}, \quad \gamma = 0.$$

Osservazioni conclusive La scelta del metodo gerarchico agglomerativo dipende dagli scopi che il ricercatore si propone poiché ogni metodo definisce un diverso concetto di omogeneità all'interno dei cluster. Non esiste un metodo migliore, ma *ogni metodo ha i suoi vantaggi e i suoi svantaggi*. Se non si ha nessuna informazione sulla struttura dell'insieme da investigare e soprattutto se non si conosce la forma dei cluster da individuare, è sempre interessante applicare il metodo del legame singolo e il metodo del legame completo. Occorre sottolineare che il metodo del legame singolo è in grado di individuare cluster di qualsiasi forma ma può dare origine alla formazione di una catena. Con il metodo del legame completo i cluster sono sicuramente ben separati ma l'algoritmo privilegia l'omogeneità tra gli elementi interni ai vari gruppi.

Le tecniche di tipo gerarchico sono sicuramente appropriate per dati numerici di tipo biologico o zoologico per i quali si può ragionevolmente assumere che esista una struttura gerarchica. Tali tecniche comunque trovano applicazione anche in numerosi altri campi scientifici ed hanno il notevole vantaggio rispetto alle tecniche di enumerazione completa di richiedere minore tempo computazionale, permettendo così il loro utilizzo anche in presenza di un numero considerevole di dati numerici.

Occorre infine sottolineare che *i metodi gerarchici hanno due vantaggi*:

- fornire una visione completa dell'insieme in termini di distanze, seppure condizionata dalla scelta del metodo scelto;

- non comportare la scelta a priori del numero di cluster oppure la scelta a priori dei parametri per la determinazione automatica del loro numero.

Terminato un qualsiasi algoritmo gerarchico si possono selezionare il numero di cluster che il ricercatore ritiene più adeguato al problema oggetto di studio.

7.3 Analisi del dendrogramma

Ci proponiamo ora di analizzare il dendrogramma ottenuto con un particolare metodo gerarchico e di calcolare, fissato il numero di cluster, le misure di non omogeneità della partizione individuata.

7.3.1 Disegnare rettangoli che evidenziano i cluster

Consideriamo un particolare dendrogramma ottenuto a partire dalla funzione `hclust`. La funzione `rect.hclust()` permette di *disegnare dei rettangoli intorno ai cluster*, individuati in base all'altezza `h` alla quale si opera il taglio del dendrogramma oppure in base al numero `k` di cluster che si vogliono ottenere attraverso la funzione

```
> rect.hclust(z, h = NULL, k = NULL, border = "color")
```

dove

- `z` è l'oggetto creato (output) dalla funzione `hclust`;

- h è l'altezza alla quale si inserisce il taglio;

- k è il numero di cluster che si vogliono ottenere;

- border è il colore dei contorni dei rettangoli.

Esempio 7.6 Consideriamo due caratteristiche C_1 e C_2, con le stesse unità di misura, osservate per otto differenti individui $I_1, I_2, I_3, I_4, I_5, I_6, I_7, I_8$.

$$X = \begin{array}{c} \\ I_1 \\ I_2 \\ I_3 \\ I_4 \\ I_5 \\ I_6 \\ I_7 \\ I_8 \end{array} \begin{array}{c} \begin{array}{cc} C_1 & C_2 \end{array} \\ \begin{pmatrix} 0 & 0 \\ 1 & 1 \\ 2 & 2 \\ 4 & 3 \\ 5 & 3 \\ 4 & 4 \\ 6 & 5 \\ 7 & 5 \end{pmatrix} \end{array}$$

Utilizzando R definiamo il data frame dei dati:

```
> X<-data.frame(c1=c(0,1,2,4,5,4,6,7),c2=c(0,1,2,3,3,4,5,5))
> row.names(X)<-c("I1","I2","I3","I4","I5","I6","I7","I8")
> X # visualizza il data frame X
   c1 c2
I1  0  0
I2  1  1
I3  2  2
I4  4  3
I5  5  3
I6  4  4
I7  6  5
I8  7  5
```

Possiamo rappresentare gli otto punti relativi agli individui $I_1, I_2, \ldots, I_7, I_8$ in uno scatterplot. Le seguenti linee di codice

```
> plot(X$c1,X$c2,col="red",xlab="C1",
+ ylab="C2",ylim=c(0,5.5))
> text(X$c1,X$c2+0.2,c("I1","I2","I3","I4","I5","I6","I7","I8"))
>
> abline(lm(X$c2~X$c1),lty=2,col="blue")
```

producono lo scatterplot di Figura 7.12 in cui è anche indicata la retta di regressione. Calcoliamo ora la matrice contenente i quadrati delle distanze euclidee e applichiamo il metodo gerarchico del centroide.

```
> d<-dist(X,method="euclidean",diag=TRUE,upper=TRUE)
> d2<-d^2
> d2 #visualizza la matrice con i quadrati delle distanze euclidee
   I1 I2 I3 I4 I5 I6 I7 I8
I1  0  2  8 25 34 32 61 74
I2  2  0  2 13 20 18 41 52
I3  8  2  0  5 10  8 25 34
```

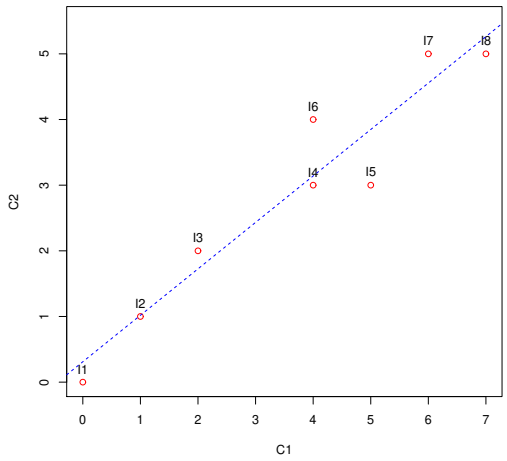

Figura 7.12: Scatterplot associato agli otto individui.

```
I4 25 13  5  0  1  1  8 13
I5 34 20 10  1  0  2  5  8
I6 32 18  8  1  2  0  5 10
I7 61 41 25  8  5  5  0  1
I8 74 52 34 13  8 10  1  0
>
> tree <- hclust(d2, method = "centroid")
>
> str(tree) # visualizza la struttura dell'oggetto tree
List of 7
 $ merge      : int [1:7, 1:2] -4 -7 -6 -1 -3 2 5 -5 -8 1 ...
 $ height     : num [1:7] 1 1 1.25 2 4.5 ...
 $ order      : int [1:8] 3 1 2 7 8 6 4 5
 $ labels     : chr [1:8] "I1" "I2" "I3" "I4" ...
 $ method     : chr "centroid"
 $ call       : language hclust(d = d2, method = "centroid")
 $ dist.method: chr "euclidean"
 - attr(*, "class")= chr "hclust"
```

Utilizzando i comandi tree$merge, tree$height e tree$order è possibile visualizzare l'output ottenuto con il metodo del centroide; i risultati sono illustrati nella Tabella 7.9.

Costruiamo ora il dendrogramma utilizzando R. Le seguenti linee di codice

```
> plot(tree,hang=-1,xlab="Metodo gerarchico agglomerativo",
+ sub="del centroide")
> axis(side=4,at=round(c(0,tree$height),2))
```

producono il grafico di Figura 7.13.

Tabella 7.9: Analisi dell'output del metodo gerarchico del centroide

		Agglomerazione	Distanza
-4	-5	Al livello 1 si uniscono gli individui I_4 e I_5 (primo cluster)	1.000000
-7	-8	Al livello 2 si uniscono gli individui I_7 e I_8 (secondo cluster)	1.000000
-6	1	Al livello 3 si uniscono l'individuo I_6 con il primo cluster (formato dagli individui I_4 e I_5) costituendo il terzo cluster	1.250000
-1	-2	Al livello 4 si uniscono gli individui I_1 e I_2 (quarto cluster)	2.000000
-3	4	Al livello 5 si unisce l'individuo I_3 con il quarto cluster (formato dagli individui I_1 e I_2) costituendo il quinto cluster	4.500000
2	3	Al livello 6 si unisce il secondo cluster (formato dagli individui I_7 e I_8) con il terzo cluster (formato dagli individui I_4,I_5 e I_6) costituendo il sesto cluster	7.472222
5	6	Al livello 7 si unisce il quinto cluster (formato dagli individui I_1, I_2 e I_3) con il sesto cluster (formato dagli individui I_4,I_5, I_6, I_7 e I_8)	26.640000

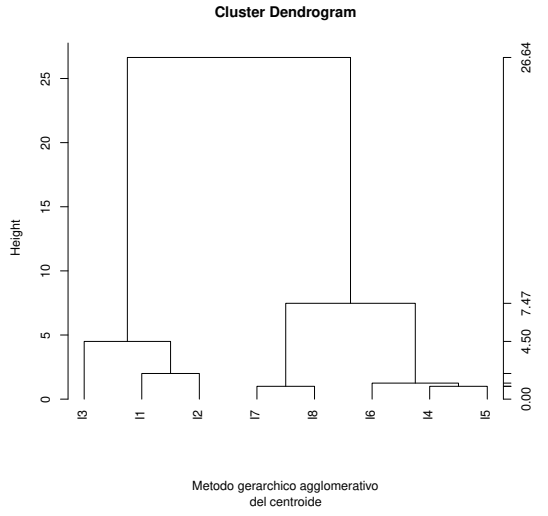

Figura 7.13: Dendrogramma ottenuto con il metodo del centroide.

La sequenza delle agglomerazioni del metodo del centroide è pertanto rappresentata nella Tabella 7.10 in cui sono anche indicati i corrispondenti livelli di distanza.

Tabella 7.10: Metodo gerarchico del centroide

Numero di cluster	Cluster	Livello di distanza
8	$\{I_1\},\{I_2\},\{I_3\},\{I_4\},\{I_5\},\{I_6\},\{I_7\},\{I_8\}$	
7	$\{I_1\},\{I_2\},\{I_3\},\{I_4,I_5\},\{I_6\},\{I_7\},\{I_8\}$	1.000000
6	$\{I_1\},\{I_2\},\{I_3\},\{I_4,I_5\},\{I_6\},\{I_7,I_8\}$	1.000000
5	$\{I_1\},\{I_2\},\{I_3\},\{I_4,I_5,I_6\},\{I_7,I_8\}$	1.250000
4	$\{I_1,I_2\},\{I_3\},\{I_4,I_5,I_6\},\{I_7,I_8\}$	2.000000
3	$\{I_1,I_{2,3}\},\{I_4,I_5,I_6\},\{I_7,I_8\}$	4.500000
2	$\{I_1,I_{2,3}\},\{I_4,I_5,I_6,I_7,I_8\}$	7.472222
1	$\{I_1,I_{2,3},I_4,I_5,I_6,I_7,I_8\}$	26.640000

Supponiamo di voler evidenziare due partizioni mediante rettangoli colorati in rosso. La seguente linea di codice

```
> rect.hclust(tree, k = 2, border = "red")
```

producono il grafico di Figura 7.14.

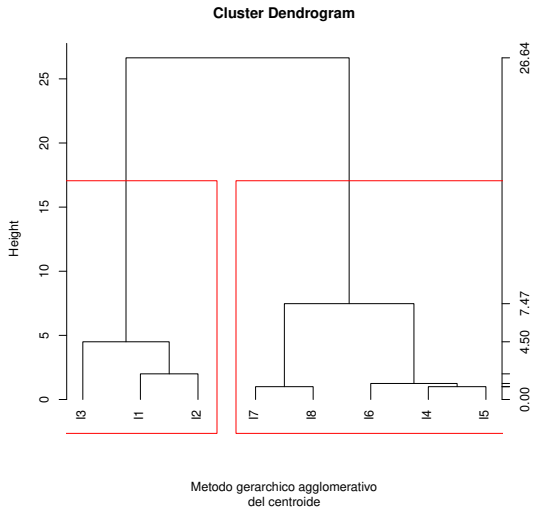

Figura 7.14: Rettangoli che evidenziano due partizioni.

Supponiamo invece di voler evidenziare tre partizioni mediante rettangoli colorati in verde. La seguente linea di codice

```
> rect.hclust(tree, k = 3, border = "green")
```

producono il grafico di Figura 7.15.

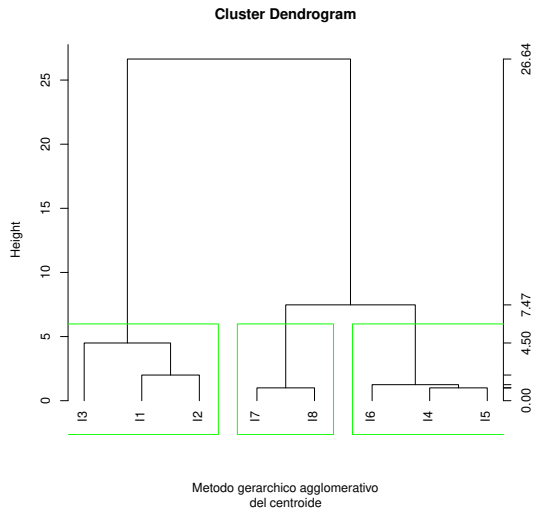

Figura 7.15: Rettangoli che evidenziano tre partizioni.

Per confrontare differenti partizioni alternative si può utilizzare più volte sullo stesso grafico la funzione rect.hclust(). Ad esempio, le seguenti linee di codice

```
> rect.hclust(tree, k = 2, border = "red")
> rect.hclust(tree, k = 3, border = "green")
```

producono il grafico di Figura 7.16 che permette di confrontare su uno stesso grafico le partizioni di Figura 7.14 e di Figura 7.15.

7.3.2 Inserire gli individui nei cluster

Considerato un particolare dendrogramma, per ottenere una *suddivisione degli individui in cluster* in corrispondenza di un determinato livello di distanza oppure in corrispondenza di un prefissato numero di cluster, R utilizza anche la funzione cutree() nel seguente modo:

> cutree(tree, k = NULL, h = NULL)

dove:

- tree rappresenta un oggetto (che individua un dendrogramma) creato tramite la funzione hclust();

- k è il numero prefissato di cluster;

- h è l'altezza alla quale il dendrogramma viene tagliato.

L'output della funzione cutree() è un vettore contenente numeri interi positivi associati ai cluster in cui sono stati inseriti i vari individui. Inoltre, per vedere

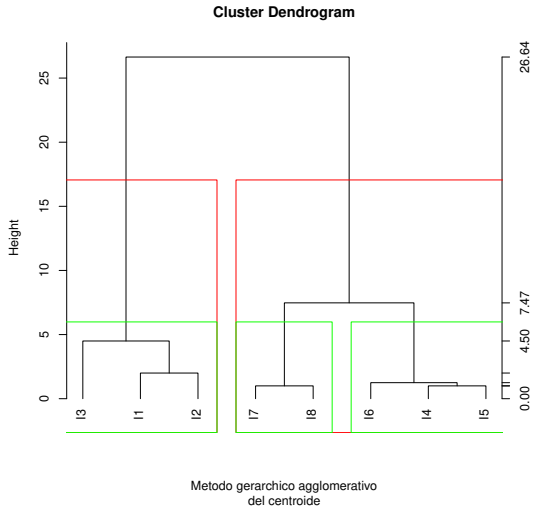

Figura 7.16: Rettangoli che evidenziano due partizioni (in rosso) e tre partizioni (in verde).

come vengono classificati gli individui al'aumentare del numero di cluster si può considerare la funzione

`> cutree(tree, k = 1:n)`

dove n indica il numero di individui. L'output di tale funzione `cutree()` è una matrice in cui la colonna k-esima contiene numeri interi positivi associati ai cluster in cui sono stati inseriti i vari individui.

Esempio 7.7 Riconsideriamo l'Esempio 7.6. Se utilizziamo la funzione `cutree()` fissando a due il numero di cluster in cui tagliare il dendrogramma si ha:

```
> cutree(tree, k = 2, h = NULL)
I1 I2 I3 I4 I5 I6 I7 I8
 1  1  1  2  2  2  2  2
```

che individua la seguente partizione in due cluster $G_1 = \{I_1, I_2, I_3\}$ e $G_2 = \{I_4, I_5, I_6, I_7, I_8\}$. Se invece utilizziamo la funzione `cutree()` fissando un livello di distanza pari a 15 in cui tagliare il dendrogramma si ottiene:

```
> cutree(tree, k = NULL, h =15)
I1 I2 I3 I4 I5 I6 I7 I8
 1  1  1  2  2  2  2  2
```

che individua di nuovo la partizione in due cluster $G_1 = \{I_1, I_2, I_3\}$ e $G_2 = \{I_4, I_5, I_6, I_7, I_8\}$.

Per ottenere il numero di unità (individui) in ciascun cluster si può applicare la funzione `table()` al risultato della funzione `cutree()` ottenendo:

```
> table(cutree(tree, k =2, h = NULL))
1 2
3 5
```

che mostra che nel primo cluster sono presenti tre individui e nel secondo cluster cinque individui. Infine, la funzione

```
> cutree(tree, k = 1:8)
   1 2 3 4 5 6 7 8
I1 1 1 1 1 1 1 1 1
I2 1 1 1 1 2 2 2 2
I3 1 1 1 2 3 3 3 3
I4 1 2 2 3 4 4 4 4
I5 1 2 2 3 4 4 4 5
I6 1 2 2 3 4 5 5 6
I7 1 2 3 4 5 6 6 7
I8 1 2 3 4 5 6 7 8
```

permette di classificati gli individui al'aumentare del numero di cluster. Ad esempio, la partizione in quattro cluster è $G_1 = \{I_1, I_2\}$ e $G_2 = \{I_3\}$, $G_3 = \{I_4, I_5, I_6\}$ e $G_3 = \{I_7, I_8\}$.

7.3.3 Misure di sintesi associate ai cluster

In R è inoltre possibile ricavare misure di sintesi (ad esempio, la media campionaria, la varianza campionaria, la deviazione standard,...) sulle colonne dei singoli cluster, ottenuti tagliando il dendrogramma tramite la funzione `cutree()`, utilizzando la funzione `aggregate()` nel seguente modo:

`> aggregate(X, by, FUN)`

dove:

- `X` rappresenta una matrice numerica o un data frame;
- `by` è una *lista* di indici sulla base dei quali le colonne di `X` vanno aggregate;
- `FUN` è la funzione da applicare alle colonne di `X`, separatamente per i vari gruppi individuati in base a `by`.

L'output della funzione `aggregate()` è una struttura contenente i valori ottenuti applicando la funzione `FUN` (ad esempio, la media campionaria, la varianza campionaria, la deviazione standard,...) ad ognuna delle caratteristiche associate ai diversi cluster che sono stati aggregati.

Esempio 7.8 Riconsideriamo l'Esempio 7.6 con matrice dei dati:

```
> X<-data.frame(c1=c(0,1,2,4,5,4,6,7),c2=c(0,1,2,3,3,4,5,5))
> row.names(X)<-c("I1","I2","I3","I4","I5","I6","I7","I8")
> X # visualizza il data frame X
   c1 c2
I1  0  0
I2  1  1
I3  2  2
```

```
I4  4  3
I5  5  3
I6  4  4
I7  6  5
I8  7  5
```

Desideriamo utilizzare la funzione aggregate() per calcolare le medie campionarie, le varianze campionarie e le deviazioni standard delle caratteristiche dei due cluster precedentemente individuati $G_1 = \{I_1, I_2, I_3\}$ e $G_2 = \{I_4, I_5, I_6, I_7, I_8\}$:

```
> taglio<-cutree(tree, k =2, h = NULL)
> tagliolist<-list(taglio) # lista di indici  per  i gruppi
>
> aggregate(X, tagliolist, mean)
  Group.1  c1 c2
1       1 1.0  1
2       2 5.2  4
>
> aggregate(X, tagliolist, var)
  Group.1  c1 c2
1       1 1.0  1
2       2 1.7  1
>
> aggregate(X, tagliolist, sd)
  Group.1      c1 c2
1       1 1.00000  1
2       2 1.30384  1
```

che mostra che il centroide di G_1 ha coordinate $P_1 = (1.0, 1.0)$ e il centroide di G_2 è $P_2 = (5.2, 4)$.

Occorre ricordare che *per il calcolo della varianza campionaria e della deviazione standard campionaria occorre che nel cluster siano presenti almeno due unità (individui).*

Se esistono due sole caratteristiche nella matrice dei dati, per rappresentare graficamente i due cluster ottenuti con la funzione cutree() ed anche i centroidi dei due cluster, si utilizzano le seguenti linee di codice:

```
> agmean<-aggregate(X, tagliolist, mean)[, -1]
>
> plot(X, col = taglio, main = "Metodo gerarchico del centroide")
> points(agmean,col = 1:2,pch=8,cex=1)
```

dove X è la matrice delle misure e taglio è l'oggetto creato con la funzione cutree(), col() individua i colori da associare ai differenti cluster, pch controlla il tipo di carattere da utilizzare e cex la grandezza del testo e dei simboli generati. Per motivi computazionali bisogna eliminare la prima colonna della matrice dei centroidi ottenuta con la funzione aggregate() che si riferisce alle etichette dei cluster. Si ottiene la Figura 7.17 dove sull'asse orizzontale sono riportati i valori della caratteristica C_1 e sull'asse verticale quelli della caratteristica C_2. Gli individui appartenenti al cluster G_1 sono evidenziati con pallini neri mentre quelli del cluster G_2 con pallini rossi. Sono anche evidenziati il centroide $P_1 = (1, 1)$ del cluster G_1 e il centroide $P_2 = (5.2, 4)$ del cluster G_2.

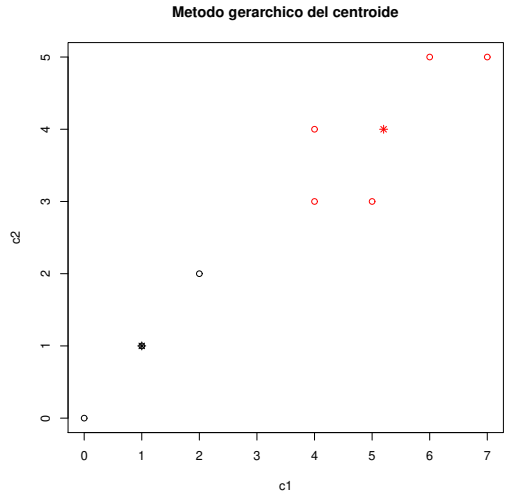

Figura 7.17: Partizione in due cluster ottenuta con il metodo del centroide.

Se le caratteristiche sono più di due si può invece utilizzare la funzione scatterplot() al posto della funzione plot().

7.3.4 Misure di non omogeneità statistiche

Dopo aver effettuato il taglio, siamo interessati a calcolare le misure di non omogeneità statistica relative all'insieme totale di individui (tr T), ai singoli cluster ottenuti effettuando il taglio e alla somma delle loro misure di non omogeneità (tr S) e alla misura di omogeneità tra i cluster (tr B):

$$\text{tr } T = \text{tr } S + \text{tr } B,$$

o equivalentemente:

$$1 = \frac{\text{tr } S}{\text{tr } T} + \frac{\text{tr } B}{\text{tr } T}.$$

Poiché per ogni fissata matrice X dei dati si ha che la tr T è fissata, i cluster dovrebbero essere individuati in modo da *minimizzare la misura di non omogeneità statistica all'interno dei cluster (within) e massimizzare la misura di non omogeneità statistica tra i gruppi (between).*

Se, fissato il numero di cluster, due differenti metodi gerarchici conducono a due diverse partizioni, occorre scegliere quella partizione con misura di non omogeneità statistica all'interno dei cluster (tr S) più piccola, che corrisponde a maggiore omogeneità interna.

Esempio 7.9 Riconsideriamo l'Esempio 7.6. Calcoliamo ora le misure di non omogeneità statistica relative all'insieme totale $I = G_1 \cup G_2$ e ai due cluster

$G_1 = \{I_1, I_2, I_3\}$ e $G_2 = \{I_4, I_5, I_6, I_7, I_8\}$ individuati con l'analisi gerarchica utilizzando il metodo del centroide. Per l'insieme totale I si ha:

```
> n<-nrow(X)
# n>1
> trHI<-(n-1)*sum(apply(X,2,var))
> trHI# visualizza la misura di non omogeneita' totale
[1] 64.75
```

La misura di non omogenità statistica totale è quindi $\mathrm{tr}\, H_I = 64.75$. Calcoliamo ora le misure di non omogeneità statistiche dei due gruppi G_1 e G_2 seguendo due differenti metodi.

Metodo 1

Nel primo metodo occorre definire le matrici dei dati relative ai due gruppi e a partire da esse determinare le misure di non omogeneità statistiche. Per il primo gruppo G_1 consideriamo le seguenti linee di codice:

```
> X1<-data.frame(c1=c(0,1,2),c2=c(0,1,2))
> rownames(X1)<-c("I1","I2","I3")
>  # n1>1
> n1<-nrow(X1)
> trH1<-(n1-1)*sum(apply(X1,2,var))
> trH1 # misura di non omogeneita' statistica del primo gruppo
[1] 4
```

La traccia della matrice di non omogeneità statistica del primo gruppo G_1 è quindi $\mathrm{tr}\, H_{G_1} = 4$. Per il secondo gruppo consideriamo le seguenti linee di codice:

```
> X2<-data.frame(c1=c(4,5,4,6,7),c2=c(3,3,4,5,5))
> rownames(X2)<-c("I4","I5","I6","I7","I8")
> # n2>1
> n2<-nrow(X2)
> trH2<-(n2-1)*sum(apply(X2,2,var))
> trH2 # misura di non omogeneita' statistica del secondo gruppo
[1] 10.8
```

La traccia della matrice di non omogeneità statistica del secondo gruppo G_2 è quindi $\mathrm{tr}\, H_{G_2} = 10.8$.

Occorre ricordare che se un cluster contiene un solo individuo (unità) la sua misura di non omogeneità statistica è nulla.

Metodo 2

Nel secondo metodo non occorre definire le matrici dei dati relative ai due gruppi e si possono ricavare le misure di non omogeneità statistica dei due gruppi utilizzando le funzioni cutree() e aggregate() come mostrato nelle seguenti linee di codice

```
> d<-dist(X,method="euclidean",diag=TRUE,upper=TRUE)
> d2<-d^2
> tree <- hclust(d2, method = "centroid")
>
> taglio<-cutree(tree, k =2, h = NULL)
> num<-table(taglio) #numero di elementi dei gruppi
> tagliolist<-list(taglio) #lista di indici per i gruppi
```

```
>
> agvar<- aggregate(X, tagliolist, var)[, -1]
> # n1>1
> trH1<-(num[[1]]-1) * sum(agvar[1, ])
> trH1 # visualizza la misura di non omogeneita' del primo gruppo
[1] 4
> # n2>1
> trH2<-(num[[2]] -1) *sum(agvar[2, ])
> trH2 # visualizza la misura di non omogeneita' del secondo gruppo
[1] 10.8
```

dove d2 è la matrice contenente i quadrati delle distanze euclidee e num è un vettore contenente il numero di individui nei due cluster. La misura di non omogeneità del primo gruppo è calcolata moltiplicando n_1-1 per la somma degli elementi della prima riga della matrice ottenuta con aggregate(X, tagliolist, var), ossia $2 \cdot (1+1) = 4$. Invece, la misura di non omogeneità del secondo gruppo è calcolata moltiplicando $n_2 - 1$ per la somma degli elementi della seconda riga della matrice ottenuta con aggregate(X, tagliolist, var), ossia $4 \cdot (1.7 + 1) = 10.8$.

Riassumendo la misura di non omogeneità statistica totale è 64.75, la misura di non omogeneità statistica all'interno dei due gruppi (*within*) è $\operatorname{tr} H_{G_1} + \operatorname{tr} H_{G_2} = 4 + 10.8 = 14.8$ e la misura di non omogeneità tra i cluster (*between*) è $\operatorname{tr} H(G_1 \cap G_2) = \operatorname{tr} H_I - \operatorname{tr} H_{G_1} - \operatorname{tr} H_{G_2} = 64.75 - 14.8 = 49.95$. In conclusione, la misura di non omogeneità all'interno dei gruppi (within) è quindi piccola rispetto la misura di non omogeneità tra i cluster (between). Inoltre,

$$\frac{\operatorname{tr} B}{\operatorname{tr} T} = \frac{49.95}{64.75} = 0.7714286.$$

7.4 Screeplot

Un metodo euristico per scegliere una buona partizione del dendrogramma considera una *procedura empirica* consistente nel costruire un grafico, detto *screeplot*. In esso si pongono sull'asse delle ordinate i numeri di gruppi ottenibili con il metodo gerarchico e sull'asse delle ascisse le distanze a cui avvengono le successive aggregazioni tra i gruppi. Se nel passaggio da k gruppi a $k-1$ gruppi si registra un forte incremento della distanza di aggregazione è consigliabile tagliare il dendrogramma in k gruppi. La *procedura empirica basata sullo screeplot non sempre fornisce la suddivisione in cluster più adeguata*; *è sempre preferibile utilizzare le misure di non omogeneità statistiche precedentemente descritte.* Inoltre, *è possibile realizzare lo screeplot a partire dal metodo del legame singolo, del legame completo o del legame medio* in cui è utilizzata la funzione distanza. Invece, nel metodo del centroide e della mediana (che utilizzano i quadrati delle distanze) le successive agglomerazioni potrebbero verificarsi ad un livello di distanza minore o uguale rispetto alle precedenti agglomerazioni. Ciò comporta che lo screeplot ottenuto a partire dal metodo del centroide o della mediana potrebbe non essere regolare e non fornire indicazioni adeguate.

Descriviamo comunque tale procedura empirica.

Allo scopo di fornire delle misure quantitative degli incrementi riscontrati tra le distanze a cui avvengono le successive agglomerazioni si possono considerare le quantità:

$$\delta_k = d_{k-1} - d_k \qquad (k = 2, \ldots, n), \tag{7.16}$$

dove d_k rappresenta il livello di distanza a cui è stata effettuata l'agglomerazione in k gruppi e n è il numero iniziale di individui. Quando l'incremento δ_k risulta sufficietemente elevato, significa che i gruppi sono sufficientemente dissimili tra loro, per cui è possibile tagliare il dendrogramma all'altezza (al livello di distanza) corrispondente alla partizione in k gruppi. Lo screeplot fornisce una visione di insieme delle altezze a cui sono avvenute le agglomerazioni e si potrebbe scegliere il valore di j per il quale $\delta_j = \max\{\delta_2, \delta_3, \ldots, \delta_n\}$.

Esempio 7.10 Consideriamo due caratteristiche C_1 e C_2, con le stesse unità di misura, osservate per otto differenti individui $I_1, I_2, I_3, I_4, I_5, I_6, I_7, I_8$ esaminati nell'Esempio 7.6. Utilizzando R definiamo la matrice delle misure:

```
> X<-data.frame(c1=c(0,1,2,4,5,4,6,7),c2=c(0,1,2,3,3,4,5,5))
> row.names(X)<-c("I1","I2","I3","I4","I5","I6","I7","I8")
> X # visualizza il data frame X
   c1 c2
I1  0  0
I2  1  1
I3  2  2
I4  4  3
I5  5  3
I6  4  4
I7  6  5
I8  7  5
```

Calcoliamo ora la matrice delle distanze euclidee e applichiamo il metodo gerarchico del legame completo.

```
> d<-dist(X,method="euclidean",diag=TRUE,upper=TRUE)
>
> hlc<-hclust(d,method="complete")
> hlc$height
[1] 1.000000 1.000000 1.414214 1.414214 2.828427 3.605551 8.602325
```

Costruiamo ora il dendrogramma utilizzando R. Le seguenti linee di codice

```
> plot(hlc,hang=-1,xlab="Metodo gerarchico agglomerativo",
+ sub="del legame completo")
> axis(side=4,at=round(c(0,hlc$height),2))
```

producono il grafico di Figura 7.18.

Vogliamo ora costruire lo screeplot. Consideriamo la seguente linea di codice

```
> plot(c(0,hlc$height),seq(8,1),type="b",
+ main="Screeplot",xlab="Distanza di aggregazione",
+ ylab="Numero di cluster", col="red")
```

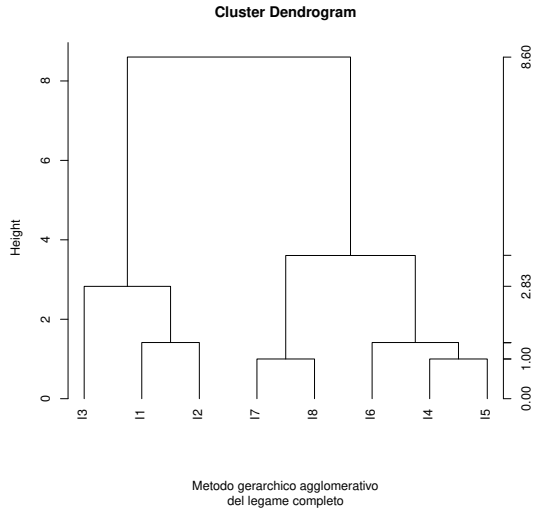

Figura 7.18: Dendrogramma ottenuto con il metodo del legame completo.

La funzione $\mathsf{c}(0, \mathsf{hlc\$height})$ permette di concatenare 0 con il vettore hlc$height delle altezze a cui sono avvenute le successive agglomerazioni, ottenendo il seguente vettore di lunghezza otto:

$$0 \quad 1.000000 \quad 1.000000 \quad 1.414214 \quad 1.414214 \quad 2.828427 \quad 3.605551 \quad 8.602325,$$

che corrispondono alle aggregazioni in 8 gruppi, in 7 gruppi, ..., fino ad arrivare ad un unico gruppo. La funzione $\mathsf{seq}(8, 1)$ permette di costruire il vettore contenente il numero di gruppi da 8 a 1. Infine la funzione $\mathsf{type} = \text{"b"}$ permette di connettere con delle linee i vari punti. La precedente linea di codice produce il grafico illustrato in Figura 7.19.

Lo screeplot in Figura 7.19 suggerisce di considerare una suddivisione in due gruppi. Infatti, nel passaggio da uno (altezza 8.602325) a due gruppi (altezza 3.605551) si registra un consistente incremento della distanza di aggregazione. Inoltre, si ha:

$$\begin{aligned}
\delta_2 &= h_1 - h_2 = 8.602325 - 3.605551 = 4.996774 \\
\delta_3 &= h_2 - h_3 = 3.605551 - 2.828427 = 0.777124 \\
\delta_4 &= h_3 - h_4 = 2.828427 - 1.414214 = 1.414213 \\
\delta_5 &= h_4 - h_5 = 1.414214 - 1.414214 = 0 \\
\delta_6 &= h_5 - h_6 = 1.414214 - 1.000000 = 0.414214 \\
\delta_7 &= h_6 - h_7 = 1.000000 - 1.000000 = 0 \\
\delta_8 &= h_7 - h_8 = 1.000000 - 0.000000 = 1.
\end{aligned}$$

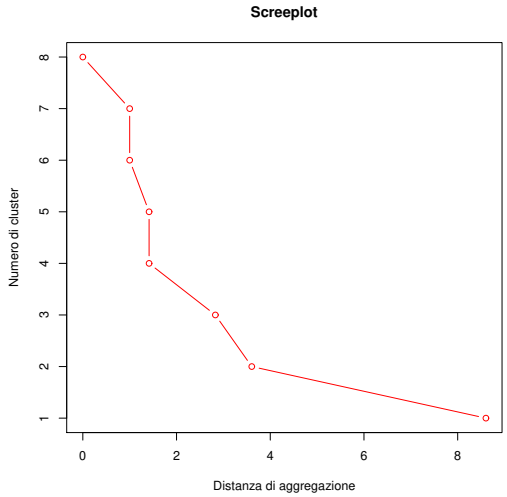

Figura 7.19: Sull'asse delle ordinate è presente il numero di gruppi e sull'asse delle ascisse la distanza in cui è avvenuta l'aggregazione tra i gruppi.

che mostra che il valore di k per il quale δ_k è massima è $k = 2$. È preferibile quindi considerare una suddivisione nei due cluster $G_1 = \{I_1, I_2, I_3\}$ e $G_2 = \{I_4, I_5, I_6, I_7, I_8\}$.

Occorre infine sottolineare che lo screeplot è un metodo empirico che realizza un grafico basato sulle altezze a cui sono avvenute le aggregazioni in un metodo gerarchico e quindi *non sempre fornisce il numero adeguato di cluster in cui suddividere gli individui.* Inoltre, è preferibile non utilizzare lo screeplot nel metodo del centroide e della mediana. Per la suddivisione in cluster in un metodo gerarchico è sempre preferibile ricorrere alle misure di non omogeneità statistiche di cui si serviranno, come vedremo, anche i metodi non gerarchici.

7.5 Metodi non gerarchici

L'obiettivo dei metodi non gerarchici è quello di ottenere un'unica partizione degli n individui di partenza in cluster. A differenza dei metodi gerarchici, in tali tecniche è consentito riallocare gli individui già classificati ad un livello precedente dell'analisi.

In letteratura esistono moltissime tecniche non gerarchiche ed è quindi impossibile ricondurre tali metodi ad un unico tipo, come invece avviene per molti metodi gerarchici di tipo agglomerativo. In molti metodi non gerarchici di clustering si assume che il numero di cluster in cui suddividere l'insieme totale degli n individui sia fissato a priori dal ricercatore, mentre in altri tale numero è determinato nel corso dell'analisi. Inoltre, molte di queste tecniche richiedono inizialmente la determinazione di un insieme iniziale di punti di riferimento

(ad esempio un insieme iniziale di punti attorno ai quali si addensano i cluster) oppure l'individuazione di una partizione iniziale dei n individui in cluster.

Gli algoritmi di tipo non gerarchico procedono, data una prima partizione, a riallocare gli individui nel gruppo con centroide più vicino, fino a che per nessun individuo si verifica che sia minima la distanza rispetto al centroide di un gruppo diverso da quello a cui esso appartiene.

Il metodo più utilizzato prende il nome di *k-means* ed è dovuto a Hartigan e Wong[1]. Tale metodo richiede che il numero di cluster sia specificato a priori e fornisce in output un'unica partizione. Esso consiste dei passi descritti nel seguente algoritmo:

◇ **Algoritmo**

- **Step 1:** Fissare a priori il numero k di cluster specificando i punti di riferimento iniziali (scegliendo in maniera opportuna alcuni individui, o unità, o prendendo la configurazione determinata con una tecnica gerarchica) che inducono una prima partizione provvisoria;
- **Step 2:** Considerare tutti gli individui e attribuire ciascuno di essi al cluster individuato dal punto di riferimento da cui ha distanza minore;
- **Step 3:** Calcolare il baricentro (il centroide) di ognuno dei k gruppi così ottenuti. Tali centroidi costituiscono i punti di riferimento per i nuovi cluster;
- **Step 4:** Valutare la distanza di ogni unità da ogni centroide ottenuto al passo precedente. Se la distanza minima non è ottenuta in corrispondenza del centroide del gruppo di appartenenza, allora si procede a spostare l'individuo presso il cluster che ha il centroide più vicino.
- **Step 5:** Ricalcolare i centroidi dei k gruppi così ottenuti.
- **Step 6:** Ripetere il procedimento a partire dal punto (4) fino a che i centroidi non subiscono ulteriori modifiche rispetto all'iterazione precedente. Si procede così iterativamente a spostamenti successivi fino a raggiungere una configurazione stabile, ossia gli individui all'interno di ogni cluster non cambiano al ripetersi del procedimento.

Nel metodo k-means, per garantire la convergenza della procedura iterativa, come misura di distanza tra i vettori delle caratteristiche e i centroidi viene utilizzata la *distanza euclidea* e, come per il metodo del centroide, si considera la matrice contenente i *quadrati delle distanze euclidee*.

I vantaggi del metodo k-means sono la velocità di esecuzione dei calcoli e l'estrema libertà che viene lasciata agli individui di raggrupparsi e allontanarsi. Uno svantaggio è invece che la classificazione finale può essere influenzata dalla scelta iniziale dei k vettori delle caratteristiche come punti di riferimento, dall'ordine in cui sono presi tali vettori e naturalmente dalle proprietà geometriche

[1] J. A. Hartigan, M. A. Wong: A K-Means Clustering Algorithm. Journal of the Royal Statistical Society. Series C (Applied Statistics), 28 (1), 100–108 (1979)

dei vettori delle misure. Infatti, l'algoritmo potrebbe convergere ad un ottimo locale e non globale, il che significa che se si inizia con un diverso insieme di punti di riferimento si può giungere ad una differente partizione finale.

L'analisi con il metodo k-means si effettua in R mediante la funzione

```
> kmeans(X, centers, iter.max = N, nstart = M)
```

dove

- `X` è la matrice dei dati;

- `centers` è il numero dei cluster che si vogliono identificare o un vettore di lunghezza pari al numero di cluster contenente un insieme di centroidi iniziali dei cluster. Nel primo caso, ossia se è numero intero, l'algoritmo sceglie casualmente i punti di riferimento e tale insieme è utilizzato per individuare la partizione iniziale. Nel secondo caso, i centroidi iniziali possono essere derivati effettuando preliminarmente un'analisi di tipo gerarchico con il metodo del centroide.

- `iter.max` è il massimo numero di iterazioni permesse. Di default `iter.max` = 10.

- `nstart` fornisce il numero di volte in cui ripetere la procedura di scelta casuale dei punti di riferimento, nel caso in cui `centers` è il numero. Di default `nstart` = 1. Se `nstart` > 1, l'algoritmo fornisce sempre come risultato la partizione con una misura di non omogeneità statistica totale all'interno dei cluster minima.

Si nota che nell'algoritmo k-means *non occorre calcolare la matrice iniziale delle distanze* (o dei quadrati delle distanze) così come invece si richiede nei metodi gerarchici.

La funzione `kmeans()` produce come output una lista, i cui elementi sono:

- un vettore di interi che indica il cluster di allocazione di ogni individuo (`$cluster`)

- una matrice che contiene i centroidi dei cluster (`$center`)

- un vettore contenente le misure di non omogeneità statistica calcolate all'interno di ognuno dei cluster; tali valori dipendono dall'omogeneità interna e dalla numerosità del gruppo (`$withinss`)

- dimensione dei gruppi (`$size`)

La misura di non omogeneità statistica complessiva all'interno dei vari cluster (within) è quindi la somma delle misure di non omogeneità statistica di ognuno dei cluster.

Esempio 7.11 Riconsideriamo la matrice delle misure `X` fornita nell'Esempio 7.6 riguardante due caratteristiche C_1 e C_2 osservate per otto differenti individui $I_1, I_2, I_3, I_4, I_5, I_6, I_7, I_8$. In base alla precedente analisi di tipo gerarchico si è giunti alla conclusione che un numero ragionevole di cluster è 2. Si

può, quindi, sfruttare questa informazione per la scelta del numero di cluster da considerare all'inizio dell'analisi con il metodo k-means.

Consideriamo tre differenti scelte iniziali:

(i) scelta casuale dei punti di riferimento;

(ii) ripetizione della procedura di scelta casuale dei punti di riferimento;

(iii) scelta dei centroidi come punti di riferimento.

(i) **Scelta casuale dei punti di riferimento**
Applichiamo ai dati dell'Esempio 7.6 il metodo non gerarchico k-means considerando una suddivisione in due cluster ed effettuando un'unica *scelta casuale dei punti di riferimento* con un numero massimo di iterazioni pari a 10.

```
> X<-data.frame(c1=c(0,1,2,4,5,4,6,7),c2=c(0,1,2,3,3,4,5,5))
> row.names(X)<-c("I1","I2","I3","I4","I5","I6","I7","I8")
>
> km<-kmeans(X,center=2,iter.max=10,nstart=1)
> km # visualizza i risultati ottenuti con kmeans
K-means clustering with 2 clusters of sizes 3, 5

Cluster means:
   c1 c2
1 1.0  1
2 5.2  4

Clustering vector:
I1 I2 I3 I4 I5 I6 I7 I8
 1  1  1  2  2  2  2  2

Within cluster sum of squares by cluster:
[1]  4.0 10.8
 (between_SS / total_SS =  77.1 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "
    tot.withinss" "betweenss"    "size"
[8] "iter"         "ifault"
```

Il metodo k-means individua la seguente partizione in due cluster $G_1 = \{I_1, I_2, I_3\}$ e $G_2 = \{I_4, I_5, I_6, I_7, I_8\}$. Il primo cluster ha centrode di coordinate $(1, 1)$ mentre il secondo cluster ha centrode di coordinate $(5.2, 4)$. Come mostrato nell'Esempio 7.8 i valori 4.0 e 10.8 appena ottenuti sono le misure di non omogeneità statistica associate ai cluster $G_1 = \{I_1, I_2, I_3\}$ e $G_2 = \{I_4, I_5, I_6.I_7, I_8\}$, mentre la misura di non omogeneità totale che avevamo precedentemente calcolato è uguale a 64.75. La misura di non omogeneità tra i cluster è quindi $64.75 - 14.8 = 49.95$. Si nota inoltre che

$$\frac{\text{tr } B}{\text{tr } T} = \frac{49.95}{64.75} = 0.7714, \qquad \frac{\text{tr } S}{\text{tr } T} = 1 - \frac{49.95}{64.75} = 0.2286.$$

La funzione str(km) permette di ottenere una lista di utili informazioni relative all'oggetto km creato con il metodo kmeans() ottenibili considerando:

km$cluster, km$centers, km$totss, km$withinss, km$tot.withinss, km$betweenss e km$size.

In particolare, per i dati dell'Esempio 7.6 si ha:

```
> km$cluster
I1 I2 I3 I4 I5 I6 I7 I8
 1  1  1  2  2  2  2  2
>
> km$centers
   c1 c2
1 1.0  1
2 5.2  4
>
> km$totss
[1] 64.75
>
> km$withinss
[1]  4.0 10.8
>
> km$tot.withinss
[1] 14.8
>
> km$betweenss
[1] 49.95
>
> km$size
[1] 3 5
```

Si nota che la misura di non omogeneità totale è 64.75, la somma delle misure di non omogeneità interne ai cluster è 14.8 (within) e la misura di non omogeneità tra i cluster è 49.95 (between).

Poiché la partizione iniziale è scelta casualmente, non è detto che la procedura del k-means conduca sempre allo stesso risultato, soprattutto nel caso in cui il numero n di individui è piccolo. Per questa ragione è consigliabile scegliere casualmente diversi insiemi di punti di riferimento ponendo nstart > 1.

(ii) **Ripetizione della procedura di scelta casuale dei punti di riferimento**

Applichiamo ai dati dell'Esempio 7.6 il metodo non gerarchico k-means richiedendo che l'algoritmo di aggregazione venga ripetuto otto volte in corrispondenza di otto ripetizioni della procedura di *scelta casuale dei punti di riferimento* con un numero massimo di iterazioni pari a 10.

```
> kmeans(X, centers=2,iter.max=10, nstart=8)
```

In questo caso il metodo k-means individua la stessa partizione in due cluster $G_1 = \{I_1, I_2, I_3\}$ e $G_2 = \{I_4, I_5, I_6, I_7, I_8\}$ ottenendo le stesse misure di non omogeneità statistica precedenti. Potrebbe invece accadere che il risultato ottenuto con una ripetizione della procedura di scelta casuale dei punti di riferimento sia differente da quello ottenuto con un'unica scelta casuale dei punti di riferimento. In generale, l'output della funzione riporta la partizione per la quale la somma delle misure di non omogeneità statistica all'interno dei gruppi (within) è la

più piccola tra le partizioni finali ottenute a partire dalle procedure ricavate a partire diversi punti di riferimento scelti casualmente.

In generale non esiste una regola per determinare il numero ottimale di ripetizioni della procedura di scelta casuale dei punti di riferimento per ottenere un risultato stabile. Empiricamente, si potrebbe provare con diversi valori crescenti di nstart fino a che il risultato non cambia.

(iii) **Scelta dei centroidi come punti di riferimento**

Facendo riferimento all'Esempio 7.6 in cui abbiamo definito la matrice X dei dati, la matrice delle distanze euclidee e la matrice d2 contenente i quadrati delle distanze euclidee, in alternativa alla scelta casuale dei punti di riferimento, si possono impiegare i *centroidi dei due cluster ottenuti con tecnica gerarchica del centroide* utilizzando la funzione aggregate():

```
> d<-dist(X,method="euclidean",diag=TRUE,upper=TRUE)
> d2<-d^2
> tree <- hclust(d2, method = "centroid")
>
> taglio<-cutree(tree, k =2, h =NULL)
> tagliolist<-list(taglio)
> centroidiIniziali<-aggregate(X, tagliolist, mean)[,-1]
> centroidiIniziali # visualizza i centroidi iniziali
   c1 c2
1 1.0  1
2 5.2  4
```

Ricordiamo che per motivi computazionali bisogna eliminare la prima colonna della matrice dei centroidi ottenuta con la funzione aggregate() che si riferisce alle etichette dei cluster. Utilizzando tali centroidi possiamo ora applicare il metodo k-means:

```
> km<-kmeans(X, centers = centroidiIniziali, iter.max = 10)
>  km # visualizza i risultati ottenuti con kmeans
K-means clustering with 2 clusters of sizes 3, 5

Cluster means:
   c1 c2
1 1.0  1
2 5.2  4

Clustering vector:
I1 I2 I3 I4 I5 I6 I7 I8
 1  1  1  2  2  2  2  2

Within cluster sum of squares by cluster:
[1]  4.0 10.8
 (between_SS / total_SS =  77.1 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "
    tot.withinss" "betweenss"    "size"
[8] "iter"         "ifault"
```

Il metodo k-means individua di nuovo la partizione in due cluster $G_1 = \{I_1, I_2, I_3\}$ e $G_2 = \{I_4, I_5, I_6, I_7, I_8\}$. Occorre nuovamente sottolineare che non esiste una regola per determinare il numero massimo di iterazioni per ottenere un risultato stabile. Empiricamente, si potrebbe provare con diversi valori crescenti di iter.max fino a che il risultato non cambia.

Per rappresentare graficamente i cluster generati mediante il metodo k-means possiamo utilizzare i comandi:

```
> plot(X, col = km$cluster, main = "Metodo non gerarchico del k-
    means")
> points(km$center, col = 1:2, pch = 8, cex=1)
```

dove km è l'oggetto generato dalla funzione kmeans(), col() individua i colori da associare ai differenti cluster, pch controlla il tipo di carattere da utilizzare e cex la grandezza del testo e dei simboli generati. Il grafico prodotto è illustrato in Figura 7.20

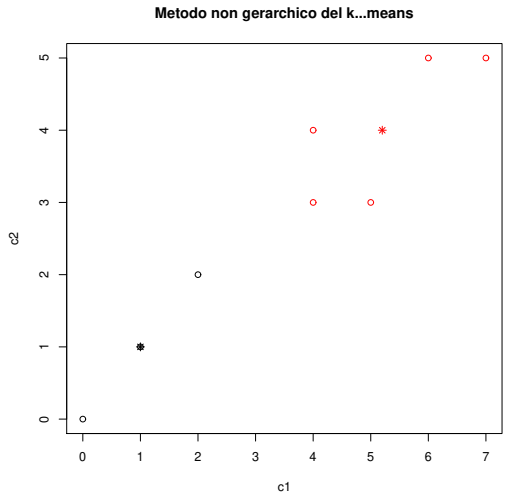

Figura 7.20: Partizione in cluster con il metodo del k-means.

In Figura 7.20 sull'asse orizzontale sono riportati i valori della caratteristica C_1 e sull'asse verticale quelli della caratteristica C_2. Gli individui appartenenti al cluster G_1 sono evidenziati con pallini rossi mentre quelli del cluster G_2 con pallini neri. Sono anche evidenziati il centroide $P_1 = (1, 1)$ del cluster G_1 e il centroide $P_2 = (5.2, 4)$ del cluster G_2.

Se il numero di caratteristiche è maggiore di due si può invece utilizzare la funzione scatterplot().

Esempio 7.12 Applichiamo il metodo k-means all'Esempio 7.1 in cui si consi-

dera la seguente matrice delle misure:

$$X = \begin{array}{c} \\ I_1 \\ I_2 \\ I_3 \\ I_4 \\ I_5 \end{array} \begin{array}{c} \begin{array}{cc} C_1 & C_2 \end{array} \\ \begin{pmatrix} 1 & 1 \\ 1 & 2 \\ 6 & 3 \\ 8 & 2 \\ 8 & 0 \end{pmatrix} \end{array}$$

Utilizzando R definiamo e scaliamo tale matrice dei dati:

```
> X<-data.frame(c1=c(1,1,6,8,8),c2=c(1,2,3,2,0))
> row.names(X)<-c("I1","I2","I3","I4","I5")
> X # visualizza il data frame X
   c1 c2
I1  1  1
I2  1  2
I3  6  3
I4  8  2
I5  8  0
>
> Z<-scale(X)
> Z # visualizza la matrice scalata
           c1         c2
I1 -1.0663057 -0.5262348
I2 -1.0663057  0.3508232
I3  0.3367281  1.2278812
I4  0.8979417  0.3508232
I5  0.8979417 -1.4032928
attr(,"scaled:center")
 c1  c2
4.8 1.6
attr(,"scaled:scale")
      c1       c2
3.563706 1.140175
```

Abbiamo mostrato che una suddivisione in *due cluster utilizzando il metodo del legame singolo, del legame completo e del legame medio* conduce alla partizione $C_1 = \{I_5\}$ e $C_2 = \{I_1, I_2, I_3, I_4\}$; la misura di non omogeneità totale relativa alla matrice scalata Z è tr $T = 8$, la misura di non omogeneità interna ai cluster (within) per la matrice scalata è tr $S = 4.530588$ e la misura di non omogeneità tra i cluster (between) è tr $B =$ tr $T -$ tr $S = 8 - 4.530588 = 3.469412$, da cui segue che

$$\frac{\text{tr } B}{\text{tr } T} = 0.433677.$$

Quindi, la suddivisione in due cluster, ottenuta con i metodi gerarchici (single, complete, average), non è risultata soddisfacente.
Consideriamo ora una suddivisione in due cluster applicando il metodo k-means:

```
> kmeans(Z,center=2,iter.max=10,nstart=1)
K-means clustering with 2 clusters of sizes 2, 3

Cluster means:
```

```
          X1          X2
1 -1.0663057 -0.08770580
2  0.7108705  0.05847053

Clustering vector:
I1 I2 I3 I4 I5
 1  1  2  2  2

Within cluster sum of squares by cluster:
[1] 0.3846154 3.7997173
 (between_SS / total_SS =  47.7 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"
[5] "tot.withinss" "betweenss"    "size"
```

Una suddivisione in due cluster a partire dalla matrice scalata Z con il metodo del k-means fornisce una differente partizione $C_1 = \{I_1, I_2\}$ e $C_2 = \{I_3, I_4, I_5\}$. La misura di non omogeneità interna (within) è $\operatorname{tr} S = 4.184332$ e quella tra i cluster $\operatorname{tr} B = \operatorname{tr} T - \operatorname{tr} S = 8 - 4.184332 = 3.815668$, da cui segue che

$$\frac{\operatorname{tr} B}{\operatorname{tr} T} = 3.815668/8 = 0.4769585.$$

La partizione ottenuta con il metodo non gerarchico è migliore rispetto a quella ottenuta con i metodi gerarchici ma è ancora non soddisfacente.
Applichiamo quindi il metodo del k-means considerando una *suddivisione in tre cluster*:

```
> kmeans(Z,center=3,iter.max=10,nstart=1)
K-means clustering with 3 clusters of sizes 1, 2, 2

Cluster means:
          X1          X2
1  0.8979417 -1.4032928
2 -1.0663057 -0.0877058
3  0.6173349  0.7893522

Clustering vector:
I1 I2 I3 I4 I5
 2  2  3  3  1

Within cluster sum of squares by cluster:
[1] 0.0000000 0.3846154 0.5420957
 (between_SS / total_SS =  88.4 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"
[5] "tot.withinss" "betweenss"    "size"
```

Si osserva che una suddivisione in tre cluster a partire dalla matrice scalata Z con il metodo del k-means fornisce la partizione $C_1 = \{I_1, I_2\}$ e $C_2 = \{I_3, I_4\}$, $C_4 = \{I_5\}$. La misura di non omogeneità del cluster C_1 è 0.3846154, del cluster

C_2 è 0.5420957 e del cluster C_3 è nulla; quindi la misura di non omogeneità interna (within) è $\operatorname{tr} S = 0.9267111$ e la misura di non omogeneità tra i cluster è $\operatorname{tr} B = \operatorname{tr} T - \operatorname{tr} S = 7.073289$. Risulta

$$\frac{\operatorname{tr} B}{\operatorname{tr} T} = 7.073289/8 = 0.8841611.$$

Pertanto, con tre cluster il metodo del k-means ha fornito una buona suddivisione. I metodi gerarchici (single, complete, average) conducono alla stessa suddivisione in tre cluster e quindi alle stesse misure di non omogeneità statistiche.

In conclusione, i metodi gerarchici presentano un evidente vantaggio dal punto di vista computazionale rispetto ai metodi di enumerazione completa; tuttavia risultano sensibili a vettori delle caratteristiche anomali e non consentono di modificare la configurazione raggiunta, ossia una volta che un individuo è stato attribuito ad un cluster permane al suo interno per sempre. I metodi non gerarchici non presentano questo problema poiché consentono di riallocare gli individui precedentemente classificati, ma richiedono una scelta opportuna della configurazione iniziale.

È buona norma applicare una pluralità di metodi per verificare la stabilità dei gruppi e scegliere la partizione (e il metodo) che, a parità di numero di cluster, fornisce le migliori misure di non omogeneità.

Indice