UNIVERSITÀ DEGLI STUDI DI SALERNO

Dipartimento di Informatica

Corso di Laurea Magistrale in Informatica: Software Engineering and IT Management

Relazione di Statistica e Analisi dei Dati

DOCENTE

Prof. **Amelia G. Nobile**

STUDENTE

Antonio Trovato

Matricola: 0522501270

PARTE UNO

Prefazione

Il presente documento descrive le fasi della attività di analisi statistica condotta sulla base di uno dei tanti dataset pubblicati dal Dipartimento della Protezione Civile. Il tema di quest'analisi consiste nel rapporto dettagliato di tutte le informazioni utili alla comprensione dello stato in cui si trova l'Italia in merito all'emergenza COVID-19 nel giorno 2 febbraio 2023.
Questo documento si suddivide in più capitoli, i quali hanno lo scopo di disquisire riguardo argomenti statistici strettamente legati fra loro. In questo modo si vuole fornire del contenuto nozionistico, riguardante esclusivamente la statistica e dunque non dedotto dallo studio statistico preso in esame, sia inferenziale; quest'ultimo contenuto riguarda le proposizioni statistiche che saranno derivate durante o studio dal dataset (tali informazioni saranno spesso anche descritte graficamente con l'uso di diagrammi diversi).
Il capitolo introduttivo è dedicato alle discipline statistiche ed alle loro suddivisioni. Successivamente inizia lo studio del dataset a partire da un'esposizione iniziale dei dati e da una descrizione delle caratteristiche che ne hanno decretato la scelta, con l'aggiunta di dettagli sui diversi criteri di suddivisione granulare selezionati.

Introduzione alla statistica

La statistica è una disciplina che studia in termini quantitativi e qualitativi un particolare fenomeno collettivo che non può essere studiato in termini deterministici. Questa incertezza può essere in sé oppure causata dalla mancanza di conoscenza (sufficiente) per poter fare previsioni. Secondo il dizionario Palazzi, la statistica è "*la scienza dei fatti sociali espressi con termini numerici*".
La statistica, avvalendosi della matematica, offre agli studiosi potenti strumenti di comprensione, sintesi e analisi dei fenomeni studiati, dando modo di rilevare eventuali correlazioni tra due o più fenomeni e misurare il grado di precisione con il quale dichiarare legami di causa-effetto. I metodi

statistici permettono di analizzare informazioni numeriche - si parla di proprietà quantitative - oppure no - quindi proprietà qualitative.
Un altro importante aspetto della statistica è il frequente uso di varie tipologie di diagrammi. Molti aspetti che saranno mostrati matematicamente verranno immediatamente confermati (se non chiariti) tramite rappresentazioni grafiche mediante diagrammi. Per generare i grafici, così come per elaborare tutti i calcoli che costituiscono lo studio, viene usato il linguaggio di programmazione R.

Il caso di studio

Come anticipato, gli argomenti statistici trattati in questo documento saranno prima esposti in maniera “data-agnostic” (indipendenti dai dati in esame) e poi messi in pratica attraverso le informazioni presenti nel dataset in esame. In particolare, questi sono dati riguardanti la situazione dello stato italiano in merito all’emergenza covid rilevati in data 2 febbraio 2023. Queste informazioni sono suddivise geograficamente in regioni italiane, in modo da partire già con un raggruppamento iniziale. In particolare, le informazioni che ci interessano per ogni regione sono: “terapia intensiva”, “totale ospedalizzati”, “totale positivi”, “tamponi”, “ingressi terapia intensiva”.
Ecco il dataset in questione.

denominazione_regione	terapia_intensiva	totale_ospedalizzati	totale_positivi	tamponi	ingressi_terapia_intensiva
Abruzzo	2	94	10376	7362338	0
Basilicata	0	19	8361	1319598	0
Calabria	10	143	1328	4196036	0
Campania	14	287	33809	20317146	0
Emilia-Romagna	29	639	5797	19185456	0
Friuli Venezia Giulia	4	66	886	7629872	2
Lazio	18	527	24013	25892909	0
Liguria	5	136	1139	6817767	1
Lombardia	21	248	14516	44798340	0
Marche	3	69	1680	3715869	0
Molise	0	6	1323	791057	0
P.A. Bolzano	1	28	287	5554873	0
P.A. Trento	2	24	325	3023006	0
Piemonte	3	167	26345	21202906	0
Puglia	10	235	15173	13696376	0
Sardegna	4	179	3462	5395855	1
Sicilia	29	424	12852	16360133	0
Toscana	4	178	47794	16548386	0
Umbria	4	119	1460	4995558	2
Valle d'Aosta	0	2	627	585904	0
Veneto	16	301	16432	36684401	4

In questo dataset, quelle che vengono chiamate regioni in realtà tecnicamente sono delimitazioni del territorio italiano. Infatti, troviamo alcune province autonome.
A questo punto comincia l'analisi dei dati considerati partendo da una vista di alto livello sulla quantità di persone che sono risultate positive il 2 febbraio 2023.

1. Grafici ed indici di posizione centrali

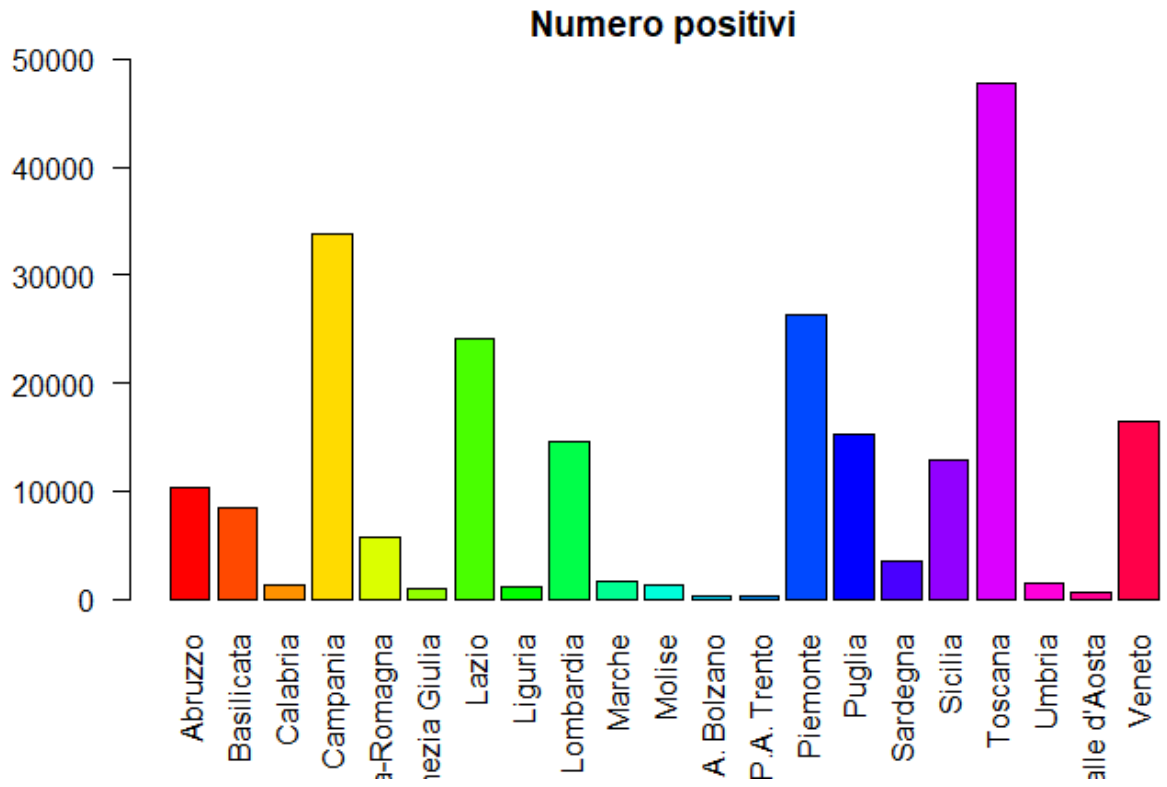
1.1 Variabili quantitative

Una variabile quantitativa $X = (x_1, x_2, \dots, x_3)$ è una collezione di n valori numerici. Spesso, i valori di una variabile quantitativa si suddividono nelle classi $z_1, z_2, \dots, z_n$ ordinate in maniera crescente. Quindi si considerano n_i = *frequenza assoluta* z_i *in* X e $f = \frac{n_i}{n}$ la frequenza relativa z_i in X. Siano inoltre $F_i = f_1 + \cdots + f_i$ le frequenze relative cumulate. I dati raccolti per le analisi statistiche saranno modellati attraverso variabili quantitative.

1.2 Grafico a barre delle persone risultate positive

Come detto, la statistica offre strumenti grafici molto potenti al fine di analizzare le relazioni quantitative e qualitative che insistono nei dati. La prima rappresentazione grafica di cui ci si avvarrà è quella del grafico a barre.
Un *grafico a barre* è uno strumento di rappresentazione grafica che permette di mostrare le differenze tra i valori presenti in una variabile quantitativa, rappresentando ogni misura tramite una barra (spesso verticale, ma può essere anche orizzontale) posizionata in uno spazio in cui è presente un'asse verticale che scandisce il progressivo aumento di misura man mano che si sale verso l'alto.
Qui verranno mostrati e analizzati una serie di grafici a barre che analizzano i dati estrapolati dal dataset in esame. Il dataset in studio è stato memorizzato in R nel data frame ***dataset***.
Mediante la variabile ***dataset$denominazione_regione*** è possibile accedere alla colonna del dataset relativa ai nomi delle regioni italiane.
Il seguente codice mostra un grafico a barre dei valori nella colonna relativa al numero di positivi.

```
barplot(dataset$totale_positivi, names=dataset$denominazione_regione,
        col=rainbow(length(dataset$denominazione_regione)), ylim=c(0, 50000), las=2,
        main="Numero positivi")
```



Dal grafico risulta evidente che, escluse le provincie autonome, la Valle d'Aosta è la regione che ha presentato il minor numero di positivi, mentre in Toscana, al contrario, è stato rilevato il maggior numero di casi di persone positive al COVID-19.

1.3 Indici di posizione centrali

Gli indici statistici di posizione che saranno descritti in questo documento si dividono in *centrali* e *non centrali*. Essi, in generale, possono fornire indicazioni approssimate circa le grandezze dei valori presenti nella variabile quantitativa considerata.

Dato un campione $(x_1, x_2, \dots, x_n)$, si considerano la media campionaria come $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$, la mediana campionaria come $\frac{x_{\frac{n}{2}} + x_{\frac{n}{2}+1}}{2}$ se n è pari, altrimenti $x_{\frac{n+1}{2}}$.

Infine, la moda campionaria corrisponde al valore (oppure modalità) più frequente. Se il campione è costituito da ripetizioni di un solo valore allora

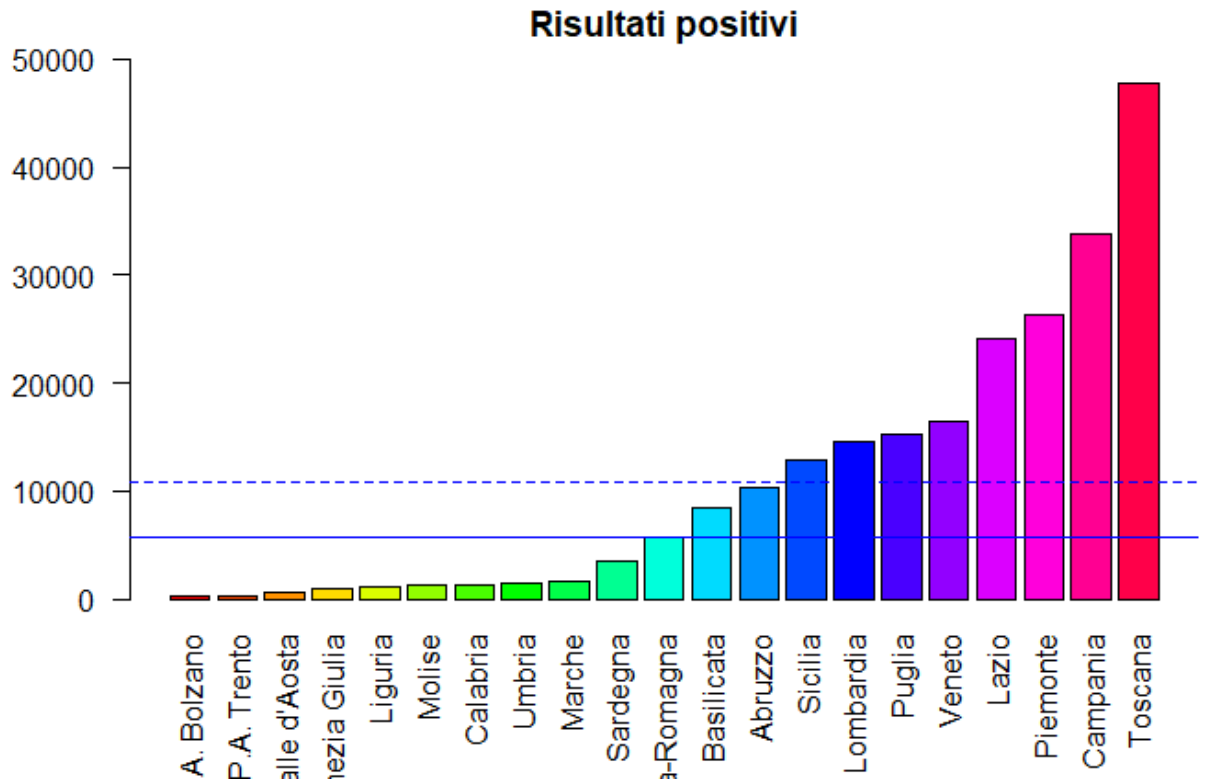
esso è priva di moda campionaria. In R, si calcola la media con il comando ***mean*** e la mediata con il comando ***median***.
Il seguente codice calcola alcune informazioni riguardo i valori degli indici di posizione centrali dei dati nella colonna relativa al numero di persone risultate positive il 2 febbraio 2023.

```
data_totale_positivi <- data.frame("Minimo"=min(dataset$totale_positivi),
                                   "Massimo"=max(dataset$totale_positivi),
                                   "Media"=mean(dataset$totale_positivi),
                                   "Mediana"=median(dataset$totale_positivi))
print(data_totale_positivi)
```

In particolare, il minimo risulta 287, il massimo 47794, la media 10856.43 e la mediana 5797.

1.4 Grafico a barre ordinate

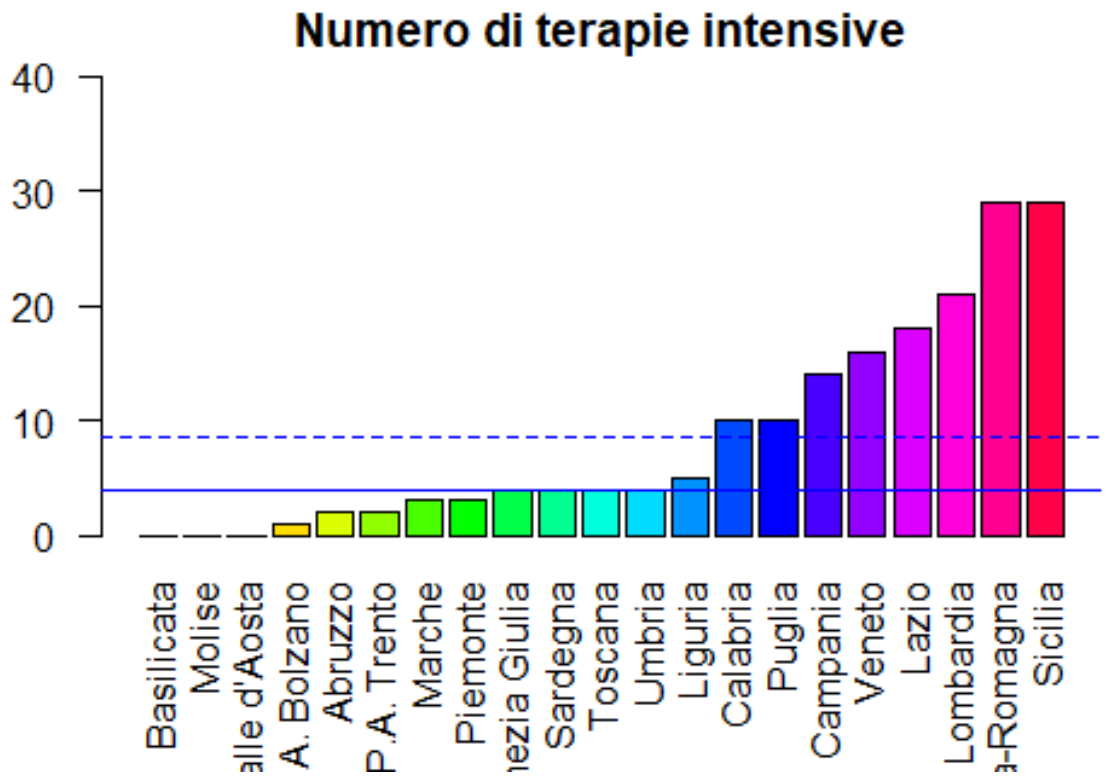
Possiamo infatti renderci conto della veridicità di questi valori dando uno sguardo a un grafico che mostra le stesse informazioni ma con un ordinamento sulle ascisse, grazie al seguente codice.



Risulta evidente il numero più basso è della provincia di Bolzano, il più alto invece è rappresentativo della Toscana. La linea tratteggiata indica la media campionaria del numero di positivi, mentre quella continua ne rappresenta la mediana.

1.5 Grafico a barre sul numero di persone in terapia intensiva

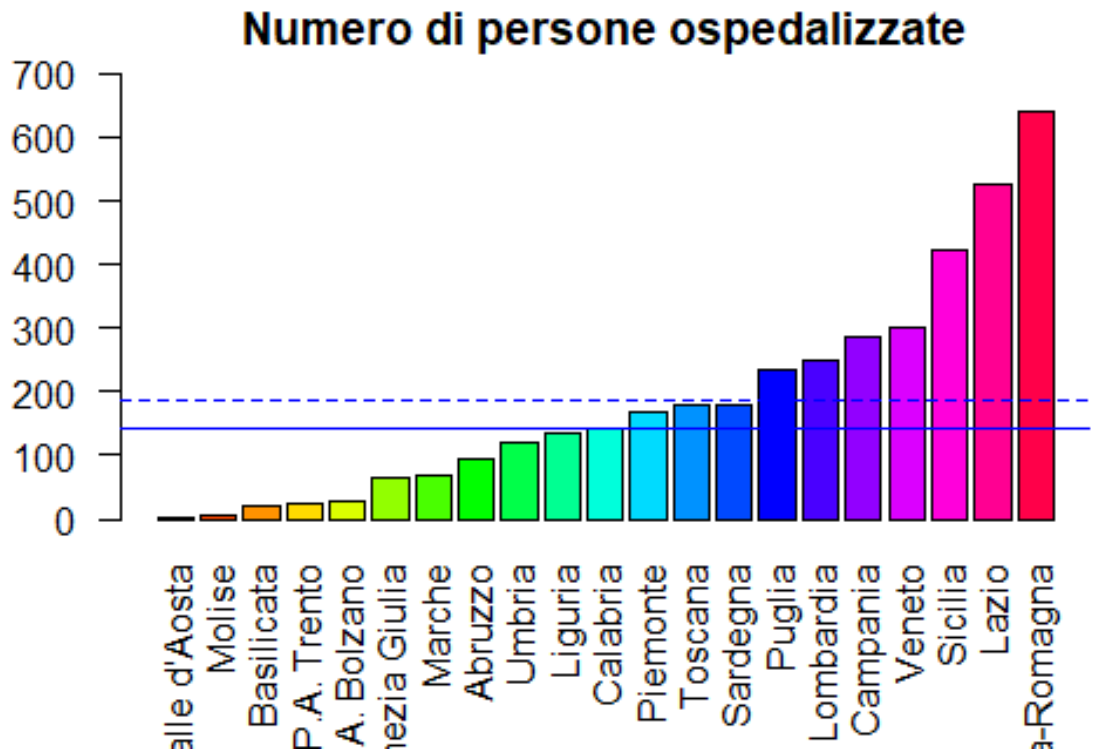
Si analizzeranno ora i dati delle altre colonne nel dataset, partendo da quella relativa al numero di persone in terapia intensiva.



Il grafico appena mostrato indica i numeri di persone, suddivisi per regione, che si trovano in terapia intensiva il 2 febbraio 2023. Mediante un codice R simile a quello visto per la generazione di informazioni riguardo indici di posizione centrali del precedente dataset, otteniamo che la media campionaria è pari a 8.5 mentre la mediana campionaria a 4.

1.6 Grafico a barre sul numero totale di persone ospedalizzate

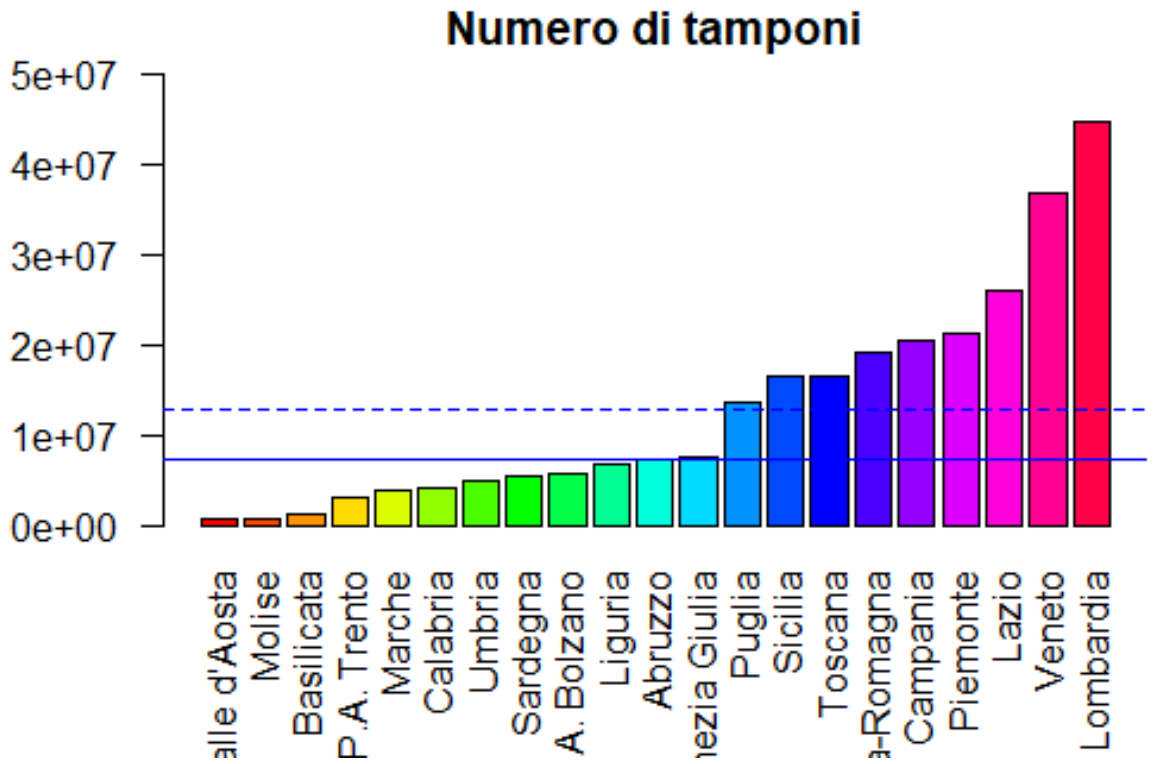
Si analizzeranno ora i dati relativi al numero di persone ospedalizzate.



Tramite R è stato possibile calcolare che la mediana campionaria risulta equivalente a 185.28 mentre la mediana campionaria è pari a 143.

1.7 Grafico a barre sul numero di tamponi

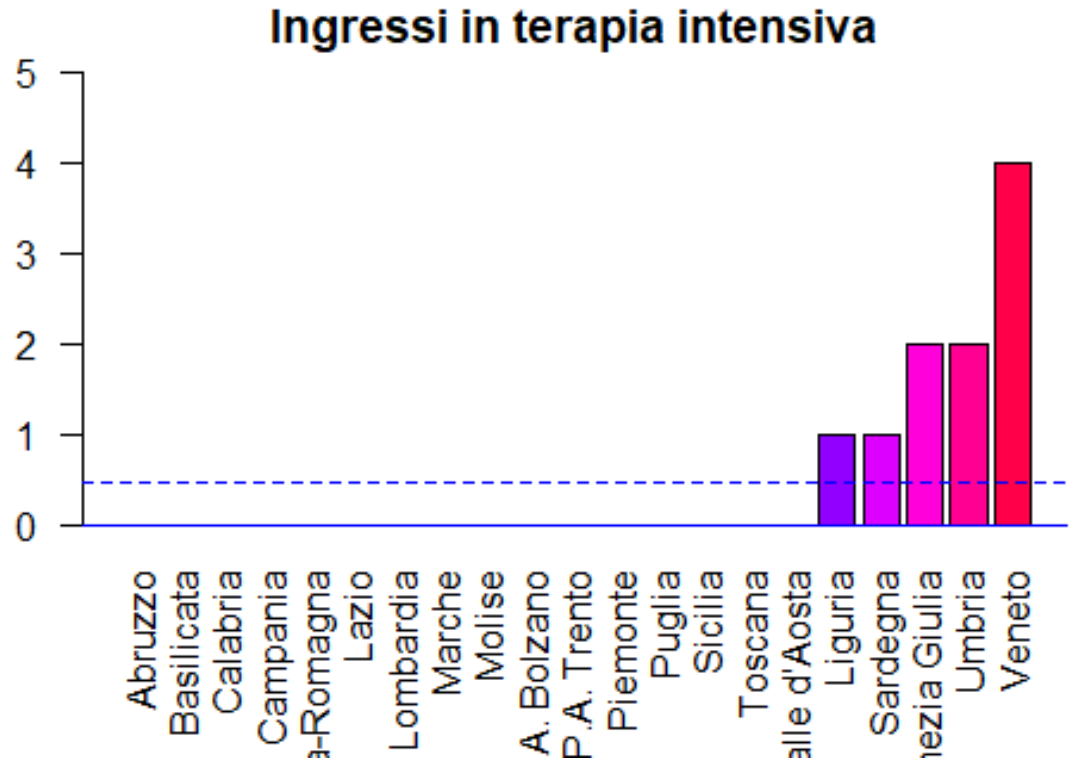
Si analizzeranno ora i dati delle relativi al numero di tamponi effettuati.



Tramite l'analisi di tali dati, la media campionaria risulta pari a 12670180, mentre la mediana campionaria è di 7362338.

1.8 Grafico a barre sul numero di ingressi in terapia intensiva

Si analizzeranno ora i dati relativi al numero di persone che sono entrate in terapia intensiva.



Dall'analisi di questi dati ricaviamo che la media campionaria è pari a circa 0.5, mentre la mediana campionari è pari a 0.

2. Frequenze e Distribuzioni

2.1 Istogramma del numero di risultati positivi

Molto utile è la suddivisione dei valori assunti da una variabile quantitativa in classi per studiarne la distribuzione di frequenze. In generale, le frequenze che non sono troppo dissimili fra loro posso essere accorpate in classi per poterne studiare l'andamento di un argomento al fine di catturarne alcune informazioni quantitative. Una delle modalità più usate per l'analisi delle distribuzioni di frequenze consiste nella realizzazione di *istogrammi*.

Un istogramma è una rappresentazione grafica il cui compito è associare ad ogni classe di frequenze un rettangolo che ha per estremi orizzontali i valori estremi della classe, mentre per altezza ha la quantità di istanze nel campione che fanno parte di quella classe. Viene rappresentata un'asse orizzontale che percorre così tutti i valori assunti dal campione, avendo poi come asse verticale i numeri che costituiscono le frequenze (assolute o relative) della distribuzione.

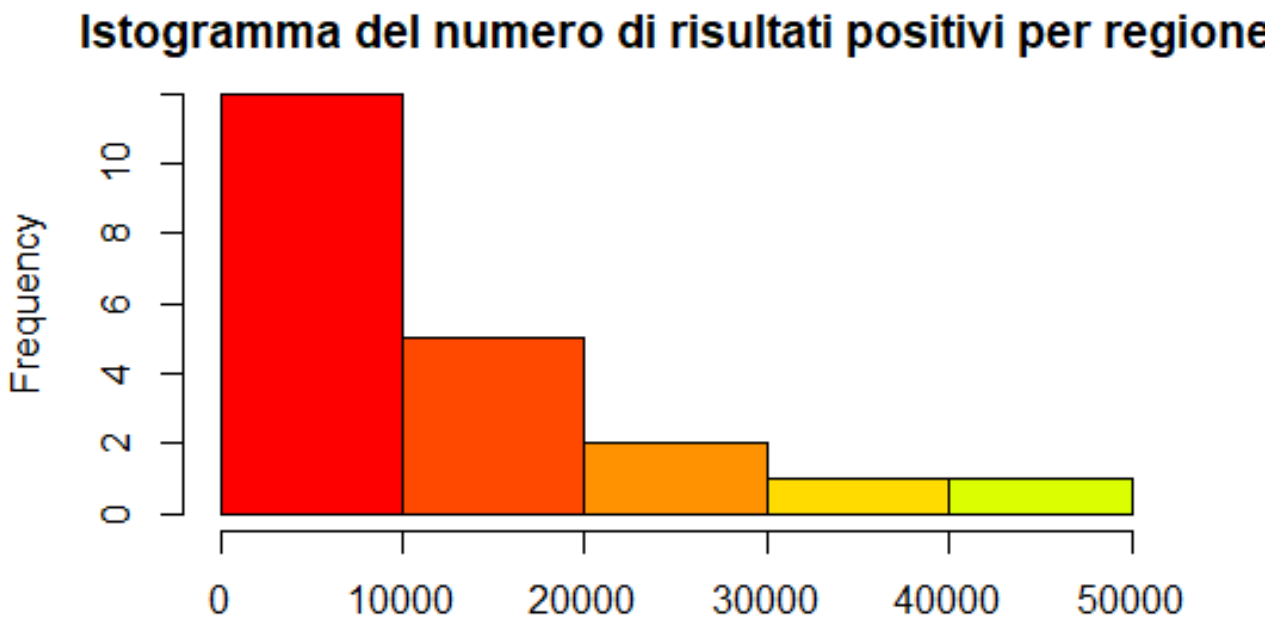
Per studiare la distribuzione risulta determinante eseguire la suddivisione in classi. Critica per ottenere informazioni utili è dunque la scelta del numero di classi. Viene sconsigliata, a causa del probabile oscuramento di dettagli utili, la suddivisione in un numero esiguo di classi di una distribuzione di frequenze molto variata.

Mettendo in pratica gli argomenti appena esposti, adesso si analizzeranno delle informazioni estrapolate dai valori del dataset preso in esame. La prima caratteristica da evidenziare corrisponde alle relazioni dei dati relativi al numero di risultati positivi ai tamponi eseguiti il 2 febbraio 2023. I diversi valori regionali sono raggruppati come mostra l'istogramma di frequenze assolute che ci si accinge a mostrare.

Il seguente codice produrrà l'istogramma richiesto.

```
istogramma_totale_positivi <- function(name, camp) {
  h_t_p <- hist(camp, freq=TRUE, main=name,
col=rainbow(length(camp)), xlab="")
  h_t_p
}
istogramma_totale_positivi("Istogramma del numero di risultati
positivi per regione", dataset$totale_positivi)
```

Di seguito è mostrato l'istogramma ottenuto.




```
$breaks
[1]     0 10000 20000 30000 40000 50000

$counts
[1] 12  5  2  1  1

$density
[1] 5.714286e-05 2.380952e-05 9.523810e-06 4.761905e-06
[5] 4.761905e-06

$mids
[1]  5000 15000 25000 35000 45000
```

Osservando l'istogramma, appare piuttosto evidente che la gran parte delle regioni sono state raggruppate nelle prime due classi (0-10000 e 10000-20000).

2.2 Indici di posizione non centrali

Tra gli indici di posizione troviamo gli indici di posizione non centrali. Questi si suddividono in quantili, quartili e percentili. Si consideri una variabile quantitativa che abbia dei valori orizzontalmente disposti in un ordine non decrescente, ossia dove il valore più a sinistra costituisce il minimo dei valori di questa. In funzione della variabile stessa, le distanze tra due valori adiacenti possono essere o non essere regolari e omogenee. Un altro strumento statistico per studiare le distribuzioni di frequenze è quello di usare quantili, quartili e percentili. Un quantile è un valore che divide in due la suddetta sequenza di valori ordinati, in modo tale che a sinistra esistano soltanto valori minori e a destra soltanto valori maggiori. Tre quantili si chiamano quartili se dividono in quattro parti la sequenza ordinata, ovvero il 25% dei dati sono alla sinistra del primo quartile, il 50% alla sinistra del secondo quartile (che coincide con la mediana della variabile quantitativa), il 75% alla sinistra del terzo quartile.

Analogamente, novantanove percentili suddividono i dati ordinati in cento sotto sequenze.

In R è possibile ottenere la suddivisione in quartili tramite la funzione *quantile*.

2.3 Boxplot

Il *boxplot*, o diagramma a scatola e baffi, è una rappresentazione grafica che ha il compito di delimitare i quartili di una variabile quantitativa. In particolare, il primo quartile e il terzo quartile delimitano gli estremi di un rettangolo (la scatola), diviso in due da un segmento all'altezza del secondo quartile. Questa scatola è confinata da ambo i lati da due baffi che hanno con estremità dei valori assunti dalla variabile quantitativa che sono rispettivamente il valore minore e il valore maggiore in un dato intervallo di tolleranza. I valori della variabile quantitativa che oltrepassano questi limiti (in entrambe le direzioni) vengono chiamati valori anomali (anche detti outliner).

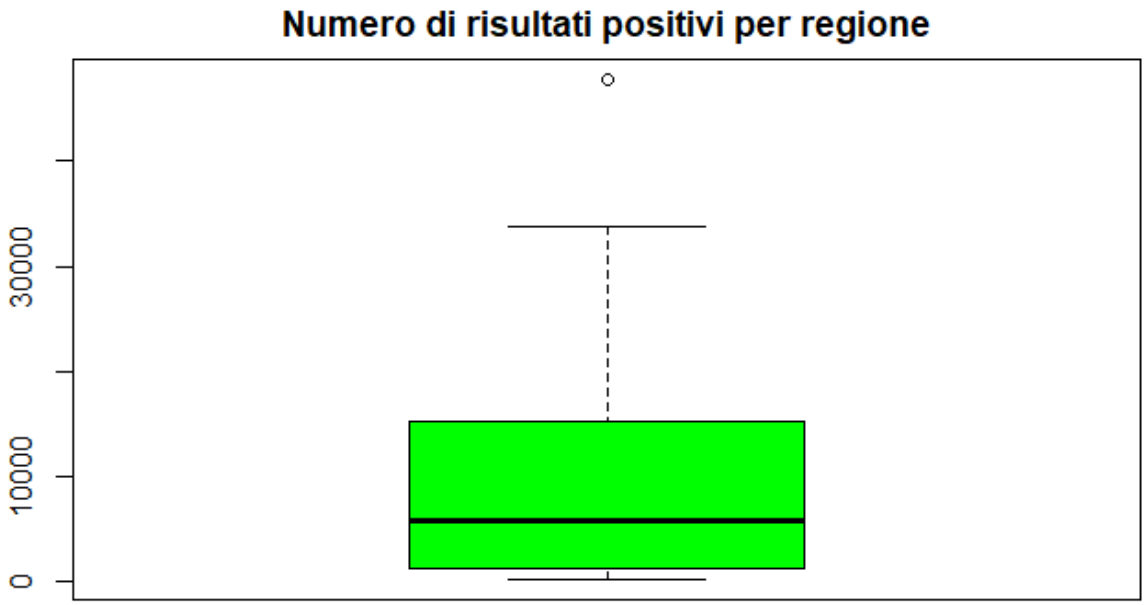
Il boxplot ad intaglio è invece una variante del boxplot di base che presenta un intaglio all'altezza della mediana la cui ampiezza esprime la

quantità di valori della variabile quantitativa appartenenti all'intervallo di confidenza del 95% per la mediana.

2.4 Boxplot del numero di risultati positivi suddivisi per regioni

Di seguito si mostrano il codice in R per la generazione del boxplot ed il boxplot stesso.

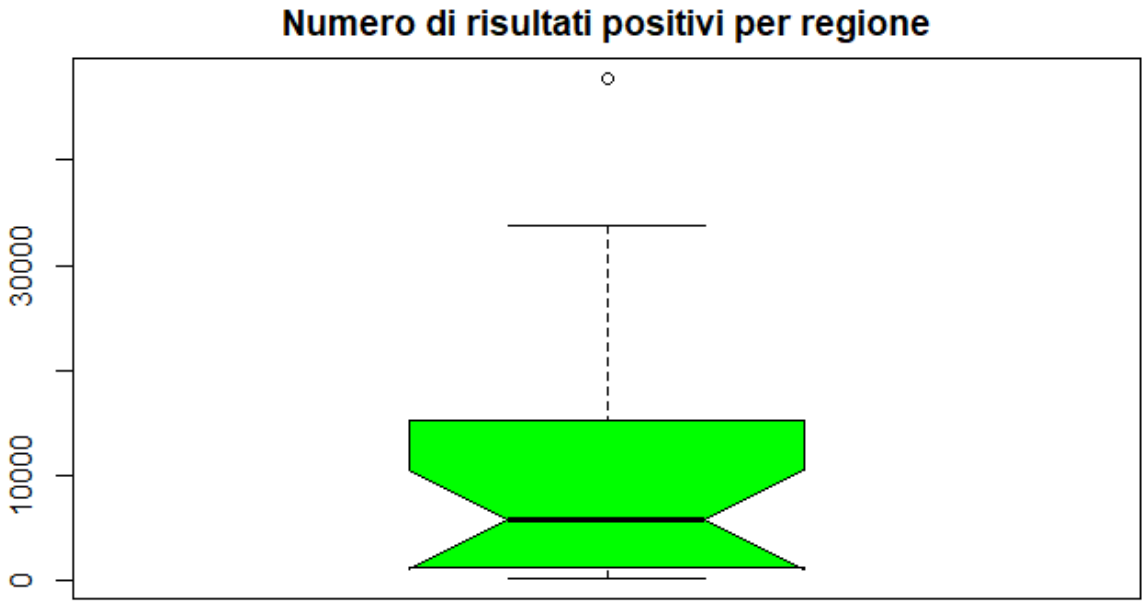
```
boxplot(dataset$totale_positivi, col="green", main="Numero di risultati
positivi per regione")
Summary<-boxplot(dataset$totale_positivi, col="green", main="Numero di
risultati positivi per regione")$stats
rownames(Summary)<-c("Min","First Quartile","Median","Third
Quartile","Maximum")
Summary
```



Di seguito si riportano dei dati analitici sul boxplot appena ottenuto.

```
Min              287
First Quartile  1323
Median          5797
Third Quartile 15173
Maximum        33809
```

Per evidenziare ulteriori informazioni mostriamo il seguente boxplot a intaglio (come detto nella sezione teorica, con intervallo di confidenza al 95% per la mediana).



2.5 Principio di Pareto

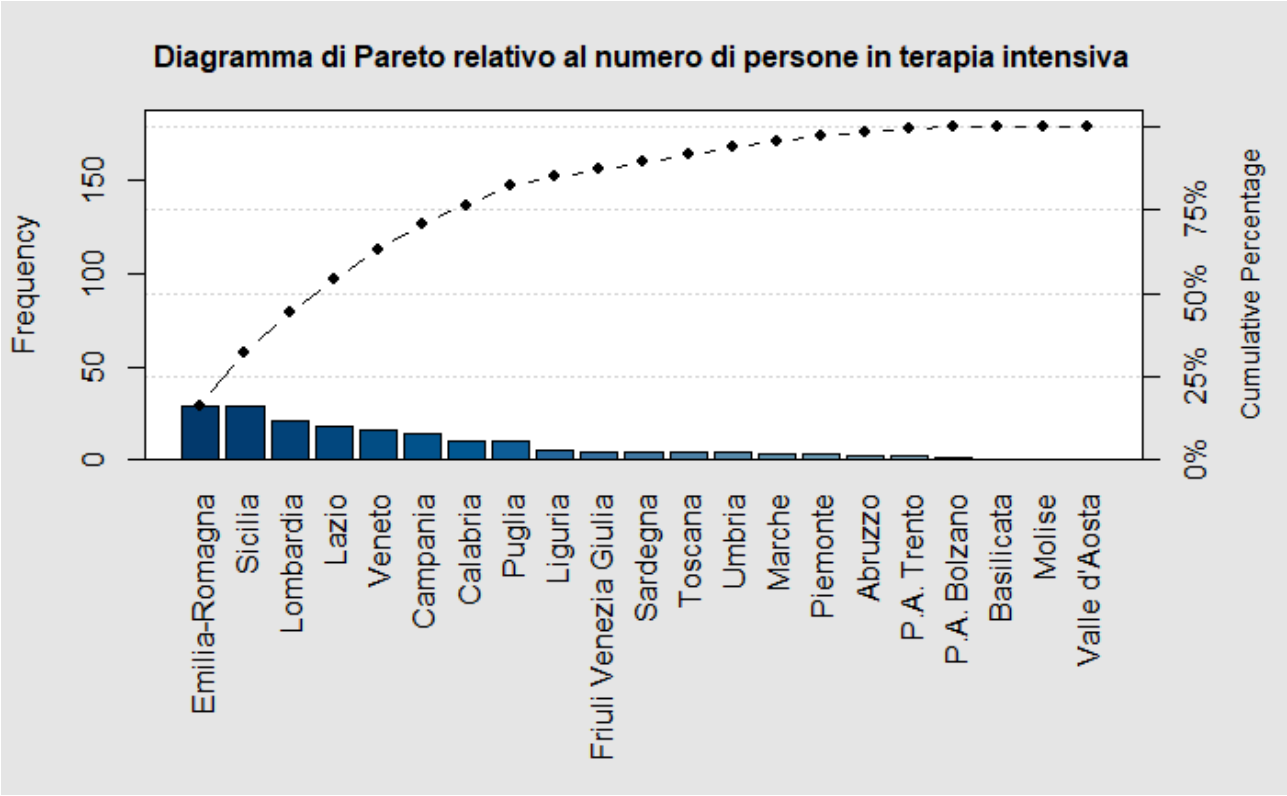
Il principio di Pareto è un principio che prevede che circa l'80% dei risultati derivino da circa il 20% delle cause. Questo è un principio empirico che si verifica spesso in sistemi complessi basati su strutture causa-effetto. Può essere ulteriormente sintetizzato nel seguente modo: "la maggior parte degli effetti è dovuto a un numero ristretto di cause". Esso è poi ulteriormente supportato dalla distribuzione paretiana.

Altri esempi possono essere:

- il 20% dei venditori esegue l'80% delle vendite;
- il 20% delle istruzioni di un programma viene eseguito l'80% del tempo (a supporto del principio di località temporale);
- l'80% dei ricavi di treni e aerei deriva dal 80% di rotte;
- l'80% del successo di una persona deriva dal 20% delle sue imprese.

Prende il nome da Vilfredo Pareto che, studiando la distribuzione dei redditi, nel 1897 evinse che solo pochi individui possedevano la maggior parte delle ricchezze in una data regione.

Ecco, infine il diagramma di Pareto relativo al numero di terapie intensive.



3. Dispersione e Funzione di Distribuzione

3.1 Varianza

Il primo indice di dispersione statistico che si evidenzia immediatamente è la varianza campionaria. Essa fornisce una misura di dispersione statistica, in particolare della variabilità dei valori assunti da una variabile statistica o aleatoria X. Dato un campione $X = (x_1, x_2, \dots, x_n)$, la varianza è pari a:

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

Come è facile notare, l'unità di misura della varianza è il quadrato dell'unità di misura dei dati del relativo campione.
Nell'ambiente R è possibile calcolare la varianza di un *dataframe df* usando direttamente il comando ***var(df)***.

3.2 Deviazione Standard

Un indicatore statistico strettamente legato alla varianza è la deviazione standard. Essa è un indice di dispersione statistico che descrive quanto i dati x_i di un certo campione X si discostano dalla *media campionaria* (indice di posizione centrale che verrà descritto nel prossimo capitolo). Dato un campione di dati $X = (x_1, x_2, \dots, x_n)$, la deviazione standard è pari a $s = \sqrt{s^2}$, ovvero la radice quadrata del campione X. Si può facilmente notare che l'unità di misura della deviazione standard è la stessa di quella dei dati del campione di partenza.
Nell'ambiente R è possibile calcolare la deviazione standard di un *dataframe df* usando direttamente il comando ***sd(df)***.

3.3 Coefficiente di variazione

Infine, il coefficiente di variazione è un altro indicatore statistico che descrive il valore di scostamento dei dati in un campione rispetto alla sua media campionaria. Esso si calcola con la seguente formula:

$$CV = \frac{s}{|\bar{x}|}$$

Questo valore è il primo indicatore statistico adimensionale esposto in questo documento. Esso è infatti il rapporto tra la deviazione standard e la media campionaria (in valore assoluto) di un campione dato, ossia un rapporto tra valori della stessa unità di misura, che quindi si elide. Questo indice di dispersione può essere considerato soltanto per campioni aventi media campionaria non nulla.
Nell'ambiente R è possibile calcolare il coefficiente di variazione di un *dataframe df* usando direttamente il comando ***cv(df)***.

3.4 Analisi degli indici di dispersione sulle variabili quantitative

Qui verranno mostrati e analizzati i dati estrapolati dal dataset per tutti i dati numerici relativo allo stato in cui si trovava l'Italia relativamente al problema COVID-19 il 2 febbraio 2023.

I dati relativi al numero di nuovi risultati positivi sono caratterizzati da una varianza pari a 166665183, una deviazione standard pari a 12909.89 ed un coefficiente di variazione pari a 118.92.
I dati relativi al numero di terapie intensive sono caratterizzati da una varianza pari a 84.4619, una deviazione standard pari a 9.19 ed un coefficiente di variazione pari a 107.81.
I dati relativi al numero di persone ospedalizzate sono caratterizzati da una varianza pari a 29950.61, una deviazione standard pari a 173.06 ed un coefficiente di variazione pari a 93.40.
I dati relativi al numero di tamponi eseguiti sono caratterizzati da una varianza pari a 1.449247e+14, una deviazione standard pari a 12038466 ed un coefficiente di variazione pari a 95.01.
I dati relativi al numero di ingressi in terapia intensiva sono caratterizzati da una varianza pari a 1.06, una deviazione standard pari a 1.03 ed un coefficiente di variazione pari a 216.40.

Come si può notare dai risultati, il coefficiente di variazione è l'unico dei tre che non varia così tanto rispetto agli altri due. Infatti, questo è l'unico numero puro tra i tre e può essere considerato un indice di informazioni assolute non dipendente dalle grandezze complessive dei dati né ovviamente dall'unità di misura.

Si evidenzia inoltre che il coefficiente di variazione è più basso nel campione relativo al numero di persone ospedalizzate, ossia quello il cui grafico a barre risulta contenere le altezze più simili (vedi capitolo 1).

3.5 Funzione di Distribuzione Empirica

In probabilità spesso le informazioni che vengono studiate insistono nelle diverse variabili aleatorie - discrete o continue - di cui si considera la funzione di distribuzione. Nel caso dell'analisi dei dati, pur non avendo una variabile aleatoria si ha comunque la necessità di comporre in qualche modo una funzione di distribuzione.

Ecco che si crea una funzione di distribuzione empirica continua o discreta, in base al tipo di dati numerici (discreta se per esempio si considera il numero di studenti laureati con la lode nelle varie facoltà, continua per esempio la media dei voti di laurea di un gruppo di studenti universitari).

3.6 Funzione di Distribuzione Empirica Discreta

Quando si raggruppano le frequenze in classi, e non si considerano valori intermedi, allora una funzione di distribuzione empirica discreta può essere di aiuto. Questa è così definita:

$$F(x) = \frac{|\{x_i < x : i \in \{1,2,\dots,n\}\}|}{n}$$

Si verifica che $F(x) = 0 \Leftrightarrow x < z_i, F(x) = F_1 \Leftrightarrow z_1 \leq x \leq z_2, \dots,$
$F(x) = F_i \Leftrightarrow z_i \leq x \leq z_{i+1}, \dots, F(x) = 1 \Leftrightarrow x \geq z_k.$

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

- è una funzione a gradini
- è una funzione non decrescente

- $F(x) = 0, \forall x < z_1$ e $F(x) = 1, \forall x \geq z_k$

Nell'ambiente R, il grafico di una funzione di distribuzione empirica discreta può essere mostrato usando la seguente funzione:

```
discrDistr <- function(name, camp) {
  func <- round(cumsum(camp / length(camp)), 3)
  plot(ecdf(func), main=name, verticals=FALSE, col="red", xlab="")
  func
}
```

3.7 Funzione di Distribuzione Empirica Continua

Quando bisogna classificare i dati di una variabile quantitativa in classi $C_i = [z_{i-1}, z_i)$, modellare in qualche modo tutti i valori intermedi che insistono negli estremi di una classe, allora è bene considerare una funzione di distribuzione empirica continua. Questa è costruita come segue:

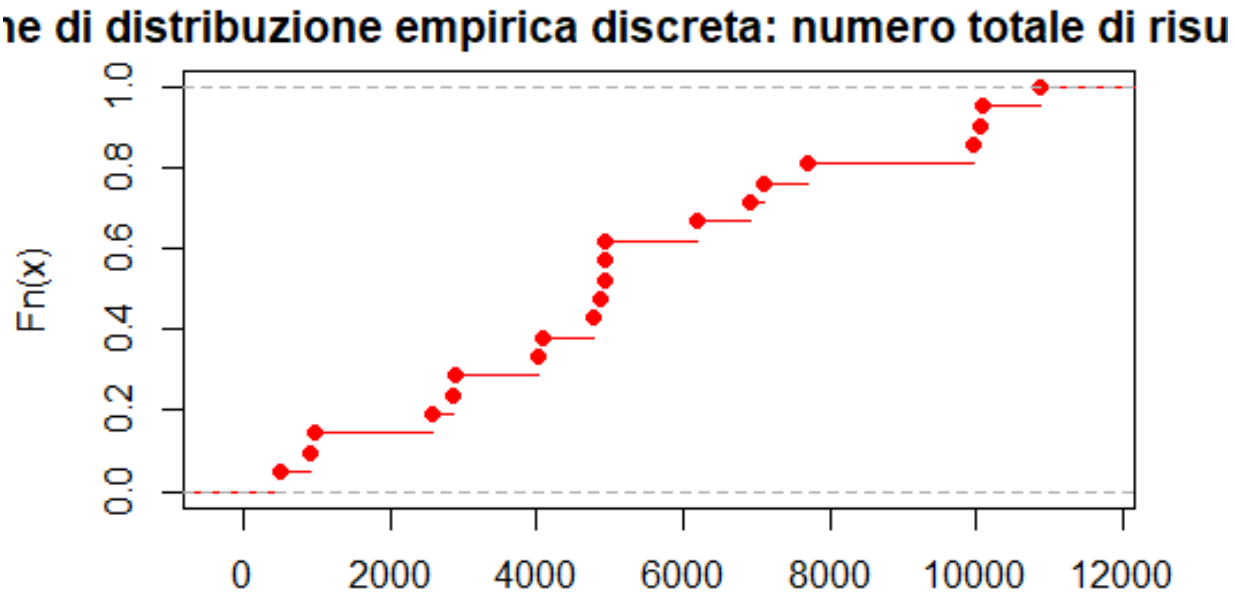
$$F(x) = \frac{F_i - F_{i-1}}{z_i - z_{i-1}} x + \frac{z_i F_{i-1} - z_{i-1} F_i}{z_i - z_{i-1}} \Leftrightarrow z_{i-1} \leq x \leq z_i$$

Si verifica che $F(z_i) = F_{z_i}$.

3.8 Funzione di Distribuzione Empirica Discreta del Numero di Risultati Positivi

Si analizza adesso la funzione di distribuzione empirica discreta ottenuta mediante i dati della colonna nel dataset corrispondente al numero di nuovi risultati positivi (ai tamponi) per regione. In seguito, è mostrato il grafico della suddetta funzione.

Il codice mostrato precedentemente, produce il prossimo grafico nel documento.



In seguito, saranno illustrati metodi più analitici per ricavare informazioni riguardo caratteristiche grafiche di una distribuzione di frequenze.

3.9 Skewness e Curtosi Campionarie

La statistica offre anche strumenti più analitici per caratterizzare la simmetria e la piccatezza di una distribuzione di frequenze. La skewness campionaria è un indice di simmetria della distribuzione di frequenze, e si calcola tramite i momenti centrati campionari di ordine due e tre. La curtosi campionaria, invece, indica la piccatezza di una distribuzione di frequenze e viene calcolata tramite l'indice di Pearson (che si basa sui momenti centrali campionari di ordine due e quattro).

La skewness campionaria e la curtosi campionaria del campione corrispondente alla colonna del numero totale di risultati positivi ai tamponi nel dataset possono essere calcolate in R tramite il seguente programma.

```r
skw <- function(camp) {
  n <- length(camp)
  m2 <- (n-1) * var(camp)/n
  m3 <- sum((camp - mean(camp))^3) / n
  m3 / (m2^1.5)
}
curt <- function(x) {
  n <- length(x)
  m2 <- (n-1) * var(x) / n
  m4 <- sum((x - mean(x))^4) / n
  m4 / m2^2 - 3
}
```

In particolare, la skewness campionaria della distribuzione di frequenze appena ottenuta è pari a 0.271 poiché infatti la distribuzione presenta una asimmetria negativa, ossia ha una coda sinistra più lunga come tendenza

La curtosi campionaria della distribuzione di frequenze appena ottenuta è pari -0.858, poiché infatti la distribuzione presenta una platicurtosi.

4. Regressioni Lineari

In questo nuovo capitolo si presenteranno vari argomenti riguardanti la statistica descrittiva multivariata. In particolare, ci si soffermerà sugli strumenti statistici per la rilevazione di relazioni tra due variabili, come la covarianza, la correlazione e le regressioni. Nell'applicare questi strumenti verrà ristretto il campo alla statistica descrittiva bivariata.

La statistica descrittiva bivariata è un ramo della statistica che si occupa dei metodi grafici e statistici atti a descrivere le relazioni che intercorrono tra due variabili.

Le modalità di rappresentazione grafica consistono nella composizione di diagrammi di dispersione (anche detti *scatter plot*) composti da un piano euclideo e da una serie di punti nel piano. Ogni punto è associato a una coppia di osservazioni appartenenti alle due variabili di cui si ha interesse a mostrare il grado di relazione reciproco.

4.1 Studio sulla correlazione tra il numero totale di persone ospedalizzate ed il numero totale di persone in terapia intensiva

La statistica descrittiva bivariata fornisce strumenti matematici che permettono di descrivere le relazioni che intercorrono tra due variabili: nella sezione di questo documento si mostreranno gli esiti degli studi effettuati sulla esistenza di relazioni tra il numero totale di persone ospedalizzate ed il numero totale di persone in terapia intensiva. Si descriveranno prima graficamente le correlazioni tra i vari campioni del dataset e si tireranno le somme riguardo a correlazioni più marcate rispetto ad altre.

In generale, le relazioni tra due variabili possono avere diversa natura, nel nostro caso ci focalizzeremo su quelle di tipo lineari.

Se tra due variabili intercorre una relazione lineare, questa può essere:

- positiva, se l'aumento di valore di una variabile implica l'aumento di valore dell'altra;

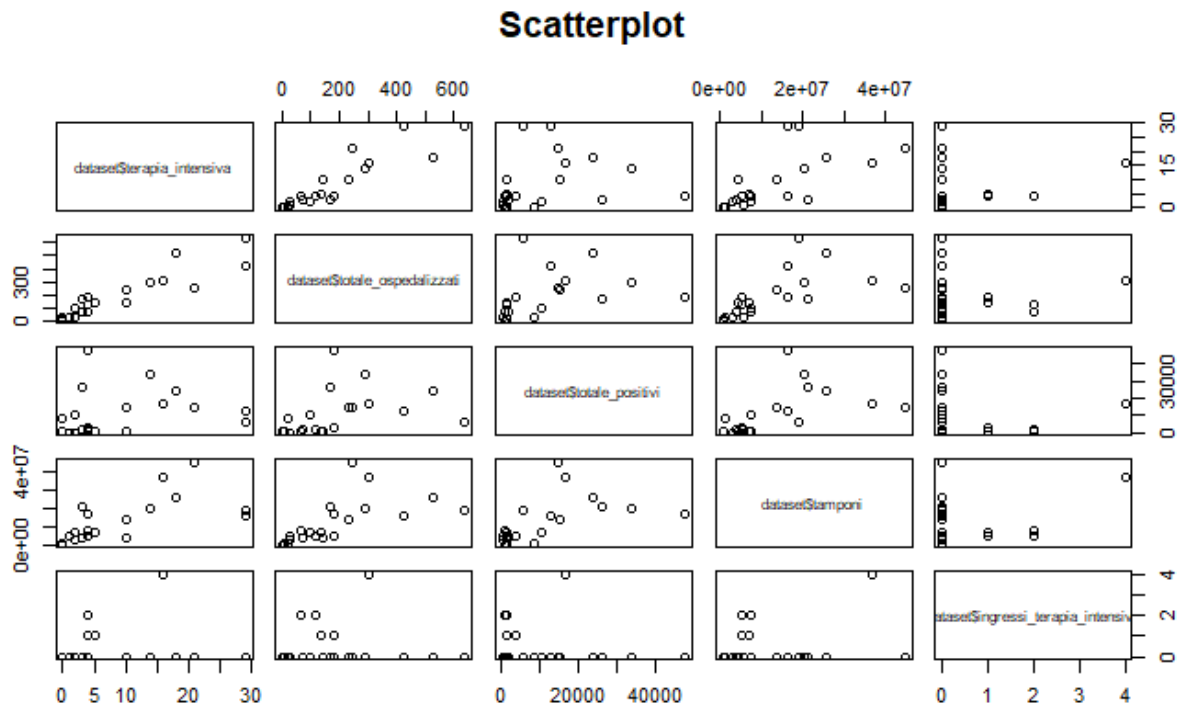
- negativa, se l'aumento di valore di una variabile implica la diminuzione di valore dell'altra.

Questo tipo di relazione può essere immediatamente riconosciuto grazie a uno scatter plot delle due variabili. Se la nuvola di punti mostrata è poco dispersa e si raggruppa attorno a una retta crescente, allora esiste una relazione lineare positiva. Se si raggruppa attorno a una retta decrescente allora esiste una relazione lineare negativa. Il grado di intensità della relazione dipende anche da quanto la nuvola di punti è fitta intorno alla retta. Chiameremo d'ora in poi retta di regressione la retta attorno alla quale si attornia la nuvola di punti.

Di seguito si ha il codice utilizzato per generare la matrice di scatter plot in R:

```
covid <- data.frame(dataset$terapia_intensiva,dataset$totale_ospedalizzati,dataset$totale_positivi,dataset$tamponi,dataset$ingr
essi_terapia_intensiva)
pairs(~ dataset$terapia_intensiva + dataset$totale_ospedalizzati + dataset$totale_positivi + dataset$tamponi +
dataset$ingressi_terapia_intensiva, data = covid, main="Scatterplot")
```

Ecco invece la matrice dei diagrammi di dispersione:



In questa immagine è evidente la presenza di alcune correlazioni abbastanza marcate tra variabili quantitative. Secondo lo scopo del

presente documento, si è deciso di focalizzarsi sullo studio delle relazioni tra il numero totale di persone ospedalizzate ed il numero totale di persone in terapia intensiva.

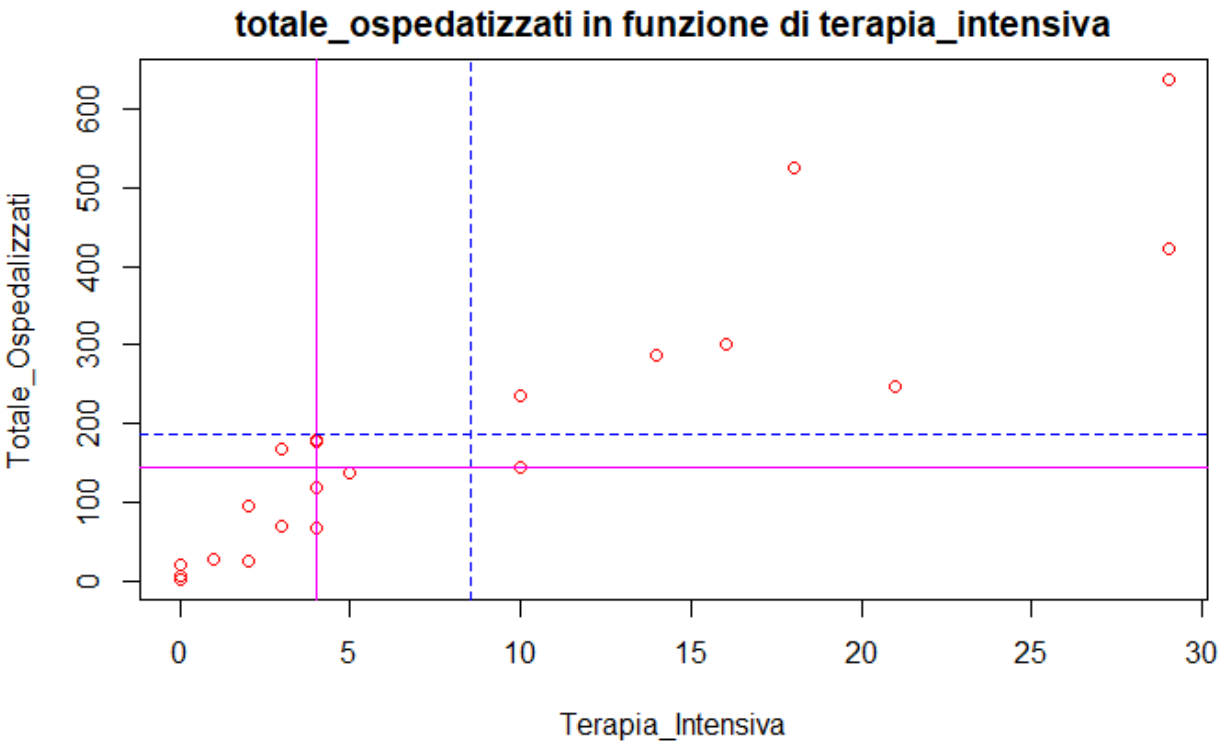
Prima di procedere allo studio vero e proprio, si inseriscono di seguito i dati numerici presi dal dataset.

denominazione_regione	terapia_intensiva	totale_ospedalizzati
Abruzzo	2	94
Basilicata	0	19
Calabria	10	143
Campania	14	287
Emilia-Romagna	29	639
Friuli Venezia Giulia	4	66
Lazio	18	527
Liguria	5	136
Lombardia	21	248
Marche	3	69
Molise	0	6
P.A. Bolzano	1	28
P.A. Trento	2	24
Piemonte	3	167
Puglia	10	235
Sardegna	4	179
Sicilia	29	424
Toscana	4	178
Umbria	4	119
Valle d'Aosta	0	2
Veneto	16	301

Si ribadisce che la media del campione “terapia_intensiva” è 8.5 e la mediana è 4. La media del campione “totale_ospedalizzati” è 185.28 e la mediana 143.

Di seguito viene mostrato un plot in cui vengono relazionati i dati tra il numero totale di persone ospedalizzate ed il numero totale di persone in terapia intensiva, contenente anche media e mediana dei dati. Questo viene generato dal seguente codice R:

```
plot(dataset$terapia_intensiva, dataset$totale_ospedalizzati, main="totale_ospedatizzati in funzione di terapia_intensiva",
xlab="Terapia_Intensiva", ylab="Totale_Ospedalizzati",
col="red")
abline(v=median(dataset$terapia_intensiva), lty=1, col="magenta")
abline(v=mean(dataset$terapia_intensiva), lty=2, col="blue")
abline(h=median(dataset$totale_ospedalizzati), lty=1, col="magenta")
abline(h=mean(dataset$totale_ospedalizzati), lty=2, col="blue")
legend(40, 700, c("Mediana", "Media"), pch=0, col=c("magenta", "blue"), cex=0.6)
```



Si può notare che i dati sembrano essere posizionati intorno ad una retta scendente e ciò induce a pensare che esista una correlazione lineare positiva tra le variabili.

4.2 Covarianza e Correlazione Campionaria

Per essere certi della reale presenza di una relazione tra le variabili X e Y e per ottenere precisione informazioni rispetto la natura (positiva o negativa) della relazione, si è previsto il calcolo di covarianza e coefficiente di correlazione.

La covarianza è un numero che misura il grado di dipendenza tra la variabile indipendente X e la variabile dipendente Y.

$$C_{xy} = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})$$

Si possono verificare i seguenti casi:

- covarianza positiva, ossia la variabile dipendente è correlata positivamente alla variabile indipendente;

- covarianza negativa, ossia la variabile dipendente è correlata negativamente alla variabile indipendente;
- covarianza nulla, ossia le variabili non sono correlate.

Il coefficiente di correlazione permette di ottenere ancora più informazioni rispetto la dipendenza di una variabile dipendente da una variabile Y indipendente X.

$$r_{xy} = \frac{C_{xy}}{s_x s_y}$$

Il coefficiente di correlazione assume un valore appartenente all'intervallo [− 1, 1]. Si possono verificare i seguenti casi:

- $r_{xy} = 1$, ossia la nuvola di punti giace esclusivamente su una retta ascendente xy = 1 (correlazione positiva perfetta);
- $r_{xy} \in (0,1)$, ossia la nuvola di punti è addensata a una retta interpolante ascendente $xy \in$ (0, 1) (correlazione positiva);
- $r_{xy} = 0$, ossia non esiste correlazione tra le variabili considerate;
- $r_{xy} \in (-1,0)$, ovvero la nuvola di punti è addensata attorno una retta interpolante discendente (correlazione negativa);
- $r_{xy} = -1$, ossia la nuvola di punti giace esclusivamente su una retta discendente (correlazione negativa perfetta).

La covarianza e la correlazione campionaria possono essere calcolate in R rispettivamente tramite le formule cov(x, y) e cor(x, y).

Calcoliamo la covarianza e la correlazione tramite il seguente codice in R:

```
cov(dataset$terapia_intensiva,dataset$totale_ospedalizzati)
cor(dataset$terapia_intensiva,dataset$totale_ospedalizzati)
```

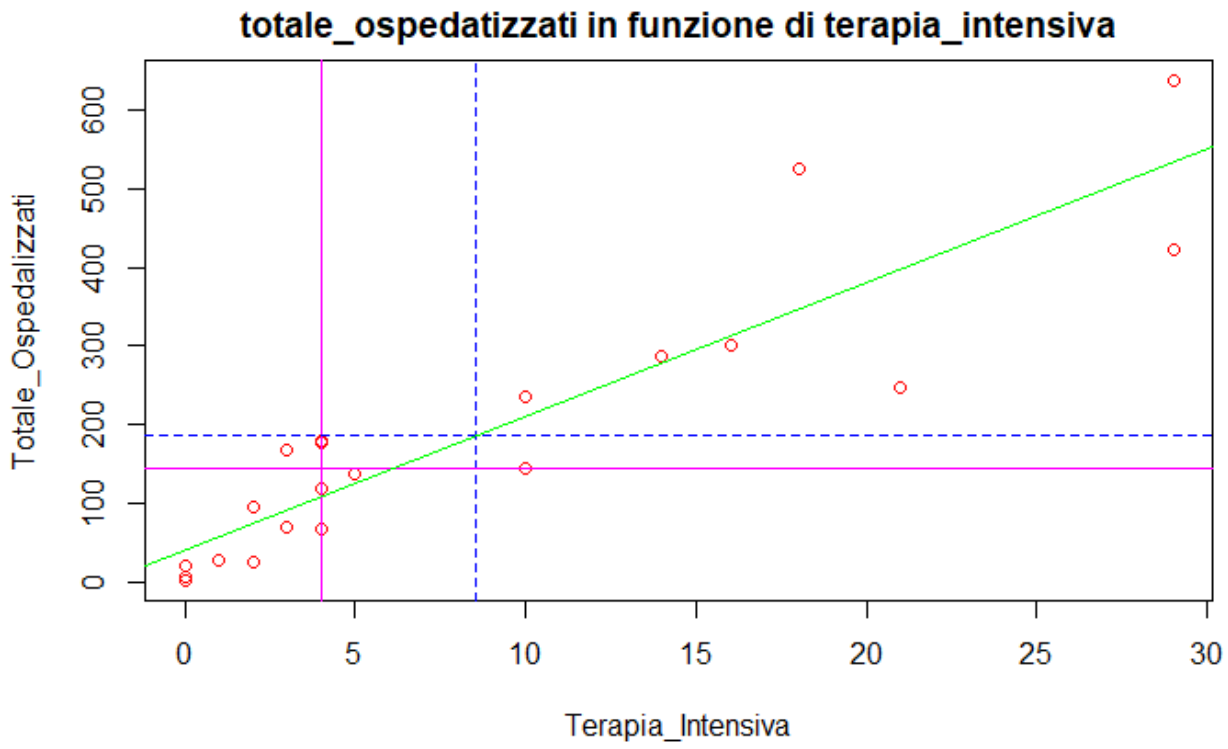
```
[1] 1439.443
[1] 0.9050261
```

Osserviamo dunque che la covarianza assume un valore molto alto, dunque il numero totale di ospedalizzati ha una forte correlazione positiva

con il numero totale di terapie intensive. A testimonianza di ciò, si nota come infatti il coefficiente di correlazione abbia quindi un valore molto vicino all'1.

Il passo successivo consiste nel realizzare il grafico contenente la retta di regressione che interpola i punti mediante il seguente codice in R:

```
plot(dataset$terapia_intensiva, dataset$totale_ospedalizzati , main="totale_ospedatizzati in funzione di terapia_intensiva",
xlab="Terapia_Intensiva", ylab="Totale_Ospedalizzati",
col="red")
abline(v=median(dataset$terapia_intensiva), lty=1, col="magenta")
abline(v=mean(dataset$terapia_intensiva), lty=2, col="blue")
abline(h=median(dataset$totale_ospedalizzati), lty=1, col="magenta")
abline(h=mean(dataset$totale_ospedalizzati), lty=2, col="blue")
abline(lm(dataset$totale_ospedalizzati ~ dataset$terapia_intensiva), col="green")
legend(40, 700, c("Mediana", "Media"), pch=0, col=c("magenta", "blue"), cex=0.6)
```



Si nota subito che il punto della media si trova sulla retta di regressione ascendente: ciò sta a significare che la retta rappresenta al meglio il legame tra X e Y ed a testimonianza di ciò, difatti, si ha il coefficiente di correlazione campionario uguale a 0. 90, molto vicino a 1. Quest'ultimo risultato indica una correlazione (negativa) molto forte.

4.3 Regressione Lineare Semplice

Dopo aver calcolato covarianza e correlazione, si passa al calcolo di α e β per scrivere l'equazione della retta:

$$Y = \alpha + \beta X$$

Dove:

- Abbiamo α che è l'*intercetta*: $\alpha = \bar{y} - \beta\bar{x}$;
- Mente β è il *coefficiente angolare* della retta: $\beta = \frac{s_y}{s_x} r_{xy}$;
- Y è la variabile dipendente;
- X è la variabile indipendente.

Tramite il seguente codice in R è possibile calcolare l'intercetta ed il coefficiente angolare:

```
beta <- sd(dataset$totale_ospedalizzati) / sd(dataset$terapia_intensiva) *
cor(dataset$terapia_intensiva,dataset$totale_ospedalizzati)
alpha <- mean(dataset$totale_ospedalizzati) - beta * mean(dataset$terapia_intensiva)
c(alpha, beta)
```

Output:

```
[1] 40.01861 17.04251
```

I coefficienti possono essere alternativamente calcolati in R tramite la funzione *lm(y~x)*. 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.

Quindi, alpha e beta possono essere calcolati in maniera più rapida utilizzando quest'altro codice in R:

```
lm ( dataset$totale_ospedalizzati ~ dataset$terapia_intensiva)
```

Output:

```
Call:
lm(formula = dataset$totale_ospedalizzati ~ dataset$terapia_intensiva)

Coefficients:
              (Intercept)  dataset$terapia_intensiva
                    40.02                      17.04
```

In conclusione, la retta di regressione avrà come equazione:

Y = 40.02 +17.04 X.

4.4 Valori Stimati

Una volta calcolati i valori dei coefficienti α e β e disegnata la retta di regressione che interpola la nuvola dei punti nel corrispondente scatter plot, è possibile osservare quanto questa retta si adatta ai punti che individuano le osservazioni.

In generale, esisteranno degli scostamenti (detti residui) tra le ordinate dei punti y_i (ovvero i valori osservati) e i corrispondenti valori stimati $\hat{y}_i$ nella retta di regressione, dove:

$$\hat{y}_i = \alpha + \beta x_i \; per \; i \in \{1, 2, \dots, n\}$$

Data la retta di regressione lineare dei punti costituiti da osservazioni di X e Y, si verifica che la media campionaria dei valori osservati y_i coincide con la media campionaria dei valori stimati $\hat{y}_i$ ella suddetta retta di regressione; possiamo verificare quanto detto dal seguente codice in R:

```
stime <- fitted(lm(dataset$totale_ospedalizzati ~ dataset$terapia_intensiva))
c(mean(stime),mean(dataset$totale_ospedalizzati))
```

Output:

```
[1] 185.2857 185.2857
```

Come si vede, la media è esattamente uguale.
In R, per il calcolo dei valori stimati si utilizzano le seguenti righe di codice:

```
fitted (lm(dataset$totale_ospedalizzati ~ dataset$terapia_intensiva))
```

Output:

```
       1        2        3        4        5        6        7        8        9       10       11       12
74.10363 40.01861 210.44371 278.61375 534.25140 108.18865 346.78379 125.23116 397.91132 91.14614 40.01861 57.06112
      13       14       15       16       17       18       19       20       21
74.10363 91.14614 210.44371 108.18865 534.25140 108.18865 108.18865 40.01861 312.69877
```

4.5 Residui

I valori residui mostrano gli scostamenti dei valori osservati dai valori stimati nella retta di regressione. Questi si indicano nel seguente modo:

$$E_i = y_i - \hat{y_i} = y_i - (\alpha + \beta x_i)\ per\ i \in \{1, 2, \dots, n\}$$

Data la retta di regressione lineare dei punti costituiti da osservazioni di X e Y, si verifica che la media dei residui è nulla. Questo perché gli scostamenti si compensano tra loro. Per calcolare i residui in R si usa la funzione ***resid(lm(y~x))***:

```
resid(lm(dataset$totale_ospedalizzati ~ dataset$terapia_intensiva))
```

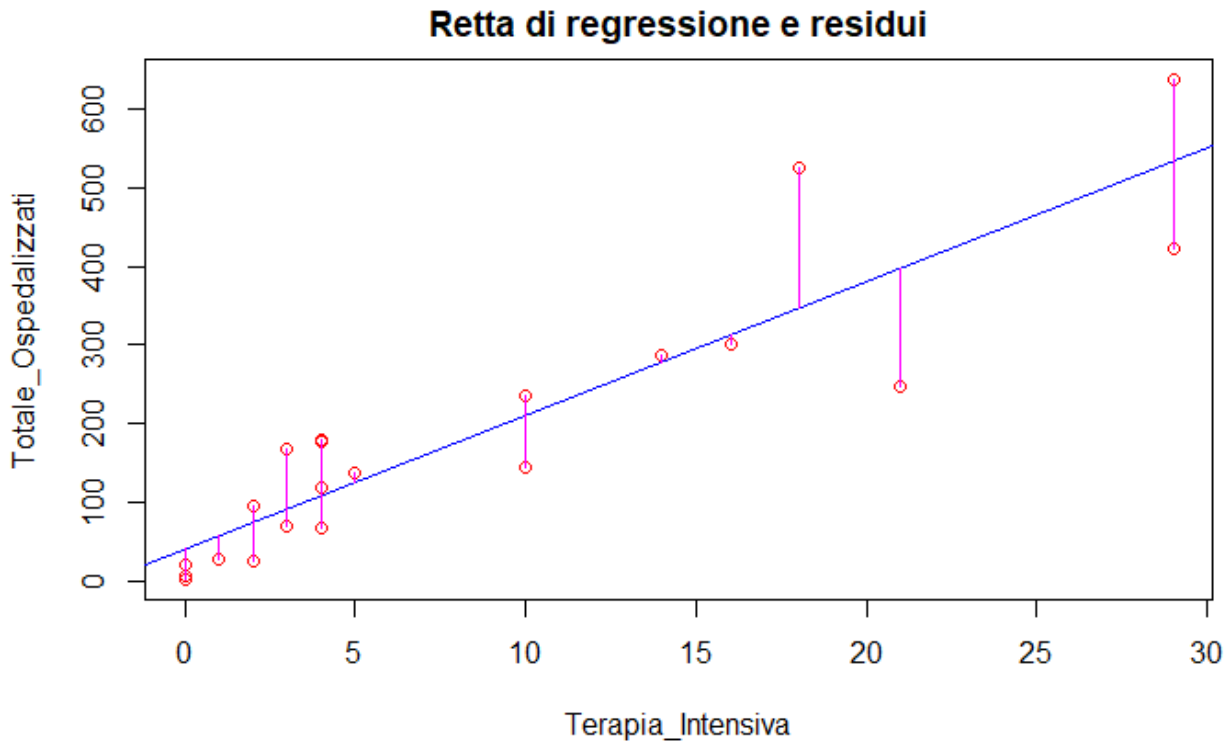
Output:

```
        1         2         3         4         5         6         7         8         9        10
 19.896375 -21.018605 -67.443705   8.386255 104.748605 -42.188645 180.216215  10.768845 -149.911315 -22.146135
       11        12        13        14        15        16        17        18        19        20
-34.018605 -29.061115 -50.103625 75.853865  24.556295  70.811355 -110.251395  69.811355  10.811355 -38.018605
       21
-11.698765
```

I residui possono anche essere rappresentati graficamente:

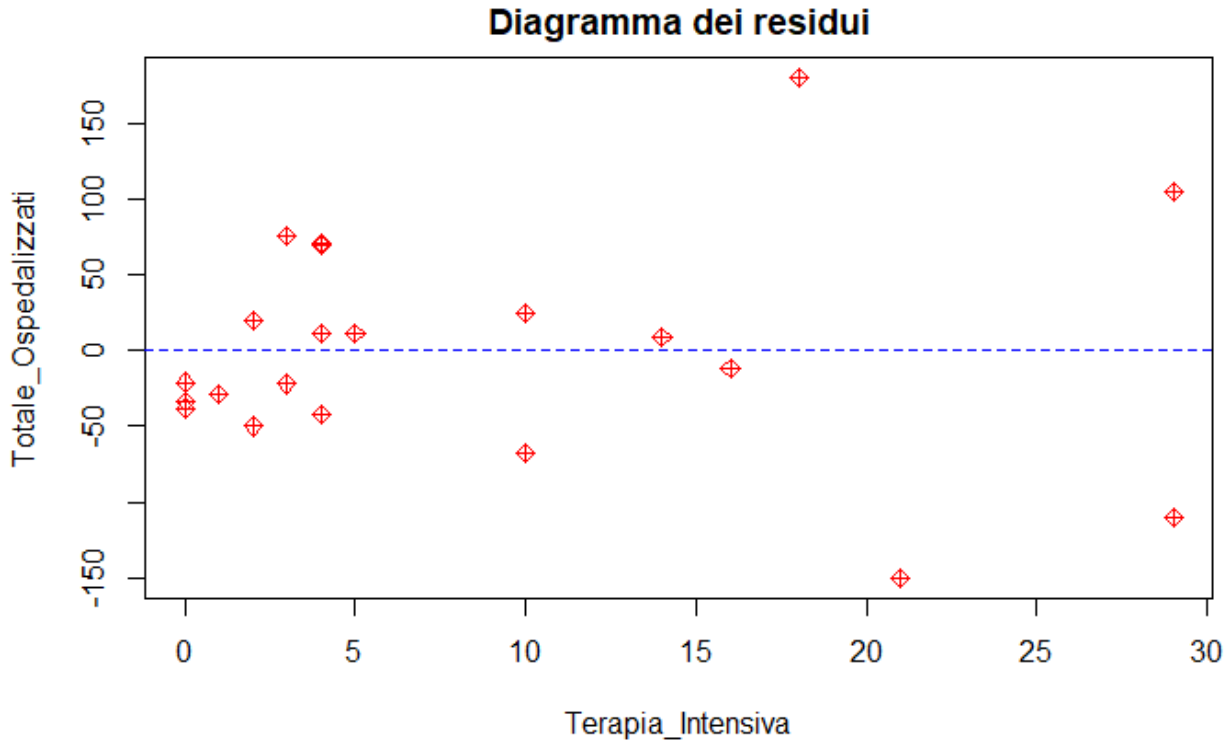
- Tracciando dei segmenti verticali che congiungono i valori stimati (sulla retta di regressione) e i valori osservati y_i, usando:

```
segmentiVerticaliResidui <- function(x, y) {
plot(x, y, main="Retta di regressione e residui", xlab="Terapia_Intensiva", ylab="Totale_Ospedalizzati",
col="red")
abline (lm(y~x), col=" blue ")
stime <- fitted(lm(y~x))
segments(x, stime, x, y, col="magenta")
}
segmentiVerticaliResidui(dataset$terapia_intensiva,dataset$totale_ospedalizzati)
```



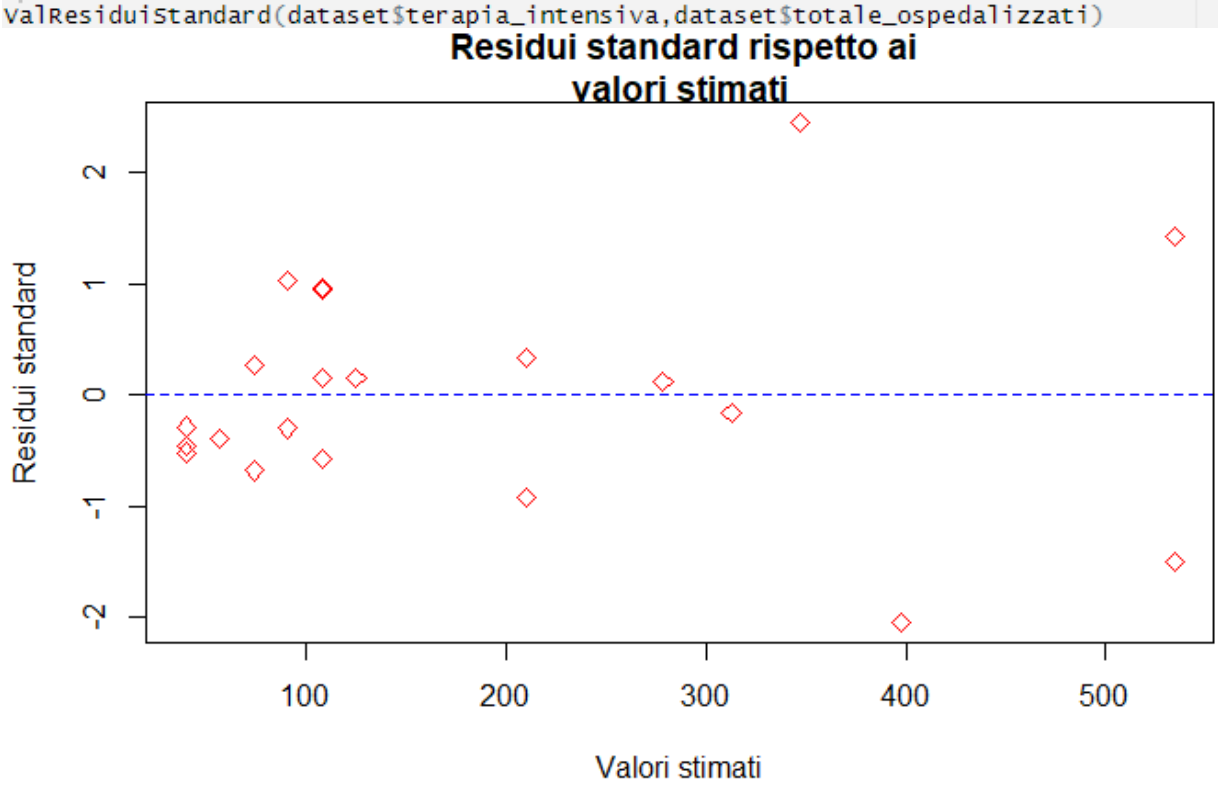
- Rappresentando i valori dei residui E_i rispetto alle osservazioni x_i, usando

```
valResiduiValX <- function(x,y) {
residui <- resid ( lm ( y ~ x ) )
plot ( x , residui , main = " Diagramma dei residui " , xlab = "Terapia_Intensiva" , ylab = "Totale_Ospedalizzati" ,
pch =9 , col = "red ")
abline (h =0 , col =" blue " , lty =2)
}
valResiduiValX(dataset$terapia_intensiva,dataset$totale_ospedalizzati)
```



- Rappresentando i residui standardizzati rispetto ai valori stimati, usando

```
valResiduiStandard <- function(x,y) {
stime <- fitted(lm(y~x))
residui <- resid(lm(y~x))
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)
}
valResiduiStandard(dataset$terapia_intensiva,dataset$totale_ospedalizzati)
```



4.6 Coefficiente di Determinazione

Poiché si è interessati a vedere quanto la retta si adatta ai dati, si stabilisce il coefficiente di determinazione.

Il coefficiente di determinazione è un indicatore di adattamento della retta di regressione ai dati, ed è il rapporto tra la varianza dei valori stimati nella retta di regressione e la varianza dei valori osservati. Questo coefficiente appartiene sempre all'intervallo [0, 1]: quanto più è maggiore tanto più i dati stimati sono rappresentativi dei dati osservati.

Il coefficiente di determinazione può essere calcolato in due modi:

```
coefficienteDeterminazione <- function(x, y) {
firstMethod <- (cor(x, y))^2
secondMethod <- summary(lm(y~x))$r.square
c(firstMethod, secondMethod)
}
coefficienteDeterminazione(dataset$terapia_intensiva,dataset$totale_ospedalizzati)
```

Output:

```
[1] 0.8190723 0.8190723
```

Con le verifiche fatte finora, quindi, possiamo affermare che è presente una forte dipendenza tra il numero totale di persone ospedalizzate ed il numero totale di persone in terapia intensiva.

4.7 Studio sulla Regressione Lineare Multipla del Medesimo Oggetto

Nella sezione precedente si è evidenziata la regressione lineare semplice, mentre nella corrente si andrà ad approfondire la regressione lineare multipla.

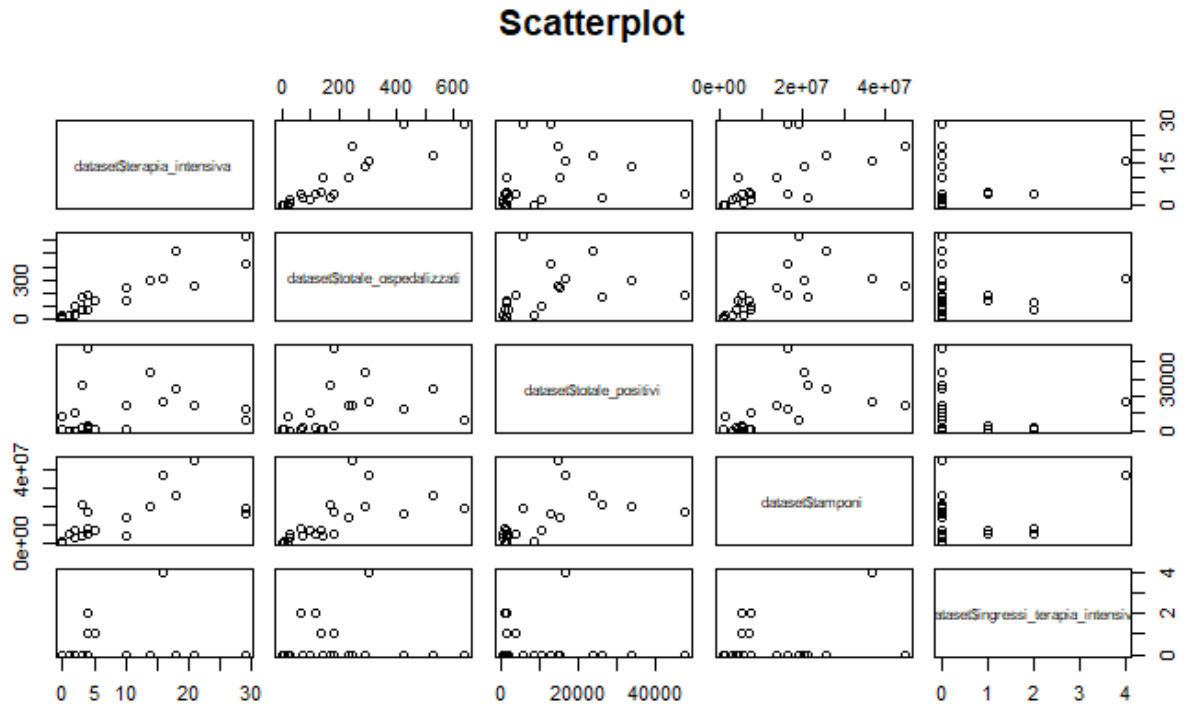
Questo nuovo modello viene utilizzato per spiegare la relazione tra una variabile quantitativa dipendente Y ed una serie di variabili quantitative indipendenti $X_1, X_2, \dots, X_p$.

Una regressione lineare multipla si esprime nel seguente modo:

$$Y = \alpha + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p$$

Dove α è l'intercetta, mentre le β_i sono i regressori.

Innanzitutto, ci generiamo tramite la funzione **pairs()** di R, un'unica finestra grafica contenente scatter plot multipli:



Successivamente viene calcolata la covarianza e correlazione di un data frame (contenente i dati del grafico soprastante) contenente n osservazioni:

```r
```{r}
getCovMultipla <- function() {
  covid <- data.frame(dataset$terapia_intensiva,dataset$totale_ospedalizzati,dataset$totale_positivi,dataset$tamponi,dataset$in
gressi_terapia_intensiva)
  cov(covid)
}
getCovMultipla()
```
```

```
                                dataset.terapia_intensiva dataset.totale_ospedalizzati dataset.totale_positivi
dataset.terapia_intensiva                    8.446190e+01                 1.439443e+03           2.808526e+04
dataset.totale_ospedalizzati                 1.439443e+03                 2.995061e+04           8.379838e+05
dataset.totale_positivi                      2.808526e+04                 8.379838e+05           1.666652e+08
dataset.tamponi                              7.467306e+07                 1.301447e+09           8.478381e+10
dataset.ingressi_terapia_intensiva           1.880952e-01                 1.807143e+00          -1.677164e+03
                                dataset.tamponi dataset.ingressi_terapia_intensiva
dataset.terapia_intensiva          7.467306e+07                      1.880952e-01
dataset.totale_ospedalizzati       1.301447e+09                      1.807143e+00
dataset.totale_positivi            8.478381e+10                     -1.677164e+03
dataset.tamponi                    1.449247e+14                      2.875014e+06
dataset.ingressi_terapia_intensiva 2.875014e+06                      1.061905e+00
```

Si nota che la matrice delle covarianze contiene sulla diagonale principale la varianza delle singole colonne del data frame.

```r
getCovMultipla <- function() {
  covid <- data.frame(dataset$terapia_intensiva,dataset$totale_ospedalizzati,dataset$totale_positivi,dataset$tamponi,dataset$ingressi_terapia_intensiva)
  cor(covid)
}
getCovMultipla()
```

```
                                dataset.terapia_intensiva dataset.totale_ospedalizzati dataset.totale_positivi
dataset.terapia_intensiva                      1.00000000                   0.90502615              0.2367149
dataset.totale_ospedalizzati                   0.90502615                   1.00000000              0.3750683
dataset.totale_positivi                        0.23671492                   0.37506827              1.0000000
dataset.tamponi                                0.67493568                   0.62467259              0.5455309
dataset.ingressi_terapia_intensiva             0.01986116                   0.01013321             -0.1260696
                                dataset.tamponi dataset.ingressi_terapia_intensiva
dataset.terapia_intensiva             0.6749357                        0.01986116
dataset.totale_ospedalizzati          0.6246726                        0.01013321
dataset.totale_positivi               0.5455309                       -0.12606960
dataset.tamponi                       1.0000000                        0.23175337
dataset.ingressi_terapia_intensiva    0.2317534                        1.00000000
```

Riguardo la matrice delle correlazioni, invece, essa contiene il numero 1 sulla diagonale principale. Inoltre, possiamo notare che esiste una forte correlazione negativa tra i ricoverati in terapia intensiva e le persone ospedalizzate, come avevamo già correttamente studiato precedentemente.

Nel nostro caso, utilizzeremo gli ospedalizzati totali come dipendente, mentre tutte le altre saranno variabili indipendenti.

Ricordiamo che Il modello di regressione lineare multipla con p variabili indipendenti è esprimibile attraverso l'equazione:

$$Y = \alpha + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p$$

Ricaviamo, tramite il seguente codice, l'intercetta ed i regressori:

```r
alphaBetaMultiple <- function() {
  lm(dataset$totale_ospedalizzati ~ dataset$terapia_intensiva + dataset$totale_positivi + dataset$tamponi + dataset$ingressi_terapia_intensiva)
}
alphaBetaMultiple()
```

```
Call:
lm(formula = dataset$totale_ospedalizzati ~ dataset$terapia_intensiva +
    dataset$totale_positivi + dataset$tamponi + dataset$ingressi_terapia_intensiva)

Coefficients:
                  (Intercept)  dataset$terapia_intensiva    dataset$totale_positivi
                    2.115e+01                 1.817e+01                  3.386e-03
           dataset$tamponi  dataset$ingressi_terapia_intensiva
                   -2.579e-06                 1.081e+01
```

Pertanto, il modello di regressione multipla stimato è:

$$Y = 2.115e + 01 - 1.817e + 01X_1 + 3.386e - 03X_2 - 2.579e - 06X_3 + 1.081e + 01X_4$$

Riguardo i residui, invece, è necessario ricavare, come prima, i valori stimati ottenuti mediante la regressione lineare multipla:

$$\widehat{y_i} = \alpha + \beta_1 x_{i+1} + \beta_2 x_{i+2} + \cdots + \beta_n x_{i+n} \; per \; i \in \{1, 2, \ldots, n\}$$

Successivamente, possono essere calcolati i residui:

$$E_i = y_i - \widehat{y_i} = y_i - (\alpha + \beta_1 x_{i+1} + \beta_2 x_{i+2} + \cdots + \beta_n x_{i+n}) \; per \; i \in \{1, 2, \ldots, n\}$$

Tramite la seguente funzione in R è possibile calcolare i valori stimati:

```
fitted(lm(dataset$totale_ospedalizzati ~ dataset$terapia_intensiva + dataset$totale_positivi + dataset$tamponi +
dataset$ingressi_terapia_intensiva))
```

Ecco il risultato:

```
       1        2        3        4        5        6        7        8        9       10       11       12
73.64475 46.05961 196.54900 337.65693 518.30210 98.78942 362.79492 109.09960 336.39862 71.77297 23.58956 25.96874
      13       14       15       16       17       18       19       20       21
50.79891 110.20153 218.93343 102.46056 549.47865 213.01047 107.52654 21.76171 316.20198
```

Successivamente, possono essere calcolati i residui:

```
resid(lm(dataset$totale_ospedalizzati ~ dataset$terapia_intensiva + dataset$totale_positivi + dataset$tamponi +
dataset$ingressi_terapia_intensiva))
```

Ecco l'output:

```
        1         2         3         4         5         6         7         8         9        10
 20.355253 -27.059608 -53.549004 -50.656932 120.697897 -32.789424 164.205081  26.900401 -88.398624  -2.772969
       11        12        13        14        15        16        17        18        19        20
-17.589558   2.031264 -26.798906  56.798470  16.066566  76.539444 -125.478646 -35.010474  11.473464 -19.761714
       21
-15.201982
```

Si conclude osservando il valore del coefficiente di determinazione:

```
summary(lm(dataset$totale_ospedalizzati ~ dataset$terapia_intensiva + dataset$totale_positivi + dataset$tamponi +
dataset$ingressi_terapia_intensiva))$r.square
```

Output:

```
[1] 0.8567136
```

Esso è pari a 0.8567136.

5. Analisi dei Cluster

L'analisi dei cluster è una metodologia che permette di raggruppare in sottoinsiemi (detti cluster) entità appartenenti a un insieme più ampio. L'insieme originario delle entità su cui si svolge l'analisi per ricavare i cluster non è vincolato ad alcuna restrizione; infatti, esso può contenere variabili, individui, osservazioni, dati, misure, ecc... 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.

Ciò significa che questa analisi ha lo scopo di suddividere in gruppi le osservazioni in modo da garantire due proprietà:

- alta omogeneità all'interno dei gruppi, ossia bisogna che sia presente un alto grado di associazione tra membri dello stesso gruppo;
- alta eterogeneità tra gruppi diversi, ossia avere un basso grado di associazione tra membri di gruppi differenti.

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.

5.1 Distanza e Similarità

Le misure metriche di somiglianza sono soprattutto basate sulle funzioni distanza tra i vettori delle caratteristiche. Una funzione a valori reali $d(X_i, X_j)$, è detta funzione distanza se e soltanto se essa soddisfa le seguenti condizioni:

- $d(X_i, X_j) = 0$, se e solo se $X_i = X_j$, con $X_i \; e \; X_j \; in \; E_p$ (Questa proprietà implica che X_i è a distanza zero da sé stesso e che ogni due punti a distanza nulla devono essere identici);
- $d(X_i, X_j) \geq 0, \forall X_i, X_j \; in \; E_p$ (La seconda proprietà afferma che la funzione distanza è non negativa);
- $d(X_i, X_j) = d(X_j, X_i), \forall X_i, X_j \; in \; E_p$ (La terza proprietà 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);
- $d(X_i, X_j) \leq \; d(X_i, X_k) + d(X_k, X_j), \forall X_i, X_j, X_k \; in \; E_p$ (L'ultima proprietà, nota come disuguaglianza triangolare, richiede che la distanza tra X_ie X_j debba essere sempre minore o uguale della somma delle distanze di ognuno dei due vettori considerati da qualunque altro terzo vettoreX_k.

Generalmente, tutte le distanze vengono inserite in una matrice simmetrica D:

$$D = \begin{pmatrix} 0 & d_{12} & \dots & d_{1n} \\ d_{21} & 0 & \dots & d_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ d_{n1} & d_{n2} & \dots & 0 \end{pmatrix}$$

Dove $d_{ij} = d(X_i, X_j)$.

La matrice delle distanze D può essere calcolata in R tramite la funzione

dist(X, method="euclidean", diag=FALSE, upper=FALSE)

dove:

- Xrappresenta una matrice numerica o un data frame;
- method seleziona la misura di distanza da utilizzare (che di default è già euclidean);

- **diag** è posta uguale a TRUE 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 `e FALSE).

Tale matrice in R può essere ottenuta eseguendo il seguente codice:

```
getDataFrame <- function() {
  covid <- data.frame(dataset$terapia_intensiva,dataset$totale_ospedalizzati,dataset$totale_positivi,dataset$tamponi,dataset$in
gressi_terapia_intensiva)
  row.names(covid) <- c(dataset$denominazione_regione)
  covid = scale(covid)
  covid
}
getDistanza <- function() {
  df <- getDataFrame()
  dist(df, method="euclidean", diag=TRUE, upper=TRUE)
}
getDistanza()
```

Output:

	Abruzzo	Basilicata	Calabria	Campania	Emilia-Romagna	Friuli Venezia Giulia	Lazio
Abruzzo	0.00000000	0.71518238	1.18249003	2.72052994	4.43154791	2.09313413	3.57436573
Basilicata	0.71518238	0.00000000	1.43220681	3.33085562	5.00334937	2.15407115	4.25310724
Calabria	1.18249003	1.43220681	0.00000000	3.00086664	3.76275773	2.11507359	3.46655371
Campania	2.72052994	3.33085562	3.00086664	0.00000000	3.39378525	3.76773074	1.70378828
Emilia-Romagna	4.43154791	5.00334937	3.76275773	3.39378525	0.00000000	4.81612319	2.03784485
Friuli Venezia Giulia	2.09313413	2.15407115	2.11507359	3.76773074	4.81612319	0.00000000	4.32362602
Lazio	3.57436573	4.25310724	3.46655371	1.70378828	2.03784485	4.32362602	0.00000000
Liguria	1.27324066	1.48871909	1.13444864	3.21301532	4.17067287	1.05928332	3.70095669
Lombardia	3.85214453	4.49928173	3.77086915	2.64569780	3.29316422	4.35219563	2.39017003
Marche	0.76038357	0.70531269	0.87482104	3.33387600	4.53920959	1.97191405	4.00672215
Molise	1.04671997	0.55206297	1.37500189	3.73093099	5.07841160	2.09769148	4.50952796
P.A. Bolzano	0.88913121	0.72763729	1.19156002	3.53365810	4.81777973	1.98832793	4.23876063
P.A. Trento	0.94851034	0.67504221	1.11627973	3.57843797	4.82094843	2.00534365	4.29584047
Piemonte	1.74402769	2.34657154	2.51997254	1.50102445	4.24304316	3.04621761	2.67870343
Puglia	1.35515055	2.01919478	1.43369971	1.63278985	3.23403503	2.57396778	2.25828880
Sardegna	1.24249718	1.49817663	1.20351988	3.09468624	4.09282902	1.20095447	3.56914085
Sicilia	3.58643509	4.13710233	2.95436839	2.45638523	1.37735065	4.10253690	1.77799632
Toscana	3.04399723	3.45884779	3.80455060	1.68883105	5.01259082	4.23518068	3.22227675
Umbria	2.08582578	2.16081427	2.05348261	3.71372458	4.65800606	0.37901037	4.20838177
Valle d'Aosta	1.10321119	0.61012731	1.39306738	3.78487004	5.10416679	2.10612892	4.55404349
Veneto	4.99711598	5.45655812	4.99788422	4.33375142	4.86560548	3.81975595	4.23891032

	Liguria	Lombardia	Marche	Molise	P.A. Bolzano	P.A. Trento	Piemonte	Puglia
Abruzzo	1.27324066	3.85214453	0.76038357	1.04671997	0.88913121	0.94851034	1.74402769	1.35515055
Basilicata	1.48871909	4.49928173	0.70531269	0.55206297	0.72763729	0.67504221	2.34657154	2.01919478
Calabria	1.13444864	3.77086915	0.87482104	1.37500189	1.19156002	1.11627973	2.51997254	1.43369971
Campania	3.21301532	2.64569780	3.33387600	3.73093099	3.53365810	3.57843797	1.50102445	1.63278985
Emilia-Romagna	4.17067287	3.29316422	4.53920959	5.07841160	4.81777973	4.82094843	4.24304316	3.23403503
Friuli Venezia Giulia	1.05928332	4.35219563	1.97191405	2.09769148	1.98832793	2.00534365	3.04621761	2.57396778
Lazio	3.70095669	2.39017003	4.00672215	4.50952796	4.23876063	4.29584047	2.67870343	2.25828880
Liguria	0.00000000	3.92667362	1.09867664	1.43275340	1.23933052	1.25316511	2.50221862	1.75303890
Lombardia	3.92667362	0.00000000	4.18811047	4.64586582	4.26534962	4.38174572	2.95572016	2.84878136
Marche	1.09867664	4.18811047	0.00000000	0.54668744	0.37210613	0.30623468	2.46594900	1.81104758
Molise	1.43275340	4.64586582	0.54668744	0.00000000	0.43707063	0.31388885	2.75745335	2.28801781
P.A. Bolzano	1.23933052	4.26534962	0.37210613	0.43707063	0.00000000	0.23793871	2.54090147	2.04367471
P.A. Trento	1.25316511	4.38174572	0.30623468	0.31388885	0.23793871	0.00000000	2.65281570	2.08640315
Piemonte	2.50221862	2.95572016	2.46594900	2.75745335	2.54090147	2.65281570	0.00000000	1.36829372
Puglia	1.75303890	2.84878136	1.81104758	2.28801781	2.04367471	2.08640315	1.36829372	0.00000000
Sardegna	0.34627129	3.99603880	1.18155407	1.51795280	1.36755485	1.37444980	2.41334987	1.66461405
Sicilia	3.46152900	2.71826161	3.75012434	4.27328551	4.03373404	4.01773437	3.38571603	2.35543261
Toscana	3.83745099	3.96693797	3.78205172	3.98094095	3.90299072	3.95239603	1.71048245	2.64114868
Umbria	0.99333916	4.43829655	1.96817597	2.12242264	2.03967541	2.03717494	3.06328416	2.50858019
Valle d'Aosta	1.45146930	4.67837803	0.57505507	0.06108311	0.45329089	0.32411251	2.81372692	2.33498123
Veneto	4.28663387	3.99165977	5.26031595	5.59135089	5.33574027	5.41523657	4.46218550	4.39258632

	Sardegna	Sicilia	Toscana	Umbria	Valle d'Aosta	Veneto
Abruzzo	1.24249718	3.58643509	3.04399723	2.08582578	1.10321119	4.99711598
Basilicata	1.49817663	4.13710233	3.45884779	2.16081427	0.61012731	5.45655812
Calabria	1.20351988	2.95436839	3.80455060	2.05348261	1.39306738	4.99788422
Campania	3.09468624	2.45638523	1.68883105	3.71372458	3.78487004	4.33375142
Emilia-Romagna	4.09282902	1.37735065	5.01259082	4.65800606	5.10416679	4.86560548
Friuli Venezia Giulia	1.20095447	4.10253690	4.23518068	0.37901037	2.10612892	3.81975595
Lazio	3.56914085	1.77799632	3.22227675	4.20838177	4.55404349	4.23891032
Liguria	0.34627129	3.46152900	3.83745099	0.99333916	1.45146930	4.28663387
Lombardia	3.99603880	2.71826161	3.96693797	4.43829655	4.67837803	3.99165977
Marche	1.18155407	3.75012434	3.78205172	1.96817597	0.57505507	5.26031595
Molise	1.51795280	4.27328551	3.98094095	2.12242264	0.06108311	5.59135089
P.A. Bolzano	1.36755485	4.03373404	3.90299072	2.03967541	0.45329089	5.33574027
P.A. Trento	1.37444980	4.01773437	3.95239603	2.03717494	0.32411251	5.41523657
Piemonte	2.41334987	3.38571603	1.71048245	3.06328416	2.81372692	4.46218550
Puglia	1.66461405	2.35543261	2.64114868	2.50858019	2.33498123	4.39258632
Sardegna	0.00000000	3.42113465	3.68673724	1.04261968	1.54435206	4.29436620
Sicilia	3.42113465	0.00000000	4.09222212	3.99279331	4.30310590	4.52774527
Toscana	3.68673724	4.09222212	0.00000000	4.20537027	4.04107722	5.09671784
Umbria	1.04261968	3.99279331	4.20537027	0.00000000	2.13345727	3.85378461
Valle d'Aosta	1.54435206	4.30310590	4.04107722	2.13345727	0.00000000	5.61903407
Veneto	4.29436620	4.52774527	5.09671784	3.85378461	5.61903407	0.00000000

Il cluster con le distanze è stato calcolato applicando il metodo euclideo.

Come possiamo notare, i valori sulla diagonale sono tutti nulli, ovvero pari a zero, mentre quelli al di sotto e quelli al di sopra risultano essere equivalenti. Quindi, il numero di distanze che è necessario conoscere

affinché sia definita la posizione di ciascuna delle n variabili rispetto alle rimanenti n − 1 è n(n − 1)/2. Ciò significa che è sufficiente considerare la matrice triangolare al di sopra o al di sotto della diagonale principale di D.

Infatti, potremmo ottimizzare ulteriormente l'output precedente applicando questa modifica al codice precedente:

```
getDataFrame <- function () {
  covid <- data.frame(dataset$terapia_intensiva,dataset$totale_ospedalizzati,dataset$totale_positivi,dataset$tamponi,dataset$in
gressi_terapia_intensiva)
  row.names(covid) <- c(dataset$denominazione_regione)
  covid = scale(covid)
  covid
}
getDistanza <- function() {
  df <- getDataFrame()
  dist(df, method="euclidean", diag=FALSE, upper=FALSE)
}
getDistanza()
```

Output:

```
                    Abruzzo Basilicata   Calabria   Campania Emilia-Romagna Friuli Venezia Giulia      Lazio
Basilicata        0.71518238
Calabria          1.18249003 1.43220681
Campania          2.72052994 3.33085562 3.00086664
Emilia-Romagna    4.43154791 5.00334937 3.76275773 3.39378525
Friuli Venezia Giulia 2.09313413 2.15407115 2.11507359 3.76773074     4.81612319
Lazio             3.57436573 4.25310724 3.46655371 1.70378828     2.03784485           4.32362602
Liguria           1.27324066 1.48871909 1.13444864 3.21301532     4.17067287           1.05928332 3.70095669
Lombardia         3.85214453 4.49928173 3.77086915 2.64569780     3.29316422           4.35219563 2.39017003
Marche            0.76038357 0.70531269 0.87482104 3.33387600     4.53920959           1.97191405 4.00672215
Molise            1.04671997 0.55206297 1.37500189 3.73093099     5.07841160           2.09769148 4.50952796
P.A. Bolzano      0.88913121 0.72763729 1.19156002 3.53365810     4.81777973           1.98832793 4.23876063
P.A. Trento       0.94851034 0.67504221 1.11627973 3.57843797     4.82094843           2.00534365 4.29584047
Piemonte          1.74402769 2.34657154 2.51997254 1.50102445     4.24304316           3.04621761 2.67870343
Puglia            1.35515055 2.01919478 1.43369971 1.63278985     3.23403503           2.57396778 2.25828880
Sardegna          1.24249718 1.49817663 1.20351988 3.09468624     4.09282902           1.20095447 3.56914085
Sicilia           3.58643509 4.13710233 2.95436839 2.45638523     1.37735065           4.10253690 1.77799632
Toscana           3.04399723 3.45884779 3.80455060 1.68883105     5.01259082           4.23518068 3.22227675
Umbria            2.08582578 2.16081427 2.05348261 3.71372458     4.65800606           0.37901037 4.20838177
Valle d'Aosta     1.10321119 0.61012731 1.39306738 3.78487004     5.10416679           2.10612892 4.55404349
Veneto            4.99711598 5.45655812 4.99788422 4.33375142     4.86560548           3.81975595 4.23891032
```

	Liguria	Lombardia	Marche	Molise	P.A. Bolzano	P.A. Trento	Piemonte	Puglia
Basilicata								
Calabria								
Campania								
Emilia-Romagna								
Friuli Venezia Giulia								
Lazio								
Liguria								
Lombardia	3.92667362							
Marche	1.09867664	4.18811047						
Molise	1.43275340	4.64586582	0.54668744					
P.A. Bolzano	1.23933052	4.26534962	0.37210613	0.43707063				
P.A. Trento	1.25316511	4.38174572	0.30623468	0.31388885	0.23793871			
Piemonte	2.50221862	2.95572016	2.46594900	2.75745335	2.54090147	2.65281570		
Puglia	1.75303890	2.84878136	1.81104758	2.28801781	2.04367471	2.08640315	1.36829372	
Sardegna	0.34627129	3.99603880	1.18155407	1.51795280	1.36755485	1.37444980	2.41334987	1.66461405
Sicilia	3.46152900	2.71826161	3.75012434	4.27328551	4.03373404	4.01773437	3.38571603	2.35543261
Toscana	3.83745099	3.96693797	3.78205172	3.98094095	3.90299072	3.95239603	1.71048245	2.64114868
Umbria	0.99333916	4.43829655	1.96817597	2.12242264	2.03967541	2.03717494	3.06328416	2.50858019
Valle d'Aosta	1.45146930	4.67837803	0.57505507	0.06108311	0.45329089	0.32411251	2.81372692	2.33498123
Veneto	4.28663387	3.99165977	5.26031595	5.59135089	5.33574027	5.41523657	4.46218550	4.39258632

	Sardegna	Sicilia	Toscana	Umbria	Valle d'Aosta
Basilicata					
Calabria					
Campania					
Emilia-Romagna					
Friuli Venezia Giulia					
Lazio					
Liguria					
Lombardia					
Marche					
Molise					
P.A. Bolzano					
P.A. Trento					
Piemonte					
Puglia					
Sardegna					
Sicilia	3.42113465				
Toscana	3.68673724	4.09222212			
Umbria	1.04261968	3.99279331	4.20537027		
Valle d'Aosta	1.54435206	4.30310590	4.04107722	2.13345727	
Veneto	4.29436620	4.52774527	5.09671784	3.85378461	5.61903407

Questo risultato è stato ottenuto modificando:

- diag, è stato impostato su FALSE, in questo modo non appaiono i valori nulli sulla diagonale;
- upper, è stato impostato su FALSE, in questo modo, non compaiono i valori al di sopra della diagonale, perché risulterebbero duplicati dato che già sono presenti al di sotto della diagonale.

Quindi, come possiamo notare, adesso compaiono solamente i valori non nulli e non duplicati.

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 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.

Il processo di standardizzazzione è stato eseguito grazie all'uso della funzione **scale(covid)**, dove **covid** identifica il dataset in esame.

5.2 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 permette di definire in modo quantitativo la somiglianza o differenza tra due individui I_i e I_j.

Viene fornito come risultato un valore numerico compreso tra 0 e 1, dove:

- Se 0, si intende l'assoluta assenza di similarità;
- Se 1, si intende l'assoluta presenza di somiglianza.

Per definizione 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

$$
\begin{aligned}
&(i)\ s(X_i, X_i) = 1;\\
&(ii)\ 0 \leq s(X_i, X_j) \leq 1;\\
&(iii)\ s(X_i, X_j) = s(X_j, X_i)\ \textit{per ogni}\ X_i\ e\ X_j.
\end{aligned}
$$

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:

$$S = \begin{pmatrix} 1 & s_{12} & \dots & s_{1n} \\ s_{21} & 1 & \dots & s_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ s_{n1} & s_{n2} & \dots & 1 \end{pmatrix}$$

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. Supponiamo che d_{ij} sia la distanza tra X_i e X_j, allora mediante la seguente formula

$$s_{ij} = \frac{1}{1 + d_{ij}} \qquad (i, j = 1, 2, \dots, n)$$

si può effettuare la trasformazione e quindi ottenere la misura di similarità tra X_i e X_j.

Non sempre, però, è possibile effettuare il contrario, ovvero passare da una misura di similarità ad una distanza, in quanto quella di distanza deve soddisfare anche la disuguaglianza triangolare.

5.3 Clustering tramite Metodologie Gerarchiche

Gli algoritmi di clustering che trattano la divisione gerarchica si suddividono in due categorie: gli algoritmi di tipo agglomerativi partono dalle foglie del *dendrogramma* per poi arrivare alla radice, quindi seguono in bottom-up; gli algoritmi di tipo divisivo partono dalla radice per poi arrivare alle foglie, lavorando quindi in maniera opposta rispetto agli agglomerativi, quindi in top-down.

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.

Nel dendrogramma, sulle ascisse sono riportati i livelli di distanza, mentre sulle ordinate sono riportati i singoli individui. Molti metodi di analisi gerarchica hanno un algoritmo comune che è formato da cinque step:

1. Si considera la matrice delle distanze D o la matrice di similarità S tra gli individui considerati come singoli cluster contenenti un solo individuo;

2. Si individua la coppia di cluster meno distanti (o più somiglianti) e si raggruppano in un unico cluster;

3. Si costruisce una nuova matrice di distanza (o di similarità) che risulterà ridotta di una riga e di una colonna rispetto a quella che la precede;

4. Effettuare n-1 iterazioni a partire dallo step 2, in modo da arrivare alla fine ad avere una matrice di cardinalità 2x2;

5. Rappresentare graficamente il processo di agglomerazione attraverso un dendrogramma.

I cinque passi sopra delineati, come già detto, costituiscono lo scheletro comune di molti metodi gerarchici. Le uniche differenze esistenti si riscontreranno nel passo 1 e nel passo 2.

L'analisi gerarchica di tipo agglomerativo può essere effettuata in R attraverso la funzione:

```
hclust(dataset, method = "complete")
```

Alcune delle opzioni disponibili per ***method*** sono (ricordiamo che tutti questi metodi, sono stati calcolati standardizzando i dati, la variabile dataset rappresenta il dataset standardizzato):

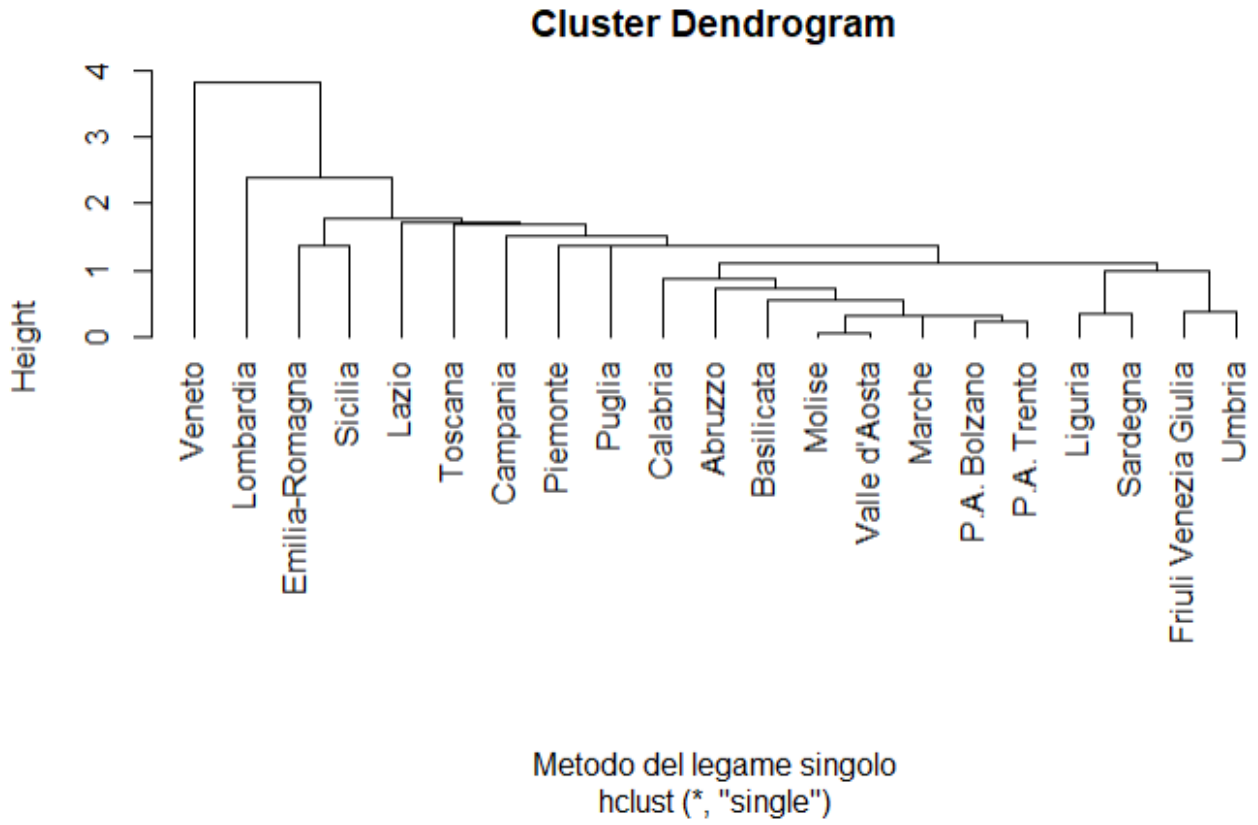
1. Metodo del legame singolo (single)

Il primo metodo è quello del legame singolo, il quale accorpa i cluster basandosi sulla distanza più piccola. Una volta fatto l'accorpamento, si ricalcolano le distanze e si riconsidera la nuova distanza minima. Si continua fino ad avere un unico cluster.

Tramite il seguente codice in R:

```
legameSingolo <- function() {
  df <- getDataFrame()
  distanze <- dist(df, method="euclidean", diag=TRUE, upper="TRUE")
  cluster <- hclust(distanze, method="single")
  plot(cluster, hang=-1, xlab="Metodo del legame singolo")
}
legameSingolo()
```

È possibile ottenere il dendrogramma generato dalle iterazioni del metodo del legame singolo:



Osservando attentamente il dendogramma, risultano esserci quattro cluster principali: il primo contiene solo il Veneto, il secondo va dalla Lombardia alla Puglia, il terzo dalla Calabria alla P. A. di Trento ed il quarto dalla Liguria all'Umbria.

È bene ribadire, però, che la suddivisione in cluster, dipende anche dal metodo utilizzato; quindi, vediamo anche gli altri metodi che cluster producono.

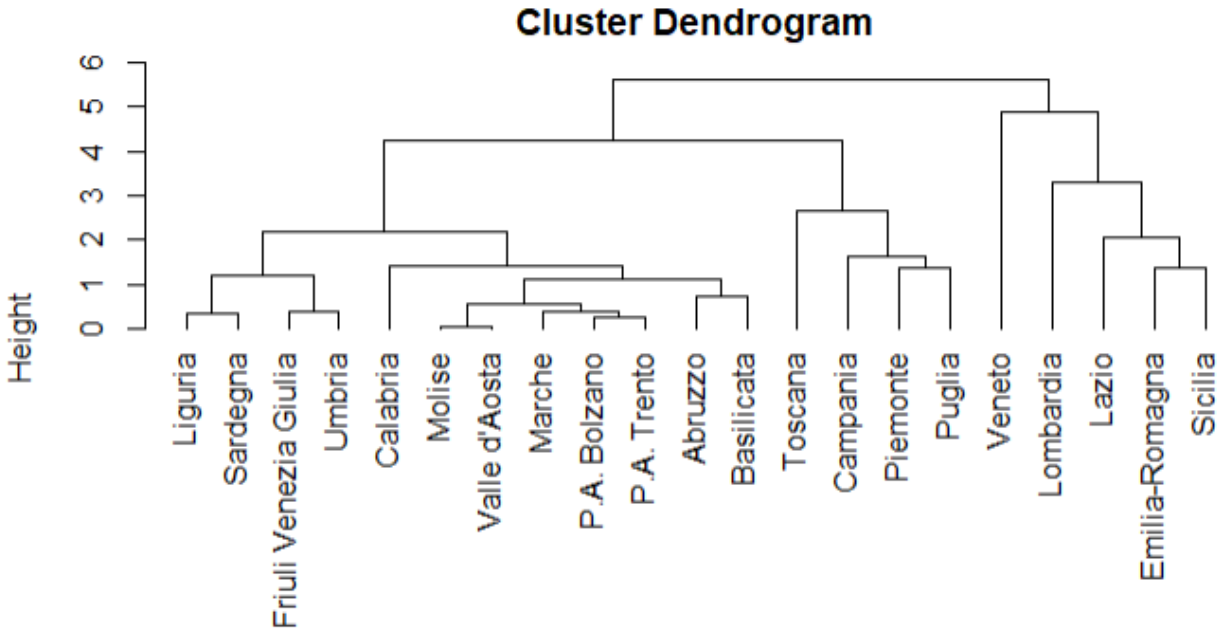
2. Metodo del legame completo (complete)

Il secondo metodo è quello del legame completo, che, a differenza del singolo, effettua un'analisi opposta, considerando le distanze massime tra gli elementi esterni al cluster.

Di seguito è possibile trovare il codice in R per tale metodo:

```
legameCompleto <- function() {
  df <- getDataFrame()
  distanze <- dist(df, method="euclidean", diag=TRUE, upper="TRUE")
  cluster <- hclust(distanze, method="complete")
  plot(cluster, hang=-1, xlab="Metodo del legame completo")
}
legameCompleto()
```

Output:



Dal seguente dendrogramma si può dedurre innanzitutto che adesso la suddivisione in cluster è cambiata rispetto al metodo singolo utilizzato precedentemente, infatti con l'utilizzo del metodo del legame completo si notano subito grandi differenze riguardo la divisione in cluster.

Quindi, tramite il metodo del legame completo, in risalto si hanno cinque cluster principali: il primo che va dalla Liguria all'Umbria, il secondo che va dalla Calabria alla Basilicata, il terzo che va dalla Toscana alla Puglia, Il quarto che contiene solo il Veneto ed il quinto che va dalla Lombardia alla Sicilia.

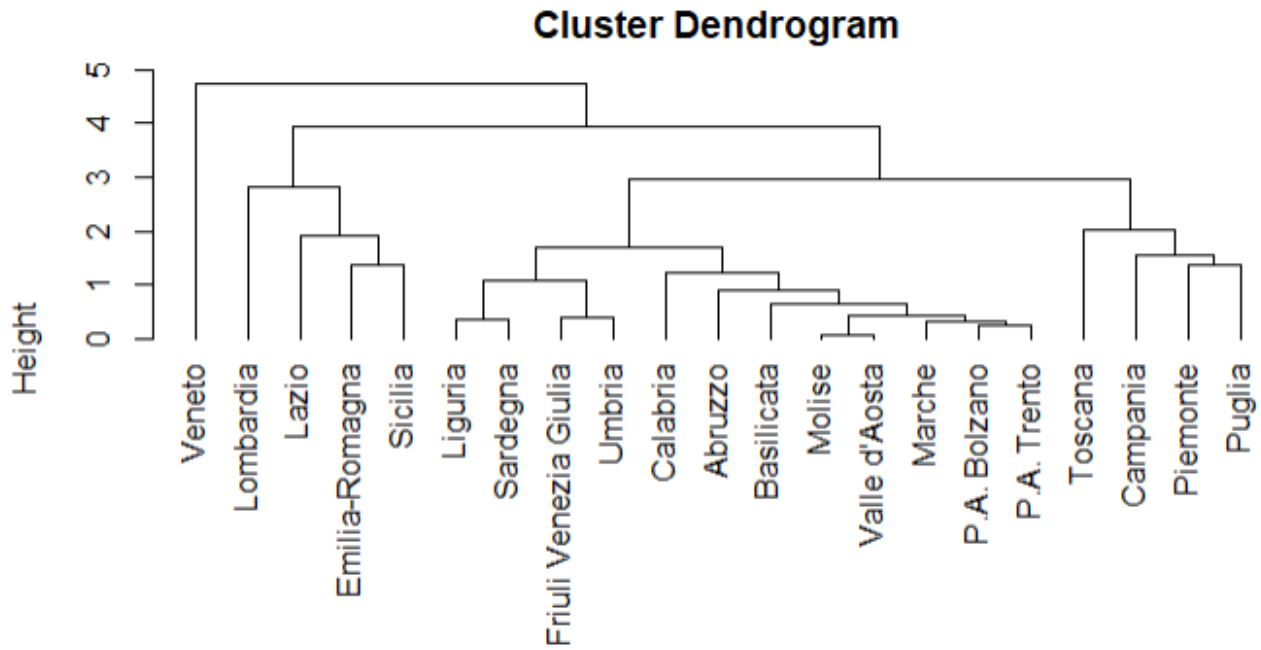
3. Metodo del legame medio (average)

Il terzo metodo pone il calcolo della nuova distanza effettuando una media tra tutte le possibili distanze tra i valori interni nel cluster appena creato e quelli all'esterno. È importante sapere che il numero di individui in ogni cluster incide molto sul calcolo delle nuove distanze, difatti cluster più grandi avranno un peso maggiore rispetto agli altri.

Di seguito è possibile trovare il codice in R per tale metodo:

```
legameMedio <- function() {
  df <- getDataFrame()
  distanze <- dist(df, method="euclidean", diag=TRUE, upper="TRUE")
  cluster <- hclust(distanze, method="average")
  plot(cluster, hang=-1, xlab="Metodo del legame medio")
}
legameMedio()
```

Output:



Quest'ultimo dendrogramma è molto simile a quello riprodotto dal metodo del legame completo: i principali cluster individuati sono gli stessi e i raggruppamenti sono molto simili.

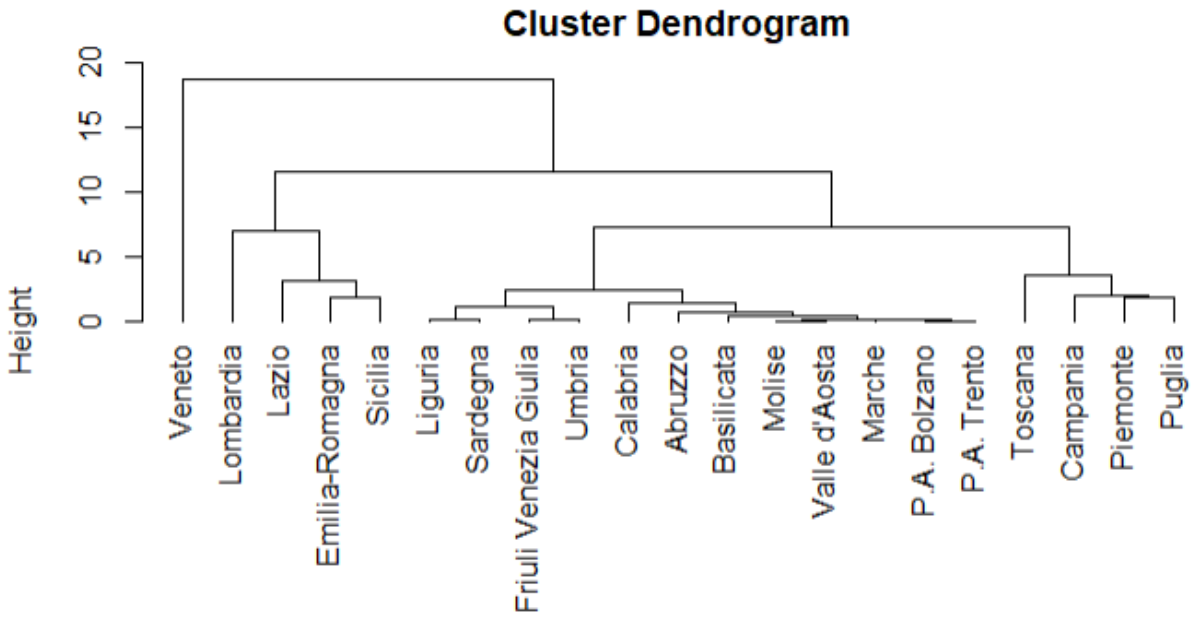
4. Metodo del centroide (centroid)

Il metodo del centroide utilizza una metrica basata sulle distanze dei centroidi dei gruppi, dove il centroide è la media campionaria tra tutti gli elementi dei due cluster in considerazione. Come per il metodo precedente, anche in questo caso è rilevante il numero degli elementi appartenenti ad un singolo cluster, siccome la metrica perde l'appellativo di distanza dato che in questo nuovo caso si considera la distanza quadratica.

Di seguito è possibile trovare il codice in R per tale metodo:

```
legameCentroide <- function() {
  df <- getDataFrame()
  distanze <- dist(df, method="euclidean", diag=TRUE, upper="TRUE")
  cluster <- hclust(distanze^2, method="centroid")
  plot(cluster, hang=-1, xlab="Metodo del centroide")
}
legameCentroide()
```

Output:



A parte le distanze quadratiche, anche questo dendrogramma risulta essere molto simile a quello del metodo del legame completo, siccome i cluster sono molto simili.

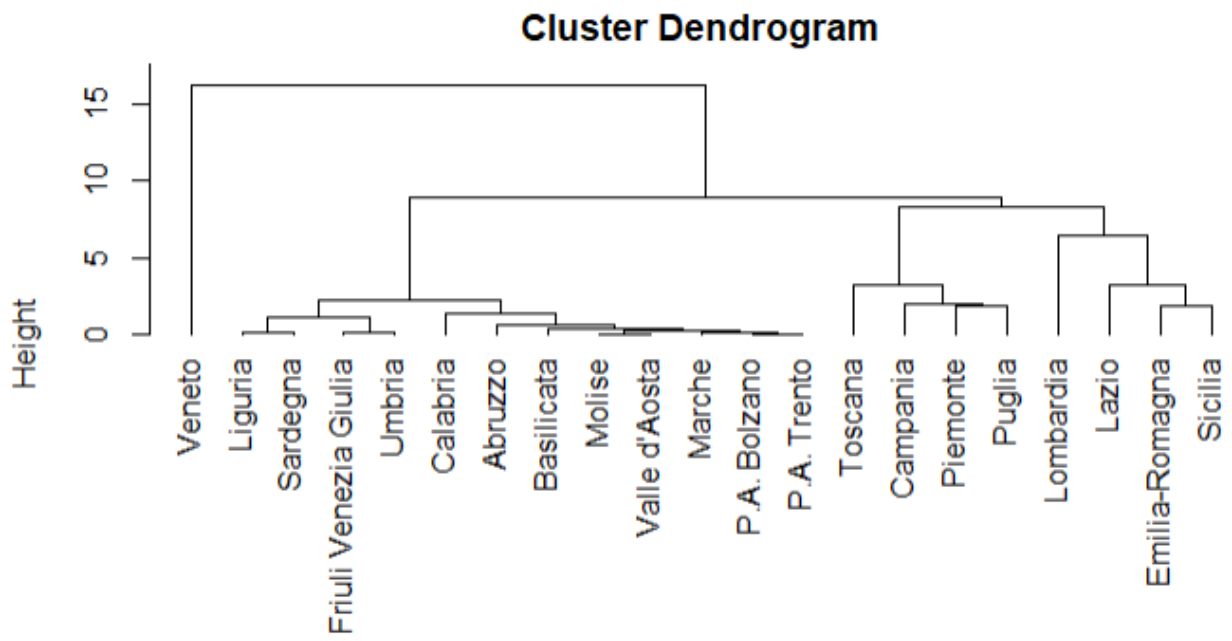
5. Metodo della mediana

Oltre al metodo del centroide, anche il metodo della mediana si basa sulle distanze elevate al quadrato. Tale metodo è molto simile a quello del centroide, con la differenza che non si basa sulla numerosità dei cluster.

Di seguito è possibile trovare il codice in R per tale metodo:

```
legameMediana <- function() {
  df <- getDataFrame()
  distanze <- dist(df, method="euclidean", diag=TRUE, upper="TRUE")
  cluster <- hclust(distanze^2, method="median")
  plot(cluster, hang=-1, xlab="Metodo della mediana")
}
legameMediana()
```

Output:



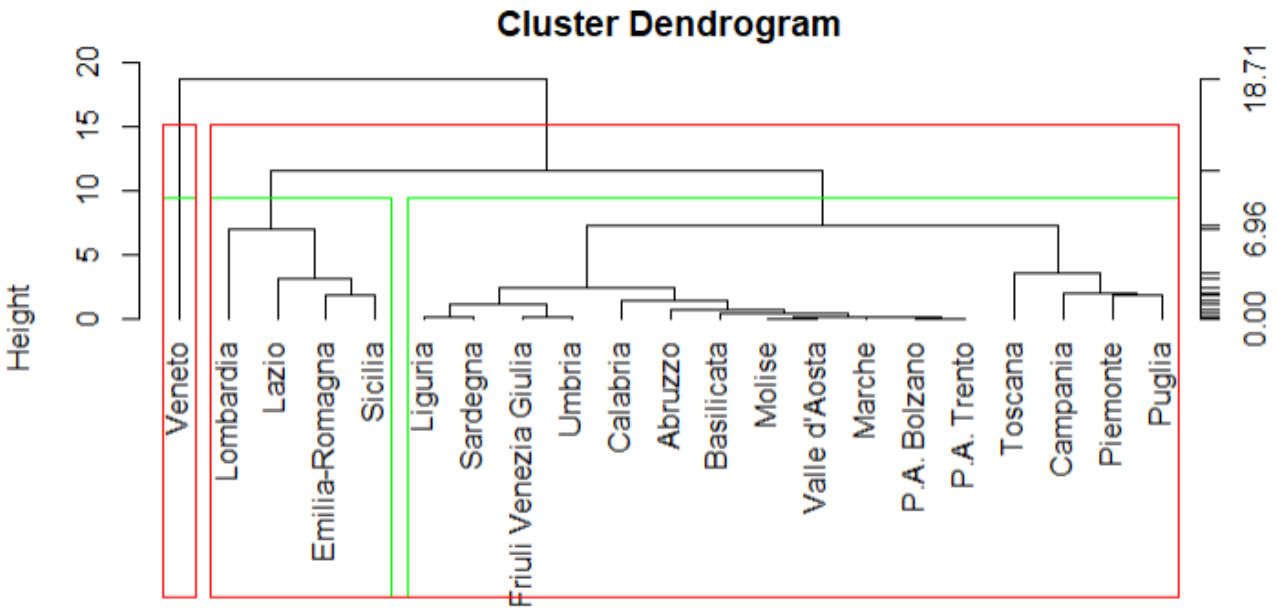
Anche il metodo della mediana sembra riportare gli stessi cluster.

5.4 Analisi del Dendogramma

Procediamo adesso ad analizzare il dendrogramma ottenuto tramite il metodo del centroide, andando a calcolare le misure di non omogeneità della partizione trovata. Utilizziamo la funzione R ***rect.hclust()*** ed in particolare gli passiamo il parametro *k* che ci permette di dividere in base al numero di cluster che si vogliono ottenere (nel nostro caso con il verde partizioniamo in 3 e in rosso partizioniamo in 2):

```
analisiDendrogramma <- function() {
  df <- getDataFrame()
  distanze <- dist(df, method="euclidean", diag=TRUE, upper="TRUE")
  cluster <- hclust(distanze^2, method="centroid")
  plot(cluster, hang=-1, xlab="Metodo del centroide")
  axis ( side =4 , at = round (c (0 , cluster$height ) ,2) )
  rect.hclust( cluster , k = 3, border = " green ")
  rect.hclust( cluster , k = 2, border = " red ")
}
analisiDendrogramma()
```

Output:



Il dendrogramma risulta diviso in 3 cluster (colore verde) ed in 2 cluster (colore rosso).

5.5 Inserire Individui nel Cluster

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():

```
df <- getDataFrame()
distanze <- dist(df, method="euclidean", diag=TRUE, upper="TRUE")
cluster <- hclust(distanze^2, method="centroid")
cutree(cluster, k = 1:20)
```

Output:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Abruzzo	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Basilicata	1	1	1	1	1	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2
Calabria	1	1	1	1	1	1	1	1	1	1	1	2	2	3	3	3	3	3	3	3
Campania	1	1	1	2	2	2	2	2	2	2	2	3	3	4	4	4	4	4	4	4
Emilia-Romagna	1	1	2	3	3	3	3	3	3	3	3	4	4	5	5	5	5	5	5	5
Friuli Venezia Giulia	1	1	1	1	1	1	1	4	4	4	4	5	5	6	6	6	6	6	6	6
Lazio	1	1	2	3	3	3	4	5	5	5	5	6	6	7	7	7	7	7	7	7
Liguria	1	1	1	1	1	1	1	4	4	4	4	5	7	8	8	8	8	8	8	8
Lombardia	1	1	2	3	4	4	5	6	6	6	6	7	8	9	9	9	9	9	9	9
Marche	1	1	1	1	1	1	1	1	1	1	1	1	1	2	10	10	10	10	10	10
Molise	1	1	1	1	1	1	1	1	1	1	1	1	1	2	10	11	11	11	11	11
P.A. Bolzano	1	1	1	1	1	1	1	1	1	1	1	1	1	2	10	10	10	10	12	12
P.A. Trento	1	1	1	1	1	1	1	1	1	1	1	1	1	2	10	10	10	10	12	13
Piemonte	1	1	1	2	2	2	2	2	7	7	7	8	9	10	11	12	12	12	13	14
Puglia	1	1	1	2	2	2	2	2	7	7	8	9	10	11	12	13	13	13	14	15
Sardegna	1	1	1	1	1	1	1	4	4	4	4	5	7	8	8	8	8	14	15	16
Sicilia	1	1	2	3	3	3	3	3	3	8	9	10	11	12	13	14	14	15	16	17
Toscana	1	1	1	2	2	5	6	7	8	9	10	11	12	13	14	15	15	16	17	18
Umbria	1	1	1	1	1	1	1	4	4	4	4	5	5	6	6	6	16	17	18	19
Valle d'Aosta	1	1	1	1	1	1	1	1	1	1	1	1	1	2	10	11	11	11	11	11
Veneto	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

L'output della funzione cutree() è un vettore contenente numeri interi positivi associati ai cluster in cui sono stati inseriti i vari individui.

5.6 Misure di sintesi associate ai cluster

In R è inoltre possibile ricavare misure di sintesi come la media campionaria, la varianza campionaria, la deviazione standard, etc, sui singoli cluster, ottenuti tagliando il dendrogramma tramite la funzione **cutree()** e successivamente utilizzando la funzione **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.

Segue il codice in R usato:

```
sintesiCluster <- function(type) {
    df <- getDataFrame()
    distanze <- dist(df, method="euclidean", diag=TRUE, upper="TRUE")
    cluster <- hclust(distanze^2, method="centroid")
    taglio <- cutree(cluster, k = 3)
    tagliolist <- list(taglio)
    aggregate(df, tagliolist, type)
}
sintesiCluster(mean)
Group.1 dataset.terapia_intensiva dataset.totale_ospedalizzati dataset.totale_positivi dataset.tamponi
      1                -0.4786353                   -0.4379096              -0.0935710      -0.4131046
      2                 1.7111698                    1.5844816               0.2663131       1.1537208
      3                 0.8134857                    0.6686273               0.4318838       1.9947908
dataset.ingressi_terapia_intensiva
                       -0.09819669
                       -0.46210208
                        3.41955537
.
> sintesiCluster <- function(type) {
+     df <- getDataFrame()
+     distanze <- dist(df, method="euclidean", diag=TRUE, upper="TRUE")
+     cluster <- hclust(distanze^2, method="centroid")
+     taglio <- cutree(cluster, k = 3)
+     tagliolist <- list(taglio)
+     aggregate(df, tagliolist, type)
+ }
> sintesiCluster(var)
  Group.1 dataset.terapia_intensiva dataset.totale_ospedalizzati dataset.totale_positivi dataset.tamponi
1       1                 0.1892372                   0.2452571              1.2251312       0.3049731
2       2                 0.3739358                   0.9211720              0.3375762       1.1304799
3       3                        NA                          NA                     NA              NA
  dataset.ingressi_terapia_intensiva
1                          0.4865471
2                          0.0000000
3                                 NA
> sintesiCluster <- function(type) {
+     df <- getDataFrame()
+     distanze <- dist(df, method="euclidean", diag=TRUE, upper="TRUE")
+     cluster <- hclust(distanze^2, method="centroid")
+     taglio <- cutree(cluster, k = 3)
+     tagliolist <- list(taglio)
+     aggregate(df, tagliolist, type)
+ }
> sintesiCluster(sd)
  Group.1 dataset.terapia_intensiva dataset.totale_ospedalizzati dataset.totale_positivi dataset.tamponi
1       1                 0.4350140                   0.4952344              1.1068565       0.5522437
2       2                 0.6115029                   0.9597770              0.5810131       1.0632403
3       3                        NA                          NA                     NA              NA
  dataset.ingressi_terapia_intensiva
1                          0.6975293
2                          0.0000000
3                                 NA
```

5.7 Misure di Non Omogeneità

A questo punto è possibile calcolare le misure di non omogeneità totale, uguale per tutti i metodi, ed interna ad ognuno dei tre cluster per ciascun metodo in modo da verificare quale metodo dia una suddivisione migliore.

Segue il codice usato in R per il calcolo della misura di non omogeneità totale:

```
getOmogeneitaTot <- function() {
  covid <- data.frame(dataset$terapia_intensiva,dataset$totale_ospedalizzati,dataset$totale_positivi,dataset$tamponi,dataset$in
gressi_terapia_intensiva)
  covid = scale(covid)
  n<-nrow(covid)
  trHI <- (n-1) * sum(apply(covid,2,var))
  trHI
}
getOmogeneitaTot()
```

Output:

```
[1] 100
```

Successivamente, tramite la funzione cutree(), andiamo ad effettuare una suddivisione degli individui in cluster. Il risultato della funzione cutree() verrà passato come parametro a table() per calcolare il numero di individui in ogni cluster. Questa operazione viene ripetuta per ogni metodo gerarchico, tramite la seguente funzione implementata in R:

```
getNumId <- function(methodval) {
  df <- getDataFrame()
  distanze <- dist(df, method = "euclidean", diag = TRUE, upper="TRUE")
  if (methodval == "centroid" || methodval == "median") {
    cluster <- hclust(distanze^2, method=methodval)
  } else {
    cluster <- hclust(distanze, method=methodval)
  }
  taglio <- cutree(cluster, k=3, h=NULL)
  num <- table(taglio)
  num
}
getNumId("single")
```

```
 taglio
  1  2  3
 19  1  1
```

```
getNumId <- function(methodVal) {
  df <- getDataFrame()
  distanze <- dist(df, method = "euclidean", diag = TRUE, upper="TRUE")
  if (methodVal == "centroid" || methodVal == "median") {
    cluster <- hclust(distanze^2, method=methodVal)
  } else {
    cluster <- hclust(distanze, method=methodVal)
  }
  taglio <- cutree(cluster, k=3, h=NULL)
  num <- table(taglio)
  num
}
getNumId("complete")
```

```
 taglio
  1  2  3
 16  4  1
```

```
getNumId <- function(methodVal) {
  df <- getDataFrame()
  distanze <- dist(df, method = "euclidean", diag = TRUE, upper="TRUE")
  if (methodVal == "centroid" || methodVal == "median") {
    cluster <- hclust(distanze^2, method=methodVal)
  } else {
    cluster <- hclust(distanze, method=methodVal)
  }
  taglio <- cutree(cluster, k=3, h=NULL)
  num <- table(taglio)
  num
}
getNumId("average")
```

```
 taglio
  1  2  3
 16  4  1
```

```r
getNumId <- function(methodVal) {
  df <- getDataFrame()
  distanze <- dist(df, method = "euclidean", diag = TRUE, upper="TRUE")
  if (methodVal == "centroid" || methodVal == "median") {
    cluster <- hclust(distanze^2, method=methodVal)
  } else {
    cluster <- hclust(distanze, method=methodVal)
  }
  taglio <- cutree(cluster, k=3, h=NULL)
  num <- table(taglio)
  num
}
getNumId("centroid")
```

```
 taglio
  1  2  3
 16  4  1
```

```r
getNumId <- function(methodVal) {
  df <- getDataFrame()
  distanze <- dist(df, method = "euclidean", diag = TRUE, upper="TRUE")
  if (methodVal == "centroid" || methodVal == "median") {
    cluster <- hclust(distanze^2, method=methodVal)
  } else {
    cluster <- hclust(distanze, method=methodVal)
  }
  taglio <- cutree(cluster, k=3, h=NULL)
  num <- table(taglio)
  num
}
getNumId("median")
```

```
 taglio
  1  2  3
 12  8  1
```

I risultati ottenuti ci confermano in parte quello che avevamo accennato nelle sezioni precedenti, ovvero che il metodo singolo si discosta da tutti gli altri; infatti, possiamo notare che il numero degli individui nei cluster è uguale per tutti gli altri metodi. In più notiamo anche una leggera differenza anche rispetto al metodo della mediana, che riporta un numero di elementi diverso.

Con il prossimo codice andremo a calcolare le misure di non omogeneità dei gruppi:

```
getOmogenCluster <- function (methodVal) {
  df <- getDataFrame()
  distanze <- dist(df, method = "euclidean", diag = TRUE, upper="TRUE")
  if (methodVal == "centroid" || methodVal == "median") {
    cluster <- hclust(distanze^2, method=methodVal)
  } else {
    cluster <- hclust(distanze, method=methodVal)
  }
  taglio <- cutree(cluster, k=3, h=NULL)
  num <- table(taglio)
  tagliolist <- list(taglio)
  agvar <- aggregate( df, tagliolist, var)
  agvar[is.na(agvar)] <- 0
  trH1 <- (num[[1]]-1)*sum(agvar[1, ])
  trH2 <- (num[[2]]-1)*sum(agvar[2, ])
  trH3 <- (num[[3]]-1)*sum(agvar[3, ])
  c(trH1, trH2, trH3)
}
getOmogenCluster("single")
```

```
 [1] 89.70556  0.00000  0.00000
```

```
getOmogenCluster <- function (methodVal) {
  df <- getDataFrame()
  distanze <- dist(df, method = "euclidean", diag = TRUE, upper="TRUE")
  if (methodVal == "centroid" || methodVal == "median") {
    cluster <- hclust(distanze^2, method=methodVal)
  } else {
    cluster <- hclust(distanze, method=methodVal)
  }
  taglio <- cutree(cluster, k=3, h=NULL)
  num <- table(taglio)
  tagliolist <- list(taglio)
  agvar <- aggregate( df, tagliolist, var)
  agvar[is.na(agvar)] <- 0
  trH1 <- (num[[1]]-1)*sum(agvar[1, ])
  trH2 <- (num[[2]]-1)*sum(agvar[2, ])
  trH3 <- (num[[3]]-1)*sum(agvar[3, ])
  c(trH1, trH2, trH3)
}
getOmogenCluster("complete")
```

```
[1] 51.76718 14.28949  0.00000
```

```r
getOmogenCluster <- function (methodVal) {
  df <- getDataFrame()
  distanze <- dist(df, method = "euclidean", diag = TRUE, upper="TRUE")
  if (methodVal == "centroid" || methodVal == "median") {
    cluster <- hclust(distanze^2, method=methodVal)
  } else {
    cluster <- hclust(distanze, method=methodVal)
  }
  taglio <- cutree(cluster, k=3, h=NULL)
  num <- table(taglio)
  tagliolist <- list(taglio)
  agvar <- aggregate( df, tagliolist, var)
  agvar[is.na(agvar)] <- 0
  trH1 <- (num[[1]]-1)*sum(agvar[1, ])
  trH2 <- (num[[2]]-1)*sum(agvar[2, ])
  trH3 <- (num[[3]]-1)*sum(agvar[3, ])
  c(trH1, trH2, trH3)
}
getOmogenCluster("average")
```

```
 [1] 51.76718 14.28949  0.00000
```

```r
getOmogenCluster <- function (methodVal) {
  df <- getDataFrame()
  distanze <- dist(df, method = "euclidean", diag = TRUE, upper="TRUE")
  if (methodVal == "centroid" || methodVal == "median") {
    cluster <- hclust(distanze^2, method=methodVal)
  } else {
    cluster <- hclust(distanze, method=methodVal)
  }
  taglio <- cutree(cluster, k=3, h=NULL)
  num <- table(taglio)
  tagliolist <- list(taglio)
  agvar <- aggregate( df, tagliolist, var)
  agvar[is.na(agvar)] <- 0
  trH1 <- (num[[1]]-1)*sum(agvar[1, ])
  trH2 <- (num[[2]]-1)*sum(agvar[2, ])
  trH3 <- (num[[3]]-1)*sum(agvar[3, ])
  c(trH1, trH2, trH3)
}
getOmogenCluster("centroid")
```

```
 [1] 51.76718 14.28949  0.00000
```

```
getOmogenCluster <- function (methodVal) {
  df <- getDataFrame()
  distanze <- dist(df, method = "euclidean", diag = TRUE, upper="TRUE")
  if (methodVal == "centroid" || methodVal == "median") {
    cluster <- hclust(distanze^2, method=methodVal)
  } else {
    cluster <- hclust(distanze, method=methodVal)
  }
  taglio <- cutree(cluster, k=3, h=NULL)
  num <- table(taglio)
  tagliolist <- list(taglio)
  agvar <- aggregate( df, tagliolist, var)
  agvar[is.na(agvar)] <- 0
  trH1 <- (num[[1]]-1)*sum(agvar[1, ])
  trH2 <- (num[[2]]-1)*sum(agvar[2, ])
  trH3 <- (num[[3]]-1)*sum(agvar[3, ])
  c(trH1, trH2, trH3)
}
getOmogenCluster("median")
```

```
[1] 21.14484 41.84372  0.00000
```

Nel metodo del legame completo, della media, del centroide e della mediana abbiamo ottenuto gli stessi valori per ogni gruppo proprio perché, come visto precedentemente, si avevano gli stessi numeri di regioni nei tre gruppi.

Una volta calcolate le misure di non omogeneità di ogni gruppo, possiamo finalmente passare al calcolo dell'indice di non omogeneità tra i cluster:

```
getIndexOmogen <- function(methodVal) {
  trHI <- getOmogeneitaTot()
  values <- getOmogenCluster(methodVal)
  values[1]
  trBetween <- trHI - (values[1] + values[2] + values[3])
  ClusterIdx <- trBetween/trHI
  ClusterIdx
}
getIndexOmogen("single")
```

```
[1] 0.1029444
```

```
getIndexOmogen <- function(methodVal) {
  trHI <- getOmogeneitaTot()
  values <- getOmogenCluster(methodVal)
  values[1]
  trBetween <- trHI - (values[1] + values[2] + values[3])
  ClusterIdx <- trBetween/trHI
  ClusterIdx
}
getIndexOmogen("complete")
```

```
[1] 0.3394332
```

```
getIndexOmogen <- function(methodVal) {
  trHI <- getOmogeneitaTot()
  values <- getOmogenCluster(methodVal)
  values[1]
  trBetween <- trHI - (values[1] + values[2] + values[3])
  ClusterIdx <- trBetween/trHI
  ClusterIdx
}
getIndexOmogen("average")
```

```
[1] 0.3394332
```

```
getIndexOmogen <- function(methodVal) {
  trHI <- getOmogeneitaTot()
  values <- getOmogenCluster(methodVal)
  values[1]
  trBetween <- trHI - (values[1] + values[2] + values[3])
  ClusterIdx <- trBetween/trHI
  ClusterIdx
}
getIndexOmogen("centroid")
```

```
[1] 0.3394332
```

```
getIndexOmogen <- function(methodVal) {
  trHI <- getOmogeneitaTot()
  values <- getOmogenCluster(methodVal)
  values[1]
  trBetween <- trHI - (values[1] + values[2] + values[3])
  ClusterIdx <- trBetween/trHI
  ClusterIdx
}
getIndexOmogen("median")
```

```
[1] 0.3701144
```

Come già detto precedentemente, anche in questo caso, l'unico valore che si discosta molto dagli altri è quello del metodo singolo. Inoltre, si può notare che il metodo di classificazione migliore è quello della mediana, con indice 0.3701144.

5.8 Metodi non gerarchici

Fino ad ora abbiamo visto i cluster ottenuti da metodi gerarchici, quindi adesso ci rigeneriamo i cluster ma questa volta tramite i 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 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.

Il metodo più utilizzato prende il nome di k-means. Tale metodo richiede che il numero di cluster sia specificato a priori e fornisce in output un'unica partizione. Esso è composto dai seguenti step:

1. Fissare a priori il numero k di cluster specificando m punti di riferimento iniziali;

2. Considerare tutti gli individui e attribuire ciascuno di essi al cluster individuato dal punto di riferimento da cui ha distanza minore;

3. Calcolare il baricentro (il centroide) di ognuno dei k gruppi così ottenuti. Tali centroidi costituiscono i punti di riferimento per i nuovi cluster;

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;

5. Si ricalcolano i centroidi dei k gruppi così ottenuti;

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.

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.

L'analisi con il metodo k-means si effettua in R mediante la funzione:

```
kmeans(X, centers, iter.max = N, nstart = M)
```

A differenza dei gerarchici, nei non gerarchici non è necessario calcolare prima le distanze.

Snippet di codice usato in R per il calcolo del kmeans:

```r
getKmeans <- function() {
  df <- getDataFrame()
  km <- kmeans(df, 3, iter.max=10, nstart=1)
  km
}
```

Output:

```
K-means clustering with 3 clusters of sizes 8, 10, 3

Cluster means:
  dataset.terapia_intensiva dataset.totale_ospedalizzati dataset.totale_positivi dataset.tamponi
1                0.81348570                    0.8831452              0.9048062       0.7957846
2               -0.63368981                   -0.6661509             -0.6170177      -0.7304876
3               -0.05699581                   -0.1345509             -0.3560911       0.3128663
  dataset.ingressi_terapia_intensiva
1                         -0.4621021
2                         -0.2680192
3                          2.1256696

Clustering vector:
              Abruzzo             Basilicata               Calabria               Campania         Emilia-Romagna
                    2                      2                      2                      1                      1
Friuli Venezia Giulia                  Lazio                Liguria              Lombardia                Marche
                    3                      1                      2                      1                      2
              Molise          P.A. Bolzano           P.A. Trento               Piemonte                 Puglia
                    2                      2                      2                      1                      1
            Sardegna               Sicilia                Toscana                 Umbria          Valle d'Aosta
                    2                      1                      1                      3                      2
             Veneto
                    3

Within cluster sum of squares by cluster:
[1] 27.843720  4.828511  9.861947
 (between_SS / total_SS =  57.5 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"    "size"
[8] "iter"         "ifault"
```

Nella funzione kmeans() di R abbiamo impostato come numero massimo di iterazioni 10 e scelta dei punti casuali 1, passando un dataframe con valori standardizzati tramite la funzione scale().

La misura di non omogeneità tra i cluster diviso la misura di non omogeneità totale è pari al 57.5% quindi maggiore del 50%.

Applicando la funzione str() possiamo ottenere maggiori info, come ad esempio:

- $cluster, un vettore di interi che indica l'allocazione di ogni individuo;
- $center, matrice contenente i centroidi dei cluster;
- $withinss, un vettore contenente le misure di non omogeneità statistica calcolate all'interno di ognuno dei cluster;
- $size, dimensione dei gruppi formati.

Segue il codice R utilizzato:

```r
getKmeans <- function(extra_info) {
  df <- getDataFrame()
  km <- kmeans(df, 3, iter.max = 10, nstart = 1)
  if (extra_info == TRUE) {
    str(km)
  } else {
    km
  }
}
```

Output:

```
List of 9
 $ cluster     : Named int [1:21] 3 3 3 2 1 3 1 3 1 3 ...
  ..- attr(*, "names")= chr [1:21] "Abruzzo" "Basilicata" "Calabria" "Campania" ...
 $ centers     : num [1:3, 1:5] 1.5316 -0.0842 -0.6101 1.4013 0.1818 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:3] "1" "2" "3"
  .. ..$ : chr [1:5] "dataset.terapia_intensiva" "dataset.totale_ospedalizzati" "dataset.totale_positivi" "dataset.tamponi"
...
 $ totss       : num 100
 $ withinss    : num [1:3] 22.25 4.89 10.14
 $ tot.withinss: num 37.3
 $ betweenss   : num 62.7
 $ size        : int [1:3] 5 4 12
 $ iter        : int 2
 $ ifault      : int 0
 - attr(*, "class")= chr "kmeans"
```

5.9 Confronto tra Metodi Gerarchici e Metodi Non Gerarchici

In conclusione, i metodi gerarchici presentano un evidente vantaggio dal punto di vista computazionale rispetto ai metodi di ottimizzazione; 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.

PARTE DUE

1. Variabile Aleatoria Binomiale

Una variabile aleatoria binomiale descrive il numero di successi di una sequenza di variabile aleatorie di Bernoulli tra loro indipendenti. Le condizioni necessarie sono l'intercambiabilità e la piena ininfluenza tra le prove di tutte le variabili aleatorie di Bernoulli.

Ecco alcuni esempi di contesti adatti all'applicazione di una variabile aleatoria binomiale. Indicare il numero di volte in cui viene lanciato un dado e il numero che esce è 3 in un numero totale di 7 lanci. Oppure quante volte si riesce a indovinare il risultato di 10 estrazioni con reinserimento di una biglia numerata da un'urna (senza il reinserimento non si verificherebbe indipendenza tra le prove, quindi non si potrebbe usare la variabile binomiale).

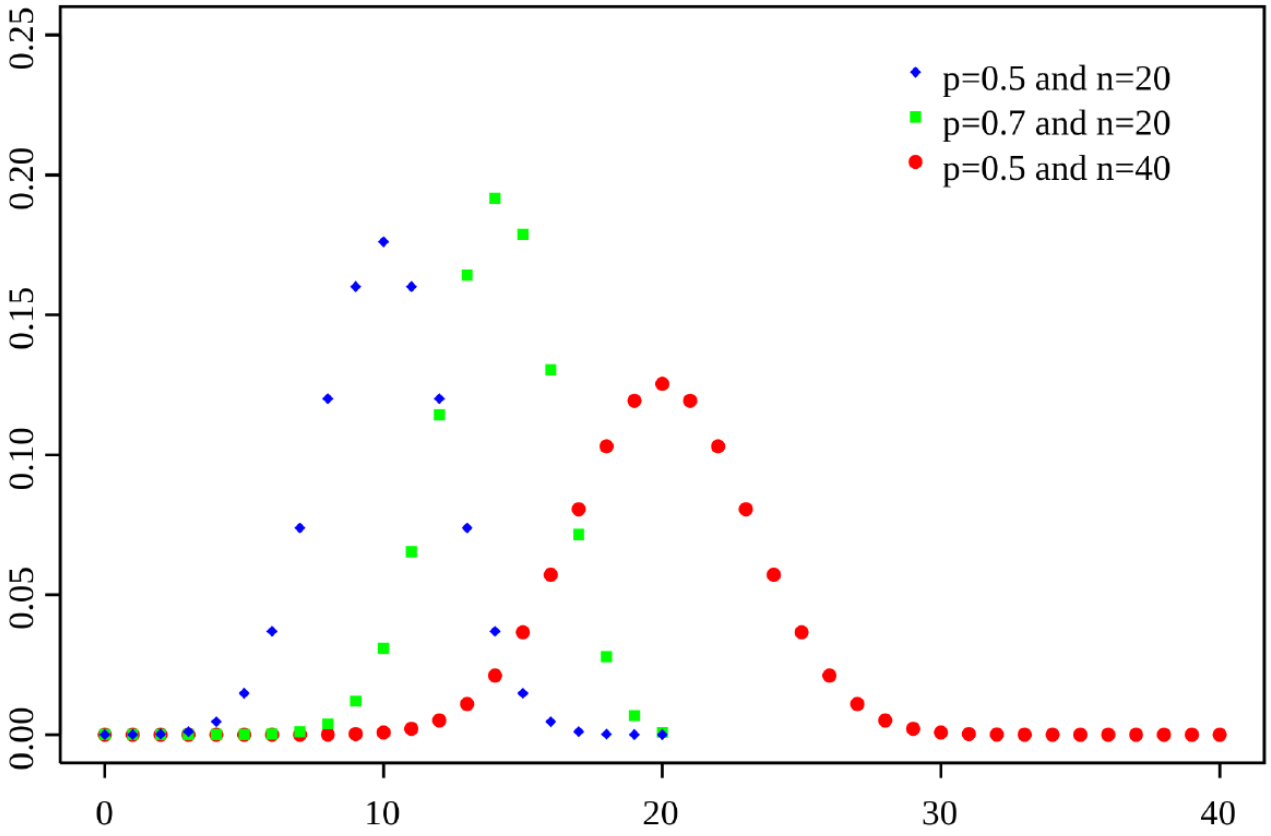
1.1 Funzione di probabilità

In maniera più formale, la variabile aleatoria binomiale avente come parametri n il numero di prove indipendenti e p la probabilità di vincita di ognuna di queste si esprime nel seguente modo.

$$P(X = x) = \frac{n!}{x!\,(n-x)!} p^x (1-p)^{n-x}$$

Se $x \in \{0,1, \dots, n\}$, altrimenti $P(X = x) = 0$.

Ecco un esempio di funzione di probabilità di tre variabili binomiali aventi diversi valori dei parametri n e p.



È facile verificare che il valore atteso $E(X) = np$.

1.2 Funzione di distribuzione

La funzione di distribuzione di una variabile aleatoria indica la somma delle probabilità che questa assuma valori minori o uguali a un valore dato. Nel caso della variabile aleatoria binomiale $F_X(x) = \sum_{i=0}^{k} \frac{n!}{i!(n-i)!} p^i (1-p)^{n-1}$ se $k \leq x < k+1$ con $k \in \{0,1,\dots,n-1\}$, oppure $F_X(x) = 0$ se $x < 0$ o $F_X(x) = 1$ se $x \geq n$.

Una variabile binomiale è la più indicata per modellare il contesto nel quale bisogna scommettere sui risultati di una serie di estrazioni di biglie numerate da 1 a 30 da un'urna con reinserimento. Diciamo che si scommette sul fatto che la biglia estratta abbia un numero non maggiore di nove, e che il numero di estrazioni è esattamente 10. Ciò significa che in 9 casi su 30 la scommessa viene vinta. Una variabile binomiale con parametri n = 10 e $p = \frac{3}{10}$ è quindi la variabile perfetta per modellare questa situazione. Ora, potendo ripetere un numero elevato di volte questo esperimento si otterrebbe una sequenza di valori riguardanti

quanti di quei dieci lanci sono stati vincenti. Nell'ambito di questo dataset, il numero di esperimenti sarà 50. Si noti che ogni esperimento è a sua volta una ripetizione di vari esperimenti (dieci in questo caso) modellabili da una variabile aleatoria di Bernoulli, per la quale in questo caso verrebbe effettuata una sola estrazione (infatti la variabile di Bernoulli è un caso particolare di variabile binomiale, il caso in cui n = 1).

Il linguaggio R permette di generare un campione di dati secondo i canoni di potenziali eventi reali modellabili da variabili binomiali con determinati parametri n e p. Il seguente codice genera un campione di dati modellabili da una variabile binomiale con n = 10 e p = 0. 3. In particolare il campione conterrà 50 simulazioni della variabile binomiale con i suddetti parametri, ovvero per ogni simulazione è come se venissero fatte 10 estrazioni con reinserimento da un'urna contenente dieci biglie numerate da 1 a 10 e si contassero solo le estrazioni di biglie con numero 1, 2 o 3.

```
n <- 10
p <- 0.3
sim <- rbinom(50, n, p)
sim
table(sim)
freq <- table(sim)/length(sim)
freq
```

Output della simulazione:

```
 [1] 2 1 4 6 3 3 2 2 1 4 3 2 3 5 1 5 2 1 4 2 3 3 4 3 1 3 4 2 4 5 2 3
[33] 2 2 4 2 4 3 5 1 4 3 1 3 2 4 4 0 2 1
sim
 0  1  2  3  4  5  6
 1  8 13 12 11  4  1
sim
   0    1    2    3    4    5    6
0.02 0.16 0.26 0.24 0.22 0.08 0.02
```

Tramite il seguente codice R è possibile disporre in maniera ordinata le frequenze relative.

```
sort(freq, decreasing=FALSE)
```

Output:

```
sim
   0    6    5    1    4    3    2
0.02 0.02 0.08 0.16 0.22 0.24 0.26
```

1.3 Stima Puntuale

Un problema centrale di inferenza statistica è quello di trovare i valori dei parametri non noti della variabile aleatoria che descrive una certa popolazione. Per fare questo è necessario introdurre la funzione di distribuzione del campione generato. Questa è la seguente.

$$F_{X_1X_2\dots X_n}(x_1x_2\dots x_n) = P(X_1 \le x_1, X_2 \le x_2, \dots, X_n \le x_n) = \dots$$

$$\dots = P(X_1 \le x_1)P(X_2 \le x_2) \dots P(X_n \le x_n) = \prod_{i=1}^{n} F_{X_i}(x_i)$$

Ogni variabile aleatoria ha uno o più parametri che permettono di determinarne la legge di probabilità. Solo quando tutti i parametri della variabile aleatoria sono già noti la legge di probabilità può dirsi completamente specificata.

1.4 Metodo dei Momenti

Il metodo dei momenti è uno dei più antichi metodi che si usa per stimare i parametri. Se esistono parametri da stimare, il metodo dei momenti consiste nell'uguagliare i primi k momenti della popolazione con in esame con i corrispondenti momenti del campione casuale. Si crea quindi il seguente sistema di k equazioni.

$$E(X^r) = M_r(x_1, x_2, \dots, x_n) = \frac{1}{n}\sum_{i=1}^{n} x_i^r \text{ per } r \in \{1, 2, \dots k)$$

Nel caso in esame, l'unico parametro da stimare è il parametro p. Nell'ambito della variabile aleatoria binomiale $X \sim B(k, p)$, è noto che $E(X) = kp$. Quindi si procede in questo modo.

$$k\hat{p} = E(X) = M(x_1, x_2, \ldots, x_n) = \frac{1}{n}\sum_{i=1}^{n} x_i = \bar{x}$$

$$\text{Quindi } \hat{p} = \frac{\bar{x}}{k}$$

Si procede a calcolare questi valori in R mediante il seguente codice.

```
p_stima <- mean(sim)/10
p_stima
```

Il valore stimato di è p = 0. 304. Si evidenzia che l'ampiezza del campione è una costante rilevante nella stima del parametro. Più è ampio il campione maggiore è la precisione di stima.

Ecco un esempio di codice R per eseguire una stima del parametro p usando solo le prime dieci misure del campione.

```
p_stima_2 <- mean(sim[0:10])/10
p_stima_2
```

Il valore risultante è 0. 32, che si discosta dal valore di p = 0. 3 in modo maggiore rispetto alla stima fatta con un numero maggiore di valori del campione.

Ovviamente una stima con soli 50 elementi non può verosimilmente essere molto accurata, però una prova della validità del metodo è il fatto che solo con questi elementi si è già riusciti ad arrivare al valore 0. 304, valore comunque abbastanza vicino a p = 0. 3.

1.5 Metodo della Massima Verosimiglianza

Il metodo della massima verosimiglianza è quello maggiormente utilizzato in statistica per stimare i parametri non noti di una variabile aleatoria che modella i dati di una popolazione. Esso è più utilizzato rispetto al metodo dei momenti descritto in precedenza. Esso consiste nel massimizzare la funzione di verosimiglianza rispetto ai parametri non noti ϑ in 1, ϑ2, ... , ϑk modo tale da ottenere una buona stima. La funzione di verosimiglianza è la seguente.

$$L(\vartheta_1, \vartheta_2, \dots, \vartheta_k) = L(\vartheta_1, \dots, \vartheta_k; x_1, \dots, x_n) = f(x_1; \vartheta_1, \vartheta_2, \dots, \vartheta_k) \dots f(x_n; \vartheta_1, \vartheta_2, \dots, \vartheta_k)$$

Per usare il metodo della massima verosimiglianza per stimare l'unico parametro non noto p si comincia con le seguenti considerazioni.

$$L(P) = \frac{k!}{x_1!(k-x_1)!} p^{x_1}(1-p)^{k-x_1} \dots \frac{k!}{x_n!(k-x_n)!} p^{x_n}(1-p)^{k-x_n}$$

Dopo varie trasformazioni matematiche si ottiene la stessa equazione risultante nella precedente sezione seguente.

$$\hat{kp} = E(X) = M(x_1, x_2, \dots, x_n) = \frac{1}{n}\sum_{i=1}^{n} x_i = \bar{x}$$

$$\text{Quindi } \hat{p} = \frac{\bar{x}}{k}$$

Ecco che questo secondo metodo porta al medesimo risultato e non è necessario simulare di nuovo la stima su R.

1.6 Stimatori e Proprietà

Per stimare parametri non noti è possibile usare tutta una serie di stimatori. Questi possono godere di tutte le seguenti proprietà:

- correttezza (oppure non distorsione);
- maggiore efficienza rispetto a un altro;
- avere varianza uniformemente minima;
- correttezza asintotica;
- consistenza.

Uno stimatore è detto corretto se il suo valore medio è pari al valore non noto calcolabile. Uno stimatore è efficiente se, tramite l'errore quadratico medio, è possibile verificare che lo scostamento quadratico dal valore

stimato è pari alla sua varianza. Tuttavia, uno stimatore non è necessariamente il migliore se ha la minore varianza, dato che deve comunque valere la seguente disuguaglianza (detta di Cramér-Rao):

$$\mathrm{Var}(\widehat{\Theta}) \geq \frac{1}{nE\left\{\left[\frac{\partial}{\partial\vartheta}\log f(X;\vartheta)\right]^2\right\}}$$

La disuguaglianza appena mostrata individua l'estremo inferiore della varianza di un dato stimatore corretto, tuttavia questo non implica che esista sempre uno stimatore avente varianza coincidente al suo estremo.

Uno stimatore è detto asintoticamente corretto se, al crescere dell'ampiezza del campione, la precisione del suo valore medio non diminuisce. Infine, uno stimatore è detto consistente se esso converge in probabilità di anche con campioni di ampiezza ϑ infinita. Ossia

$$\lim_{n \to +\infty} P(\left|\widehat{\Theta}_n - \vartheta\right| < \epsilon) = 1$$

Esistono inoltre delle condizioni sufficienti per affermare consistenza e correttezza asintotica di uno stimatore. Queste condizioni sono le seguenti.

$$\lim_{n \to +\infty} E(\widehat{\Theta}_n) = \vartheta$$

$$\lim_{n \to +\infty} var(\widehat{\Theta}_n) = \vartheta$$

1.7 Stima Intervallare

Talvolta si preferisce stimare l'appartenenza di un parametro non noto in un intervallo di valori chiamato intervallo di confidenza con un certo coefficiente detto coefficiente di confidenza o anche grado di fiducia. Esse si dicono anche statistiche, ovvero funzioni osservabili di un campione casuale, infatti l'intervallo $(C_n, \overline{C_n})$ è composto da:

$$C_n = g_1(X_1, X_2, \dots, X_n) \text{ e } \overline{C}_n = g_2(X_1, X_2, \dots, X_n).$$

Il coefficiente di confidenza è espresso come 1 − α ed è tale che $P(C_n < \vartheta < \overline{C_n}) = 1 - \alpha$.

1.8 Metodo Pivotale

Uno dei metodi usati molto spesso per effettuare una stima intervallare è il metodo pivotale. Esso consiste nell'ausilio di una variabile aleatoria di Pivot $\gamma(X_1, X_2, \dots, X_n; \vartheta)$ che:

- dipende dal campione casuale $X1$, $X2$, ..., Xn
- dipende dal parametro non noto ϑ;
- la funzione di distribuzione non contiene il parametro non noto ϑ.

Siano $\alpha_1 e\ \alpha_2$ due valori dipendenti soltanto da un fissato. Allora si avrà che

$$P(\alpha_1 < \gamma(X_1, X_2, \dots, X_n; \vartheta) < \alpha_2) = 1 - \alpha$$

Ecco che, se è possibile dimostrare che

$$\alpha_1 < \gamma(X; \vartheta) < \alpha_2 \Leftrightarrow g_1() < \vartheta < g_2() \text{ con } \forall X = (x_1, x_2, \dots, x_n),\ \vartheta \in \theta$$

allora la condizione precedente è equivalente a

$$P(g_1(X_1, X_2, \dots, X_n) < \vartheta < g_2(X_1, X_2, \dots, X_n)) = 1 - \alpha$$

Il metodo pivotale è usato su popolazioni normali. Dato un campione numeroso (almeno 30 elementi) è possibile avvalersi del teorema centrale di convergenza che afferma che la seguente variabile aleatoria converge in distribuzione ad una variabile normale standard.

$$Z_n = \frac{\overline{X}_n - \mu}{\sigma/\sqrt{n}} \xrightarrow{d} Z$$

Di fatto quindi la nostra variabile aleatoria binomiale è riconducibile a una variabile aleatoria normale su cui svolgeremo il metodo pivotale Questo procedimento è chiamato metodo pivotale approssimato. Quindi si determinerà un intervallo di confidenza dato uno specifico grado di fiducia.

Si definisce quindi una variabile aleatoria normale X tale che $E(X) = \mu$ e $var(X) = \sigma^2$. A questo punto è possibile definire una variabile aleatoria Z come una variabile aleatoria di n Pivot siccome rispetta le tre condizioni descritte all'inizio di questa sezione.

Essa può essere quindi usata per il calcolo dell'intervallo di confidenza $(C_n, \overline{C_n})$ definito anche dal grado di fiducia stabilito. Si verifica quindi la seguente relazione.

$$P\left(-\frac{z_{\alpha}}{2} < \frac{\overline{X}_n - \mu}{\sigma/\sqrt{n}} < \frac{z_{\alpha}}{2}\right) \simeq 1 - \alpha$$

Come detto, la variabile aleatoria binomiale è tale che $E(X) = kp$ e $var(X) = kp(1-p)$. Applicando il teorema centrale di convergenza risulta che

$$\frac{mean(X_n) - E(X)}{\sqrt{\frac{var(X)}{n}}} = \sqrt{n}\,\frac{mean(X_n) - E(X)}{\sqrt{var(X)}} = \sqrt{n}\,\frac{mean(X_n) - kp}{\sqrt{kp(1-p)}}$$

Assumendo campione con ampiezza minima 30 è possibile determinare l'intervallo di confidenza secondo la seguente relazione.

$$P\left(-\frac{z_{\alpha}}{2} < \sqrt{n}\,\frac{mean(X_n) - E(X)}{\sqrt{var(X)}} < \frac{z_{\alpha}}{2}\right) \approx 1 - \alpha$$

Dopo alcune trasformazioni matematiche si ottiene

$$k\left(nk + \frac{z_{\alpha}^2}{2}\right)p^2 - k\left(2n\,mean(X_n) + \frac{z_{\alpha}^2}{2}\right)p + n\,mean(X_n)^2 < 0$$

Le radici di queste disequazioni risultato infine

$$\sigma_2 = k\left(nk + \frac{z_{\frac{\alpha}{2}}^2}{2}\right)$$

$$\sigma_1 = -k\left(2n\,mean(X_n) + \frac{z_{\frac{\alpha}{2}}^2}{2}\right)$$

$$\sigma_0 = n\,mean(X_n)^2$$

Calcolando i suddetti coefficienti sarà possibile ottenere l'intervallo di confidenza.

```
n <- 10
elemCount <- 50
alpha <- 0.05
zalpha <- qnorm(1 - alpha/2, mean=0, sd=1)
medCamp <- sum(sim)/elemCount
a2 <- n * (elemCount*n + zalpha^2)
a1 <- -n * (2*elemCount*medCamp + zalpha^2)
a0 <- elemCount * medCamp^2
polyroot(c(a0, a1, a2))
```

Il suddetto codice fornisce il risultato (0. 2653021, 0. 3456867). Questo risultato è importante ed è coerente con il valore del parametro p = 0. 3.

1.9 Verifica delle Ipotesi

La verifica delle ipotesi è una importante area dell'inferenza statistica ed è molto utile, per esempio, in contesti di valutazione di prestazioni ipotizzate o di indagini sperimentali industriali.

In generale gli elementi che costituiscono il punto di partenza del procedimento di verifica delle ipotesi sono una popolazione descritta da una variabile aleatoria X caratterizzata da una funzione di probabilità o densità di probabilità $f(x; \vartheta)$, un'ipotesi su di un parametro non noto ϑ

della popolazione ed un campione casuale $X_1.X_2,\dots,X_n$ estratto dalla popolazione.

Occorre in primo luogo precisare il significato di ipotesi statistica.

Un'ipotesi statistica è una congettura su un parametro non noto ϑ. Se l'ipotesi statistica è specifica completamente $f(x;\vartheta)$ allora si dice ipotesi semplice, altrimenti ipotesi composita. Per definire una ipotesi statistica si userà il simbolo H seguito dai due punti e dall'affermazione ipotizzata.

L'ipotesi soggetta a verifica viene in genere denotata con H_0 e viene chiamata *ipotesi nulla*. Si chiama test di ipotesi il procedimento o regola con cui si decide, sulla base dei dati del campione, se accettare o rifiutare H_0. La costruzione del test richiede la formulazione, in contrapposizione all'ipotesi nulla, di una proposizione alternativa. Questa proposizione prende il nome di *ipotesi alternativa* ed è di solito indicata con H_1 .
L'ipotesi nulla (ossia quella soggetta a verifica) si ha quando $\vartheta \in \Theta_0$mentre l'ipotesi alternativa si ha quando $\vartheta \in \Theta_1$ - avendo partizionato lo spazio parametrico Θ in Θ_0 e Θ_1 – e si scrive

$$H_0: \vartheta \in \Theta_0; H_1: \vartheta \in \Theta_1$$

Il problema della verifica delle ipotesi consiste nel determinare un test ψ che permetta di suddividere, mediante opportuni criteri, l'insieme dei possibili campioni, ossia l'insieme delle n–ple $(x_1, x_2, \dots, x_n)$ assumibili dal vettore aleatorio $X_1.X_2,\dots,X_n$ in due sottoinsiemi: una regione di accettazione A dell'ipotesi nulla ed una regione di rifiuto R dell'ipotesi nulla. Il test ψ può allora essere così formulato: accettare come valida l'ipotesi nulla se il campione osservato $(x_1, x_2, \dots, x_n) \in A$, oppure rifiutare l'ipotesi nulla se invece $(x_1, x_2, \dots, x_n) \in R$.

La veridicità dell'ipotesi nulla e dell'ipotesi alternativa è mutuamente esclusiva; quindi, l'ipotesi nulla è vera se e solo se l'ipotesi alternativa è falsa e viceversa. Si dice quindi che l'ipotesi H_0 va verificata in alternativa all'ipotesi H_1.

Seguendo questo ragionamento si verificano due tipi di errore:

1. errore di tipo I: si rifiuta l'ipotesi nulla H_0 sebbene essa sia vera. Si verifica con probabilità α;

2. errore di tipo II: si accetta l'ipotesi nulla H_0 sebbene essa sia falsa. Si verifica con probabilità β.

Seguono le relazioni:

$$\alpha(\vartheta) = P(rifiutare\ H_0 \mid \vartheta), \vartheta \in \Theta_0$$

$$\beta(\vartheta) = P(accettare\ H_0 \mid \vartheta), \vartheta \in \Theta_1$$

Per campioni casuali di fissata ampiezza, la diminuzione della probabilità di commettere un errore di tipo I causa un aumento della probabilità di commettere un errore di tipo II, così come la diminuzione della probabilità di commettere un errore di tipo II causa un aumento della probabilità di commettere un errore di tipo I.

Non potendo quindi minimizzare entrambe le probabilità, si preferisce prima stabilire la probabilità di commettere un errore di tipo I (assegnando una probabilità piccola, solitamente definendo il test statisticamente significativo, definendo il 0. 05 0. 01 testo statisticamente molto significativo, oppure 0. 001 definendo il test statisticamente estremamente significativo) e poi cercare un test ψ che minimizzi la probabilità di commettere un errore di tipo II. La motivazione al fatto di stabilire prima la probabilità dell'errore di tipo I e non del tipo II è legata al fatto che nella formulazione delle ipotesi solitamente si dà priorità massima a evitare di rifiutare il vero piuttosto che l'accettare il falso. Comunque, minore è il valore di α maggiore è la credibilità di un eventuale rifiuto dell'ipotesi nulla.

I test statistici possono essere di tipo bilaterali (anche detti test bidirezionali) e si esprimono come segue.

$$H_0: \vartheta = \vartheta_0$$
$$H_1: \vartheta \neq \vartheta_0$$

Oppure possono essere test unilaterali sinistri o test unilaterali destri (test unidirezionali).

Un test unilaterale sinistro si esprime nel seguente modo.

$$H_0: \vartheta \leq \vartheta_0$$
$$H_1: \vartheta > \vartheta_0$$

Un test unilaterale destro si esprime nel seguente modo.

$$H_0: \vartheta \geq \vartheta_0$$
$$H_1: \vartheta < \vartheta_0$$

I test unilaterale sinistro e test unilaterale destro sono testati avendo fissato a priori un livello di significatività α. Passiamo alla verifica delle ipotesi sul campione da noi generato.

Avendo fissato a priori un livello di significatività α. Essendo $\mu_0 = kp_0$ e $\sigma_0^2 = kp_0(1-p_0)$, nei test unilaterali e bilaterali occorre considerare:

$$Z_{OS} = \frac{\overline{x}_n - \mu_0}{\sigma_0/\sqrt{n}} = \frac{\overline{x}_n - k\,p_0}{\sqrt{\frac{k\,p_0(1-p_0)}{n}}}$$

Considerando il nostro campione, costituito da prove e composto k = 10 da una lunghezza n = 50. Nel precedente capitolo abbiamo mostrato che una stima dell'intervallo di confidenza di grado 1 − α = 0. 95 di p è (0. 2653021, 0. 3456867).

Vogliamo verificare $H_0: kp \geq kp_0 \; con \; p_0 = 0.30$, in alternativa $H_1: kp < kp_0$ significatività α = 0. 05

Occorre considerare un test unilaterale sinistro. Utilizzando R si ha:

```
testUnilateraleSinistro <- function () {
p0 <- 0.30
alpha <- 0.05
za <- qnorm(1-alpha, mean=0, sd=1)
zos <- (medCamp - k * p0)/sqrt((k*p0*(1-p0))/n)
c(za, zos)
}
testUnilateraleSinistro()
```

Output:

```
[1] 1.644854 0.195180
```

Notiamo che z_α= 1. 644854 e z_{os} = 0.195180 e cade nella regione di rifiuto. Occorre quindi rifiutare l'ipotesi nulla con un livello di significatività del 5%.

Ora proviamo invece con $p_0 = 0.25$. Occorre considerare *test unilaterale destro*.

Con R si ha:

```
testUnilateraleDestro <- function () {
p0 <- 0.25
alpha <- 0.05
za <- qnorm(alpha, mean=0, sd=1)
zos <- (medCamp - k * p0)/sqrt((k*p0*(1-p0))/n)
c(za, zos)
}
```

Output:

```
[1] -1.644854  2.788548
```

Notiamo che -z_α= -1. 644854 e z_{os} = 2.788548 e cade nella regione di accettazione.

1.10 Criterio del chi-quadrato

Con il criterio del chi-quadrato è possibile verificare che un certo campione, descritto da una variabile aleatoria X, sia caratterizzato da una funzione di distribuzione Fx(x) con k parametri non noti da stimare. Denotiamo con H0 l'ipotesi soggetta a verifica (ipotesi nulla) e con H1 l'ipotesi alternativa. Il test chi-quadrato di misura α mira a verificare l'ipotesi nulla:

H0: X ha una funzione di distribuzione Fx(x) (avendo stimato k parametri non noti in base al campione)

In alternativa all'ipotesi H1: X non ha una funzione di distribuzione Fx(x)

Dove α è la probabilità massima di rifiutare l'ipotesi nulla quando essa è vera. Occorre determinare un test ψ di misura α che permetta di determinare una regione di accettazione e di rifiuto dell'ipotesi nulla. Il test di verifica delle ipotesi considerato è bilaterale (o a due code). Suddividiamo l'insieme dei valori che la variabile aleatoria X può assumere in r sottoinsiemi I1, I2, ..., Ir in modo che, a seconda della distribuzione ipotizzata, pi rappresenti la probabilità che la variabile aleatoria assuma un valore appartenente a Ii (i=1,2,…,r) .

Si calcola poi la quantità

$$\chi^2 = \sum_{i=1}^{r} \left(\frac{n_i - n\,p_i}{\sqrt{n\,p_i}} \right)^2 .$$

Il criterio del chi-quadrato si basa sulla seguente statistica

$$Q = \sum_{i=1}^{r} \left(\frac{N_i - n\,p_i}{\sqrt{n\,p_i}} \right)^2 ,$$

dove Ni è la variabile aleatoria che descrive il numero degli elementi del campione casuale X1,X2, . . . ,Xn (costituito da n variabili aleatorie osservabili, indipendenti e identicamente distribuite con la stessa legge di probabilità FX(x) della popolazione) che cadono nell'intervallo Ii (i = 1, 2, . . . , r).

Se la variabile aleatoria X ha una funzione di distribuzione FX(x) con k parametri non noti, si può dimostrare che per n sufficientemente grande la funzione di distribuzione della statistica Q è approssimabile con la funzione di distribuzione chi–quadrato con r−k−1 gradi di libertà.

Per garantire che ogni classe contenga in media almeno 5 elementi, si ritiene valida l'approssimazione se risulta

$$\min(np_1, np_2, \ldots, np_r) \geq 5.$$

Per un campione sufficientemente numeroso di ampiezza n, il test chi–quadrato bilaterale di misura α è il seguente:

- *si accetti l'ipotesi* H_0 *se* $\chi^2_{1-\alpha/2,r-k-1} < \chi^2 < \chi^2_{\alpha/2,r-k-1}$,
- *si rifiuti l'ipotesi* H_0 *se* $\chi^2 < \chi^2_{1-\alpha/2,r-k-1}$ *oppure* $\chi^2 > \chi^2_{\alpha/2,r-k-1}$

dove $\chi^2_{\alpha/2,r-k-1}$ *e* $\chi^2_{1-\alpha/2,r-k-1}$ *sono soluzioni delle equazioni:*

$$P(Q < \chi^2_{1-\alpha/2,r-k-1}) = \frac{\alpha}{2}, \qquad P(Q < \chi^2_{\alpha/2,r-k-1}) = 1 - \frac{\alpha}{2}.$$

Andiamo a considerare il seguente codice in R:

```
getIntervalli <- function () {
  p <- numeric(4)
  for (i in 1:4)
    p[i]<-qbinom(0.2*i, size=10, prob=0.3)
  p
}
```

Output:

```
> sim
 [1] 2 3 0 6 4 4 2 4 3 4 2 3 2 4 1 2 3 3 0 5 6 2 3 5 3 2 4 1 4 0 3 5 4 4 5 0 2 4
[39] 7 3 4 2 2 4 4 2 2 2 4 2
> n
[1] 50
> freq<-table(sim)
> freq
sim
 0  1  2  3  4  5  6  7
 4  2 14  9 14  4  2  1
```

```
> getIntervalli()
[1] 2 3 3 4
```

Otteniamo gli intervalli I1=(2) I2=(2,3) I3=(3,3) I4=(3,4) I5=(4,10).

Ora andiamo a calcolare le probabilità associate agli intervalli:

```
getProbInter <- function() {
  temp_value <- pbinom(2, size=10,prob=0.3)
  temp_value1 <- dbinom(3, size=10, prob=0.3)
  temp_value2 <- dbinom(4, size=10, prob=0.3)
  temp_value3 <- pbinom(4, size=10, prob=0.3, lower.tail = FALSE)
  c(temp_value, temp_value1, temp_value2, temp_value3)
}
```

Output:

```
> getProbInter()
[1] 0.3827828 0.2668279 0.2001209 0.1502683
> sum(getProbInter())
[1] 1
> min(getProbInter())
[1] 0.1502683
> 50*getProbInter()[4]
[1] 7.513417
```

Nello screen superiore possiamo osservare le probabilità associate agli intervalli ed inoltre possiamo anche osservare come la loro somma sia uguale ad 1 e che il valore minimo moltiplicato per 50, sia maggiore di 5.

Occorre ora determinare il numero di elementi del campione che cadono negli intervalli I1, I2, . . . , I5:

```
elemIntervalBin <- function() {
  r<-5
  a <- getIntervalliBin()
  nint<-numeric(r)
  nint[1] <- length( which ( sim < a [1]) )
  nint [2] <- length ( which (( sim >= a [1]) &( sim <a[2]) ))
  nint [3] <- length ( which (( sim >= a [2]) &( sim <a[3]) ))
  nint [4] <- length ( which (( sim >= a [3]) &( sim <a[4]) ))
  nint [5] <- length ( which ( sim >= a [4]) )
  nint
}
```

Output:

```
> elemIntervalBin()
[1]  6 14  0  9 21
> sum(elemIntervalBin())
[1] 50
```

Abbiamo così ottenuto il numero di elementi per ogni intervallo e possiamo notare come la somma di questi ultimi sia esattamente 50.

Calcoliamo ora x^2:

```
chiBin <- function () {
  nint <- elemInterval()
  chi2 <- sum ((( nint - n* val ) / sqrt ( n* val) ) ^2)
  chi2
}
```

Output:

```
> chiBin()
[1] 5.8
```

Ossia $x^2 = 5.8$ In questo caso il numero di categorie è r = 5 e occorre porre k = 1 poiché la probabilità binomiale contiene un parametro non noto, ossia p.

Pertanto, la funzione di distribuzione della statistica Q è approssimabile con la funzione di distribuzione chi–quadrato si ha r − k − 1 = 3 gradi di libertà e occorre calcolare $X^2_{\alpha 2,3}$ e $X^2_{1-\alpha/2,3}$ $con\ \alpha = 0.05$.

```r
checkchiBin <- function() {
  r <-5
  k <-1
  alpha <- 0.05
  val <- qchisq ( alpha /2, df =r -k -1)
  val2 <- qchisq (1 - alpha /2, df =r -k -1)
  c(val,val2)
}
```

Output:

```
> checkchiBin()
[1] 0.2157953 9.3484036
```

Essendo $0.21 < x^2 < 9.34$, l'ipotesi H0 di popolazione Binomiale può essere accettata.

Riproduciamo l'esperimento utilizzando una distribuzione normale.

```r
getMeanSDLength <-function() {
  n <- length(sim)
  m <- mean(sim)
  d <- sd(sim)
  c(n, m, d)
}
```

Output:

```
> getMeanSDLength <-function() {
+       n <- length(sim)
+       m <- mean(sim)
+       d <- sd(sim)
+       c(n, m, d)
+ }
> getMeanSDLength()
[1] 50.000000  3.040000  1.577391
```

Fatto ciò, possiamo adesso andare a determinare i sottoinsiemi $I_1, I_2, \dots, I_5$ tramite il seguente codice in R:

```r
getIntervalli <- function () {
  values <- getMeanSDLength()
  a <- numeric(4)
  for(i in 1:4)
    a[i] <- qnorm(0.2*i, mean=values[2], sd=values[3])
  a
}
```

Output:

```
> getIntervalli <- function () {
+     values <- getMeanSDLength()
+     a <- numeric(4)
+     for(i in 1:4)
+         a[i] <- qnorm(0.2*i, mean=values[2], sd=values[3])
+     a
+ }
> getIntervalli()
[1] 1.712434 2.640372 3.439628 4.367566
```

Gli intervalli I1, I2, . . . , I5 sono:

I1 = (0, 1.712434), I2 = (1.712434, 2.640372), I3 =(2.640372, 3.439628), I4 = (3.439628, 4.367566), I5 = (4.367566, 10)

Occorre ora determinare il numero di elementi del campione che cadono negli intervalli I1, I2, . . ., I5:

```
elemInterval <- function() {
  r<-5
  a <- getIntervalli()
  nint<-numeric(r)
  nint[1] <- length( which ( sim < a [1]) )
  nint [2] <- length ( which (( sim >= a [1]) &( sim <a[2]) ))
  nint [3] <- length ( which (( sim >= a [2]) &( sim <a[3]) ))
  nint [4] <- length ( which (( sim >= a [3]) &( sim <a[4]) ))
  nint [5] <- length ( which ( sim >= a [4]) )
  nint
}
sumElemInterval <- function () {
  e <- elemInterval()
  sum(e)
}
```

Output:

```
> elemInterval <- function() {
+     r<-5
+     a <- getIntervalli()
+     nint<-numeric(r)
+     nint[1] <- length( which ( sim < a [1]) )
+     nint [2] <- length ( which (( sim >= a [1]) &( sim <a[2]) ))
+     nint [3] <- length ( which (( sim >= a [2]) &( sim <a[3]) ))
+     nint [4] <- length ( which (( sim >= a [3]) &( sim <a[4]) ))
+     nint [5] <- length ( which ( sim >= a [4]) )
+     nint
+ }
> elemInterval()
[1]  6 14  9 14  7
```

```
> sumElemInterval <- function () {
+     e <- elemInterval()
+     sum(e)
+ }
> sumElemInterval()
[1] 50
```

Calcoliamo ora X^2

```
chi <- function () {
  nint <- elemInterval()
  chi2 <- sum ((( nint - n* 0.2) / sqrt ( n* 0.2) ) ^2)
  chi2
}
```

Output:

```
> chi()
[1] 5.8
```

Ossia $X^2 = 5.8$.

La distribuzione normale ha due parametri non noti (μ, σ 2) e quindi k = 2. Pertanto, la funzione di distribuzione della statistica Q è approssimabile con la funzione di distribuzione chi–quadrato con r – k – 1 = 2 gradi di libertà. Occorre quindi calcolare $X^2_{\alpha 2,2}$ e $X^2_{1-\alpha/2,2}$ con $\alpha = 0.05$.

```
checkchi <- function() {
  r<-5
  k<-2
  alpha <- 0.05
  val <- qchisq ( alpha /2, df =r -k -1)
  val2 <- qchisq (1 - alpha /2, df =r -k -1)
  c(val,val2)
}
```

Output:

```
> checkchi()
[1] 0.05063562 7.37775891
```

Essendo $0.05063562 \leq X^2 \leq 7.37775891$, l'ipotesi H0 di popolazione normale può essere accettata.