Statistica e analisi dei dati

Progetto di corso - analisi di un dataset

Antonio Vivone

Introduzione

Mai come al giorno d'oggi il tema immigrazione risulta essere attuale, trattato spesso dai media e discusso dai politici. L'immigrazione è anche il tema alla base di questo progetto il quale scopo è quello di analizzare i dati forniti da un'indagine effettuata dall'ISTAT. Il titolo della tavola utilizzata è *Cittadini non comunitari regolarmente presenti per motivo della presenza* e, come si evince da questo, il progetto si focalizza sul motivo della presenza di questi cittadini.

Per definizione, i cittadini comunitari sono tutti coloro che risultano essere in possesso della cittadinanza di uno degli stati membri della Comunità Europea. È bene ricordare che gli stati facenti parte dell'Unione Europea ad oggi sono: Austria, Belgio, Bulgaria, Cipro, Croazia, Danimarca, Estonia, Finlandia, Francia, Germania, Grecia, Irlanda, Italia, Lettona, Lituania, Lussemburgo, Malta, Paesi Bassi, Polonia, Portogallo, Regno Unito, Repubblica Ceca, Romania, Slovacchia, Slovenia, Spagna, Svezia, Ungheria. Lo studio è stato effettuato su cittadini non comunitari, quindi aventi cittadinanza di altri paesi.

I dati risultano essere aggiornati al 1 gennaio 2018. La tabella utilizzata è composta da 20 righe rappresentanti le regioni italiane e 5 colonne rappresentanti le maggiori motivazioni, in particolare: lavoro, famiglia, studio, asilo e altro. I dati sono sottoforma di percentuale. Utilizzando metodi statistici, il progetto verterà sul raggiungere diversi obiettivi:

- Individuare i legami fra le diverse modalità assunte.
- Individuare valori *outlaw*, ovvero anomali perchè fuori dal range
- Definire cluster per raggruppare regioni ritenute simili

Il linguaggio di programmazione utilizzato all'interno del progetto per eseguire l'analisi dei dati è R.

La prima cosa da fare è costruire la tabella all'interno del workspace di R. Iniziamo quindi a definire per ogni modalità un vettore e inserire i rispettivi valori della colonna.

```
lavoro <- c(28.3, 26.1, 30.9, 38.8, 22.1, 35.2, 23.5, 31.4, 39.3, 30.5,
            31.4, 34.1, 28.0, 10.9, 41.6, 21.6, 21.6, 23.6, 28.0, 28.2)
famiglia <- c(46.6, 53.2, 44.8, 47.3, 50.9, 48.6, 44.5, 50.7, 39.8, 48.6,
              45.0, 37.1, 45.7, 18.6, 29.1, 29.3, 25.9, 29.2, 32.8, 34.4)
studio <- c(4.8, 1.9, 2.8, 3.1, 2.6, 1.5, 2.8, 3.2, 3.2, 5.1, 3.1, 4.7,
```

```
            2.3, 0.7, 1.0, 1.6, 1.1, 1.8, 0.7, 2.1)
asilo <- c(18.1, 16.6, 17.7, 9.4, 22.4, 13.2, 26.6, 12.7, 14.5, 12.5, 18.1,
           14.3, 20.3, 67.1, 23.5, 43.0, 46.0, 42.8, 32.6, 31.0)
altro <- c(2.16, 2.20, 3.86, 1.57, 1.90, 1.54, 2.58, 2.04, 3.23, 3.29, 2.46,
           9.79, 3.70, 2.75, 4.83, 4.40, 5.46, 2.55, 5.87, 4.33)
```

Utilizzando il comando *cbind* è possibile creare una matrice a partire da vettori della stessa lunghezza. Inoltre i dati verranno inseriti per colonne (per questo la c davanti bind). Per inserire i dati per riga, il comando da utilizzare è *rbind*

```
tabella <- cbind(lavoro, famiglia, studio, asilo, altro)
```

A questo punto utilizziamo la funzione *rownames* a cui diamo in input la matrice creata in precedenza. Questa funzione ci permette di assegnare ad ogni riga un nome, che in questo caso è quello delle regioni italiane. Fatto questo stampiamo la matrice appena creata semplicemente scrivendone il nome.

```
rownames(tabella) <- c("Piemonte", "Valle d'Aos", "Liguria",
                       "Lombardia", "T-A Adige","Veneto",
                       "F-V Giulia", "Em-Romag", "Toscana",
                       "Umbria", "Marche", "Lazio", "Abruzzo",
                       "Molise", "Campania", "Puglia",
                       "Basilicata", "Calabria", "Sicilia", "Sardegna")
tabella
```

```
##             lavoro famiglia studio asilo altro
## Piemonte      28.3     46.6    4.8  18.1  2.16
## Valle d'Aos   26.1     53.2    1.9  16.6  2.20
## Liguria       30.9     44.8    2.8  17.7  3.86
## Lombardia     38.8     47.3    3.1   9.4  1.57
## T-A Adige     22.1     50.9    2.6  22.4  1.90
## Veneto        35.2     48.6    1.5  13.2  1.54
## F-V Giulia    23.5     44.5    2.8  26.6  2.58
## Em-Romag      31.4     50.7    3.2  12.7  2.04
## Toscana       39.3     39.8    3.2  14.5  3.23
## Umbria        30.5     48.6    5.1  12.5  3.29
## Marche        31.4     45.0    3.1  18.1  2.46
## Lazio         34.1     37.1    4.7  14.3  9.79
## Abruzzo       28.0     45.7    2.3  20.3  3.70
## Molise        10.9     18.6    0.7  67.1  2.75
## Campania      41.6     29.1    1.0  23.5  4.83
## Puglia        21.6     29.3    1.6  43.0  4.40
## Basilicata    21.6     25.9    1.1  46.0  5.46
```

```
## Calabria      23.6     29.2    1.8  42.8  2.55
## Sicilia       28.0     32.8    0.7  32.6  5.87
## Sardegna      28.2     34.4    2.1  31.0  4.33
```

Distribuzioni di frequenza semplice

Nella statistica descrittiva è possibile avere tre tipologie di variabili:

1. **Variabili qualitative**, ovvero variabili il cui valore non è numerico. Ad esempio, il colore dei capelli
2. **Variabili quantitative**, variabile il cui valore è di tipo numerico. Ad esempio, l'altezza.
3. **Variabili ordinabili**, in grado di essere ordinate. I valori possono essere di tipo numerico o testuale.

I dati a nostra dispozione all'interno della tabella sono di tipo numerico e ricadono quindi nella seconda tipologia descritta. Utilizziamo le *distribuzioni di frequenza* per analizzare la distribuzione dei dati, andando a sintetizzare ciò che essi rappresentano.

Siccome molto spesso è preferibile raggruppare le informazioni in classi quando si hanno variabili quantitative a disposizione, sono state definite 5 classi:

```
classi <- c(0, 15, 20, 30, 50, 70)
```

Definite le classi andiamo ad utlizzare un altro comando, *cut()*, a cui diamo in input il vettore dei dati da classificare e il vettore delle classi andando così a inserire ogni valore nell'intervallo corrispondente.

Frequenza assoluta

Per costruire una distribuzione di frequenza in R utilizziamo la funzione *table()* che ci aiuta a calcola le frequenze assolute dei dati a disposizione.

```
table(cut(lavoro, classi))
```

```
##
##  (0,15] (15,20] (20,30] (30,50] (50,70]
##       1       0      10       9       0
```

Da questa suddivisione, possiamo notare due aspetti interessanti:

- In ben nove regioni almeno 1/3 dei cittadini non comunitari si trova lì per motivi di lavoro.

- Solo in una regione, per la precisione nel Molise, il tasso dei cittadini non comunitari che giustifica la propria presenza per questioni di lavoro è estremamente basso rispetto alle altre regioni ed è del 10.9%.

Applichiamo la stessa funzione anche agli altri vettori rappresentanti le restanti modalità ed effettuiamo un'analisi.

```
table(cut(famiglia, classi))
```

```
##
##  (0,15] (15,20] (20,30] (30,50] (50,70]
##       0       1       4      12       3
```

Guardando questi dati notiamo che:

- almeno il 50% dei cittadini non comunitari residenti in tre regioni giustifica la propria presenza per questioni legate alla famiglia. Queste regioni sono Valle d'Aosta, Trentino-Alto Adige e Emilia-Romagna.
- solo in una regione il tasso è inferiore del 20%. Riferendoci alla tabella, scopriamo che è sempre il Molise con un tasso del 18.6%.

```
table(cut(studio, classi))
```

```
##
##  (0,15] (15,20] (20,30] (30,50] (50,70]
##      20       0       0       0       0
```

Notiamo:

- lo studio è motivo di presenza di una percentuale molto bassa di cittadini non comunitari. Di fatti, guardando alla tabella, è possibile notare che le percentuali sono quasi tutte inferiori al 5%.

```
table(cut(asilo, classi))
```

```
##
##  (0,15] (15,20] (20,30] (30,50] (50,70]
##       6       4       4       5       1
```

Notiamo:

- una sola regione il tasso dei cittadini non comunitari che si trovano lì per asilo è superiore al 50%. Questa regione è il Molise con una percentuale del 67.1%
- in ben sei regioni, la presenza di cittadini non comunitari per asilo è inferiore al 15%, in particolare in Lombardia, Veneto, Emilia-Romagna, Toscana, Umbria e Lazio.

```
table(cut(altro, classi))
```

```
##
##  (0,15] (15,20] (20,30] (30,50] (50,70]
##      20       0       0       0       0
```

Notiamo:

- la percentuale di cittadini non comunitari la cui presenza è dovuta a motivi non ben specificati è in tutte le regioni al di sotto del 15%. In particolare, guardando la tabella i valori non superano neanche il 10%.

Frequenze relative

Per calcolare la distribuzione delle **frequenze relative** basta dividere l'output della funzione *table()* per la lunghezza del vettore considerato, ottenuta con la funzione *length()*

```
table(cut(lavoro, classi))/length(lavoro)
```

```
##
##  (0,15] (15,20] (20,30] (30,50] (50,70]
##    0.05    0.00    0.50    0.45    0.00
```

Notiamo:

- solo il 5% delle regioni italiane ospita cittadini non comunitari la cui presenza è dovuta al lavoro per un tasso inferiore al 15%
- il restante 95% delle regioni è abitato da cittadini non comunitari presenti a causa del lavoro per un tasso superiore al 20%

Com'è possibile vedere, i risultati forniti dall'analisi delle frequenze relative corrispondono a quelli delle frequenze assolute. Completiamo calcolando le frequenze relative per le altre modalità.

```
table(cut(famiglia, classi))/length(famiglia)
```

```
##
##  (0,15] (15,20] (20,30] (30,50] (50,70]
##    0.00    0.05    0.20    0.60    0.15
```

```
table(cut(studio, classi))/length(studio)
```

```
##
##  (0,15] (15,20] (20,30] (30,50] (50,70]
##       1       0       0       0       0
```

```
table(cut(asilo, classi))/length(asilo)
```

```
##
##  (0,15] (15,20] (20,30] (30,50] (50,70]
##    0.30    0.20    0.20    0.25    0.05
```

```
table(cut(altro, classi))/length(altro)
```

```
##
##  (0,15] (15,20] (20,30] (30,50] (50,70]
##       1       0       0       0       0
```

Frequenze assolute cumulate

In R è possibile cumulare le varie frequenze sfruttando la funzione *cumsum()*. Vediamo di seguito i valori calcolati

```
cumsum(table(cut(lavoro, classi)))
```

```
##  (0,15] (15,20] (20,30] (30,50] (50,70]
##       1       1      11      20      20
```

```
cumsum(table(cut(famiglia, classi)))
```

```
##  (0,15] (15,20] (20,30] (30,50] (50,70]
##       0       1       5      17      20
```

```r
cumsum(table(cut(studio, classi)))
```

```
##  (0,15] (15,20] (20,30] (30,50] (50,70]
##      20      20      20      20      20
```

```r
cumsum(table(cut(asilo, classi)))
```

```
##  (0,15] (15,20] (20,30] (30,50] (50,70]
##       6      10      14      19      20
```

```r
cumsum(table(cut(altro, classi)))
```

```
##  (0,15] (15,20] (20,30] (30,50] (50,70]
##      20      20      20      20      20
```

Frequenze relative cumulate

```r
cumsum(table(cut(lavoro, classi))/length(lavoro))
```

```
##  (0,15] (15,20] (20,30] (30,50] (50,70]
##    0.05    0.05    0.55    1.00    1.00
```

```r
cumsum(table(cut(famiglia, classi))/length(famiglia))
```

```
##  (0,15] (15,20] (20,30] (30,50] (50,70]
##    0.00    0.05    0.25    0.85    1.00
```

```r
cumsum(table(cut(studio, classi))/length(studio))
```

```
##  (0,15] (15,20] (20,30] (30,50] (50,70]
##       1       1       1       1       1
```

```r
cumsum(table(cut(asilo, classi))/length(asilo))
```

```
##  (0,15] (15,20] (20,30] (30,50] (50,70]
##    0.30    0.50    0.70    0.95    1.00
```

```r
cumsum(table(cut(altro, classi))/length(altro))
```

```
##  (0,15] (15,20] (20,30] (30,50] (50,70]
##       1       1       1       1       1
```

Le rappresentazioni grafiche

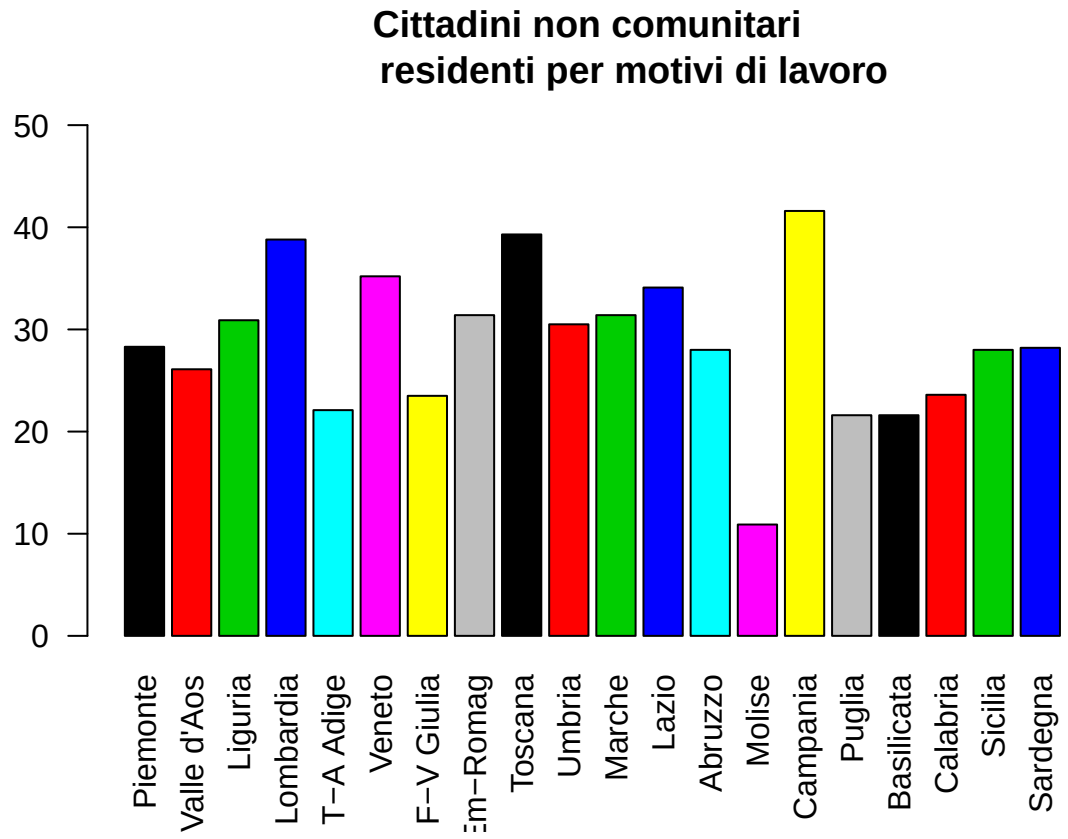
Le rappresentazioni grafiche ci permettono di confrontare e analizzare i dati contenuti nella tabella iniziale in maniera decisamente più veloce. Esistono moltissime tipologie di grafici ma non sempre sono tutti adatti a raggiungere lo scopo o a visualizzare in maniera adatta i dati che si vogliono rappresentare. La creazione di grafici e la rappresentazione di dati sono attività che ricadono nella data visualization.

Per i dati a nostra disposizione, si è scelto di utilizzare un grafico a barre in modo da rendere chiara la differenza e facilitare i confronto fra le varie regioni. In R è possibile creare un grafico a barre attraverso il comando *barplot()*. All'interno del comando *barplot* sono state specificate alcune opzioni, ovvero:

- *col* ci permette di colorare le barre del grafico
- *ylim* per indicare il punto dove far iniziare e far finire l'asse delle ordinate
- *las* per stampare le label dell'asse delle ascisse in verticale

In questo primo *barplot* sull'asse delle ascisse sono state inserite le venti regioni italiane, mentre sull'asse delle ordinate i valori della tabella corrispondenti alla colonna "lavoro".

```
barplot(tabella[,1], col=1:20, ylim = c(0, 50), las=2,
        main = "Cittadini non comunitari
        residenti per motivi di lavoro")
```

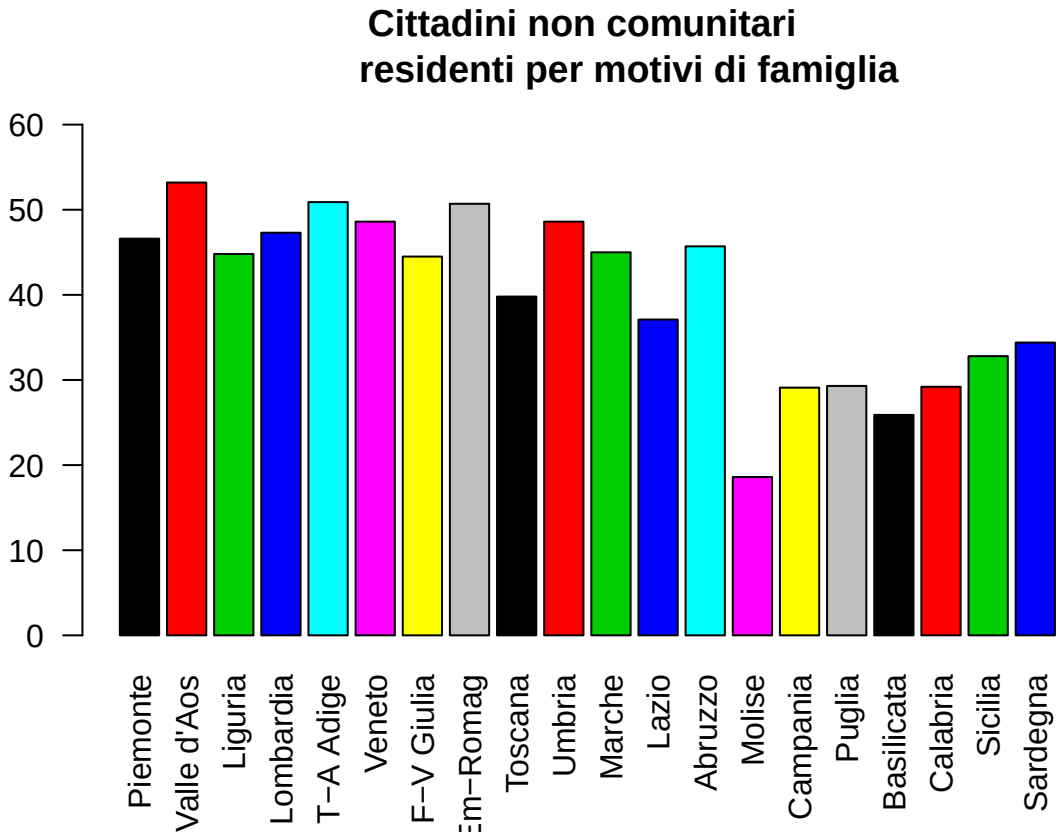


Come visto in precedenza con l'analisi delle frequenze relative, anche qui notiamo che:

- il Molise è la regione con il tasso più basso di cittadini non comunitari la cui presenza è dovuta a ragioni di lavoro.
- la Campania, la Toscana e la Lombardia sono le tre regioni con il tasso più alto.

Passiamo adesso all'analisi della modalità "famiglia"

```
barplot(tabella[,2], col=1:20, ylim = c(0,60), las=2,
        main = "Cittadini non comunitari
        residenti per motivi di famiglia")
```

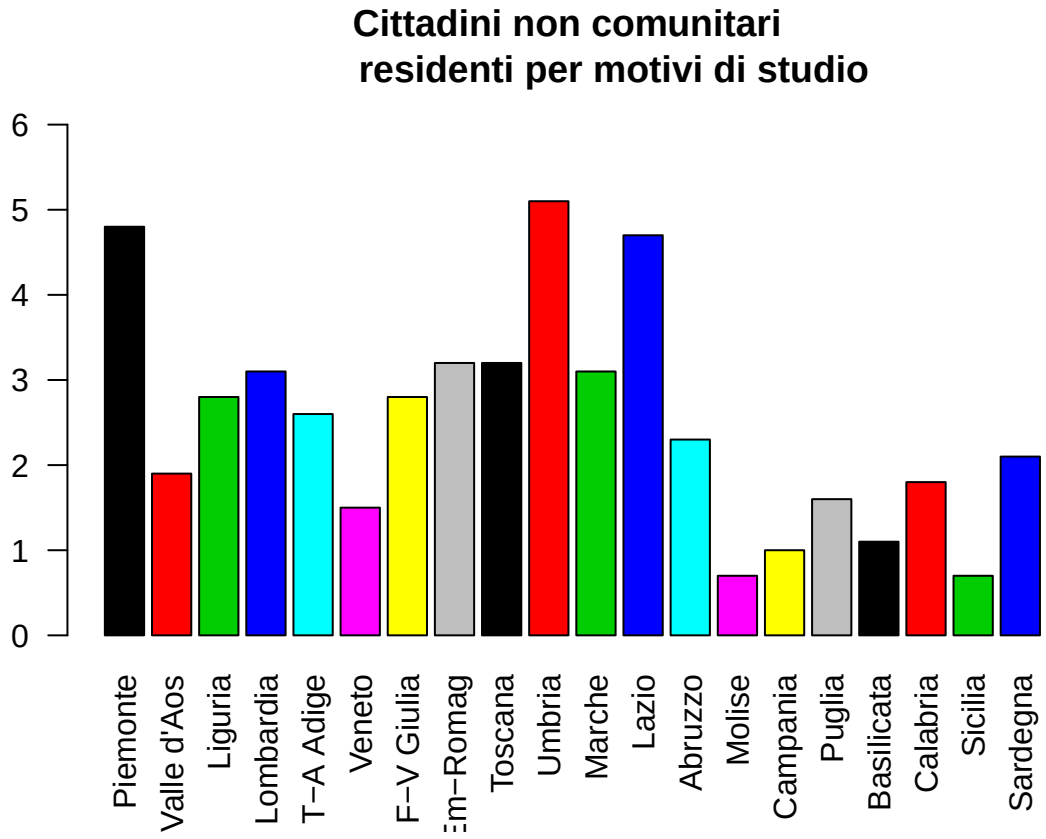


Possiamo anche qui vedere che, per quanto riguarda i motivi di famiglia:

- il Molise ha il tasso più basso fra tutte le regioni
- il tasso è più alto del 50% in Valle d'Aosta, Emilia-Romagna e Trentino-Alto Adige

Per quanto riguarda lo studio invece:

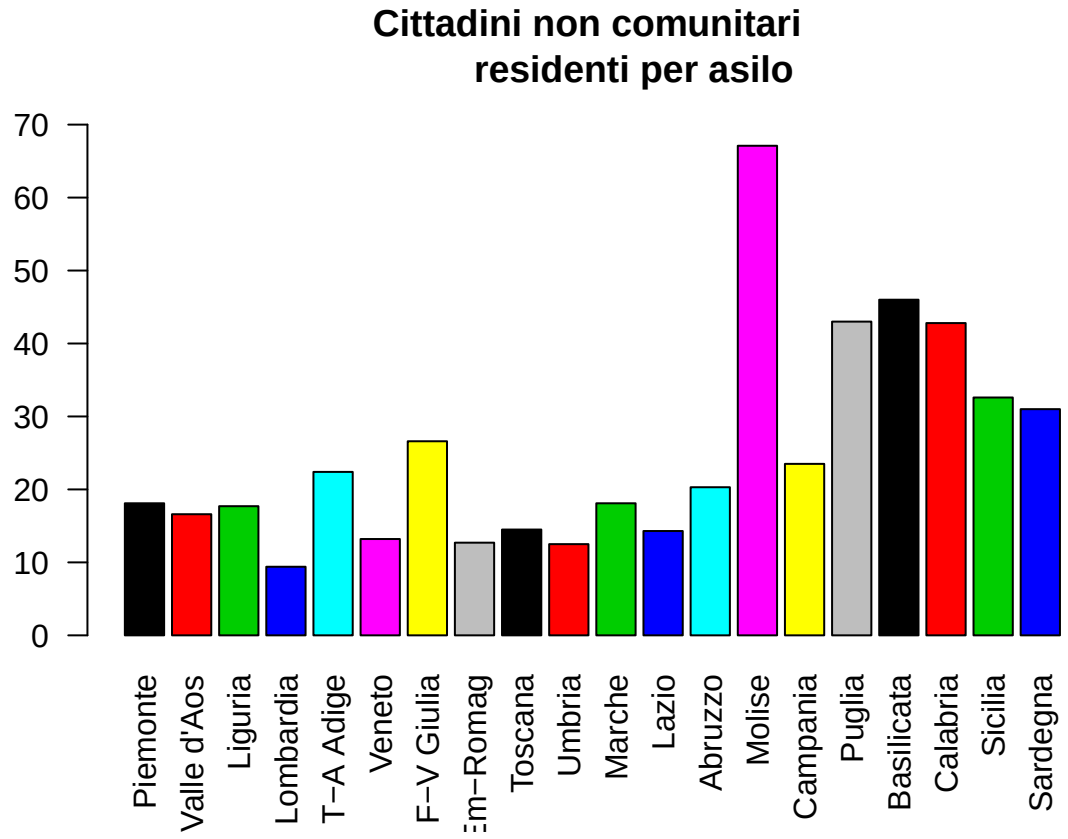
```
barplot(tabella[,3], col=1:20, ylim = c(0,6), las=2,
        main = "Cittadini non comunitari
        residenti per motivi di studio")
```



Quello che si nota è che in generale i cittadini non comunitari non vengono in Italia per studiare in quanto i tassi risultano estremamente bassi in tutte le regioni con i più bassi in Molise e Sicilia.

La situazione di asilo politico è la seguente:

```
barplot(tabella[,4], col=1:20, ylim = c(0,70), las=2,
        main = "Cittadini non comunitari
        residenti per asilo")
```

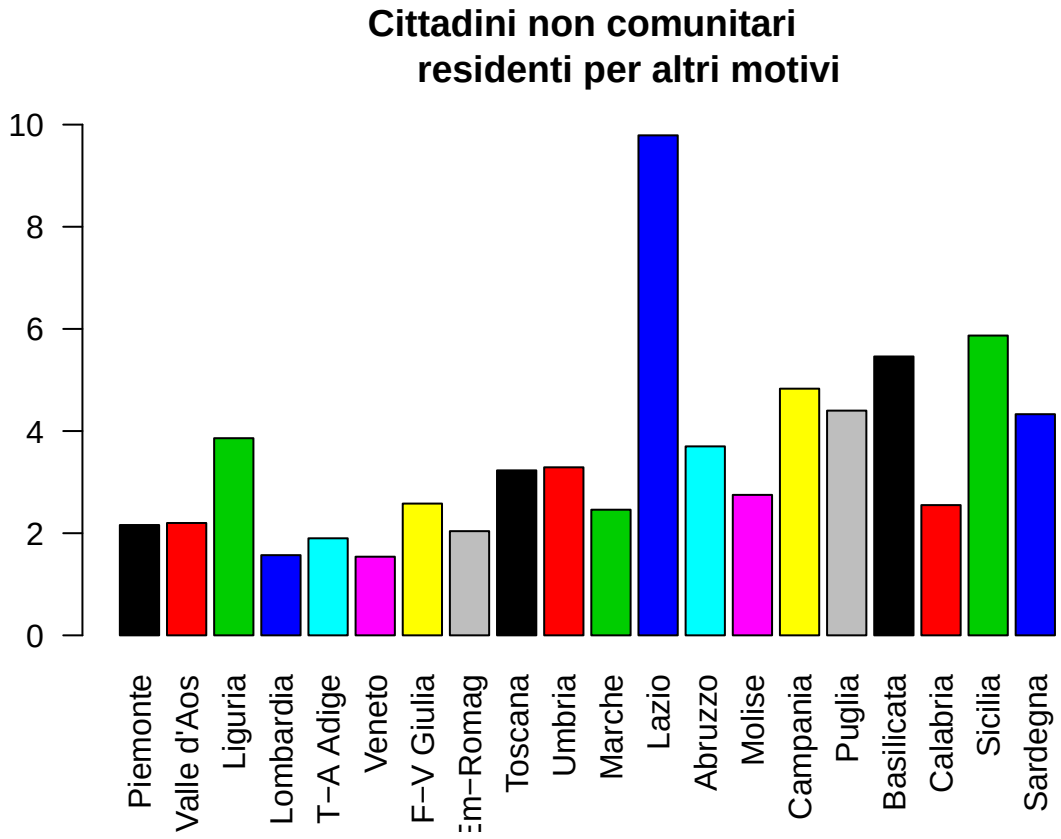


Fra le osservazioni interessanti di questa situzione abbiamo:

- il Molise risulta avere il tasso più alto di cittadini non comunitari rifugiati.
- la Lombardia, insieme a qualche altra regione del nord Italia, presenta il tasso più basso

Per quanto riguarda le motivazioni non specificate:

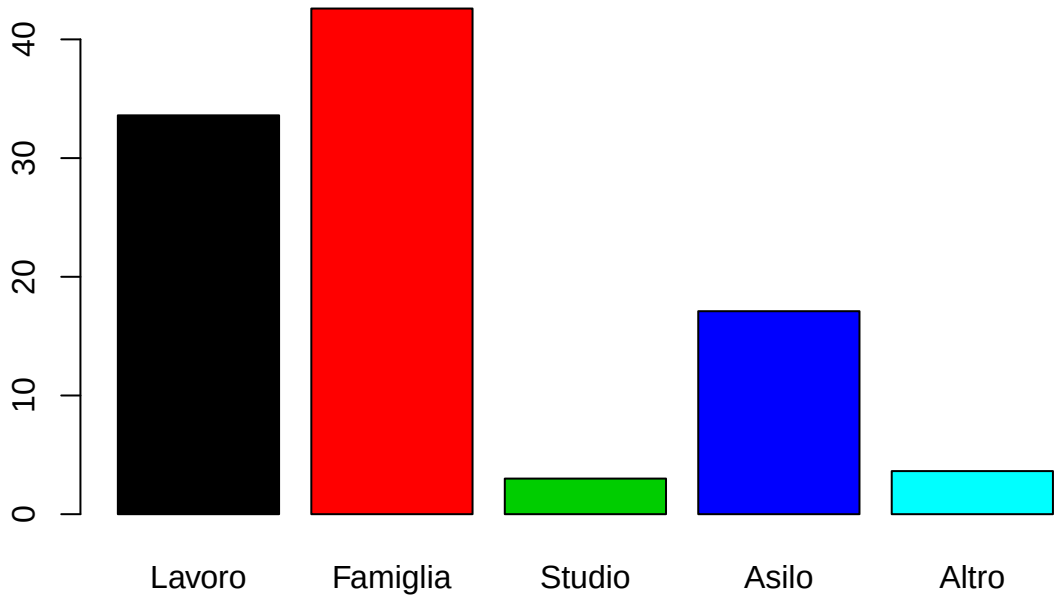
```
barplot(tabella[,5], col=1:20, ylim = c(0,10), las=2,
        main = "Cittadini non comunitari
        residenti per altri motivi")
```



È possibile notare che i dati raccolti per motivi non ben specificati sono decisamente bassi. Si differenzia solo la situazione del Lazio, che presente un tasso decisamente elevato rispetto alle altre.

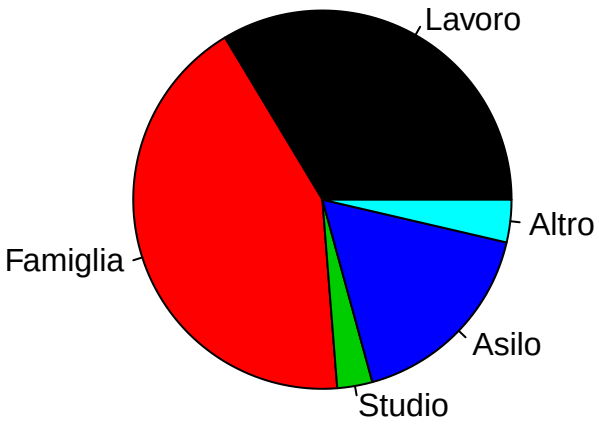
La tabella fornita dall'ISTAT comprende anche una riga riguardante tutta l'Italia. Procediamo quindi alla creazione di un grafico a barre dove sull'asse orizzontale disponiamo le modalità che possono essere assunte dal campione e sull'asse verticale il valore della corrispondente modalità Visto che utilizzeremo questi dati solo per la creazione di alcuni grafici, non aggiungiamo la riga alla matrice precedente ma ne creiamo una nuova.

```
italia <- rbind(c(33.6, 42.6, 3.0, 17.1, 3.63))
colnames(italia) <- c("Lavoro", "Famiglia", "Studio", "Asilo", "Altro")
barplot(italia[1,], col = 1:5)
```



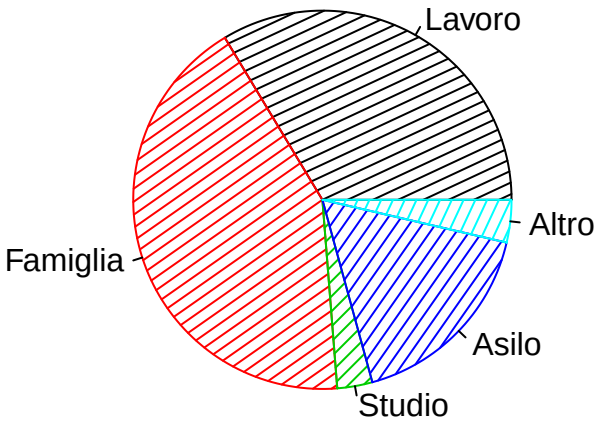
Come detto prima, R fornisce anche la possibilità di utilizzare altre tipi di diagrammi, come quelli a torta, che suddividono un cerchio in diversi settori tanti quante sono le modalità che può assumere il campione. Per creare un diagramma a torta in R è possibile utilizzare la funzione *pie()*

```
pie(italia[1,], col=1:5)
```



È possibile applicare diversi tipi di tratteggio al diagramma a torta, utilizzando linee al posto della tinta unita. Aggiungengo le opzione *density* e *angle* è possibile impostare la densità delle linee del tratteggio e l'angolo che esse devono assumere.

```
pie(italia[1,], density = 18, angle= 15+10*(1:5), col = 1:5)
```



Queste sono solo due delle diverse tipologie di grafico messe a disposizione da R. È possibile notare che la percezione della distribuzione dataci dal grafico a barre è la stessa di quella del grafico a torta, cambia solo la modalità di visualizzazione dei dati.

Diagramma di Pareto

Il diagramma di Pareto è un grafico che rappresenta l'importanza delle differenze causate da un certo fenomeno. Contiene al suo interno un diagramma a barre verticali con le modalità organizzate in maniera decrescente rispetto alla loro frequenza relativa. Le frequenze relative sono rappresentate sottoforma cumulata come una curva. Il suo scopo è quello di aiutare a capire quali sono i maggiori fattori che hanno influenza su un dato fenomeno.

Prendiamo come modalità in analisi l'asilo politico. Con l'opzione *drop=FALSE* diciamo ad R di includere anche le label della righe.

```
tabella[,4,drop=FALSE]
```

```
##             asilo
## Piemonte     18.1
## Valle d'Aos  16.6
## Liguria      17.7
## Lombardia     9.4
## T-A Adige    22.4
```

```
## Veneto       13.2
## F-V Giulia   26.6
## Em-Romag     12.7
## Toscana      14.5
## Umbria       12.5
## Marche       18.1
## Lazio        14.3
## Abruzzo      20.3
## Molise       67.1
## Campania     23.5
## Puglia       43.0
## Basilicata   46.0
## Calabria     42.8
## Sicilia      32.6
## Sardegna     31.0
```

La prima cosa che facciamo è ordinare in maniera decrescente in base alla modalità *asilo* che è quella che vogliamo analizzare.

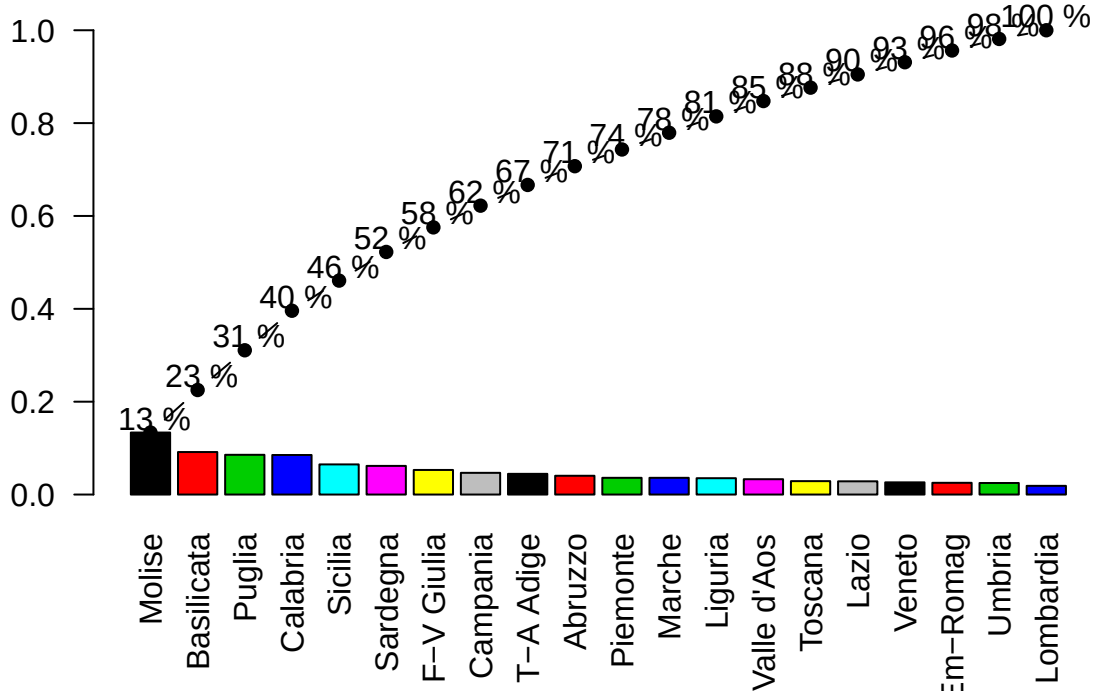
```
ordinata <- tabella[order(asilo, decreasing=TRUE),4 , drop = FALSE]
ordinata
```

```
##             asilo
## Molise       67.1
## Basilicata   46.0
## Puglia       43.0
## Calabria     42.8
## Sicilia      32.6
## Sardegna     31.0
## F-V Giulia   26.6
## Campania     23.5
## T-A Adige    22.4
## Abruzzo      20.3
## Piemonte     18.1
## Marche       18.1
## Liguria      17.7
## Valle d'Aos  16.6
## Toscana      14.5
## Lazio        14.3
## Veneto       13.2
## Em-Romag     12.7
## Umbria       12.5
## Lombardia     9.4
```

```
asilo_freqRel <- ordinata[,1]/sum(asilo)
asilo_freqRelCum <- cumsum(asilo_freqRel)
asilo_freqRelCum
```

```
##      Molise  Basilicata      Puglia    Calabria     Sicilia    Sardegna
##   0.1335589   0.2251194   0.3107086   0.3958997   0.4607882   0.5224920
##  F-V Giulia    Campania   T-A Adige     Abruzzo    Piemonte      Marche
##   0.5754379   0.6222134   0.6667994   0.7072054   0.7432325   0.7792596
##     Liguria Valle d'Aos     Toscana       Lazio      Veneto    Em-Romag
##   0.8144904   0.8475318   0.8763933   0.9048567   0.9311306   0.9564092
##      Umbria   Lombardia
##   0.9812898   1.0000000
```

```
pareto <- barplot(asilo_freqRel, ylim = c(0,1.1), col=1:20, las=2)
lines(pareto, asilo_freqRelCum, type = "b", pch=16)
text(pareto-0.01, asilo_freqRelCum+0.03,
     paste(format(asilo_freqRelCum *100, digits=2), "%"))
```

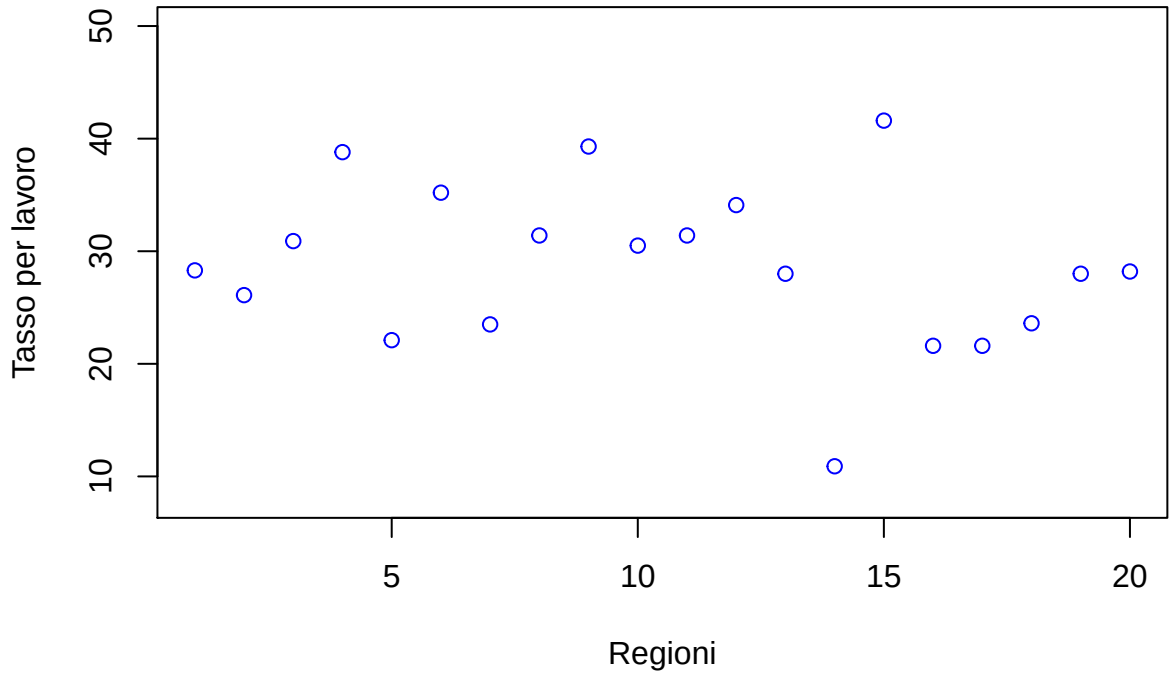


Quello che possiamo notare dal diagramma di Pareto è che circa il 70% dei cittadini non comunitari presenti in Italia per motivi di asilo politico risiede nelle prime dieci regioni sopra

elencate. Notasi anche che in queste dieci regioni figura tutto il Sud Italia, composto da ben 8 regioni.

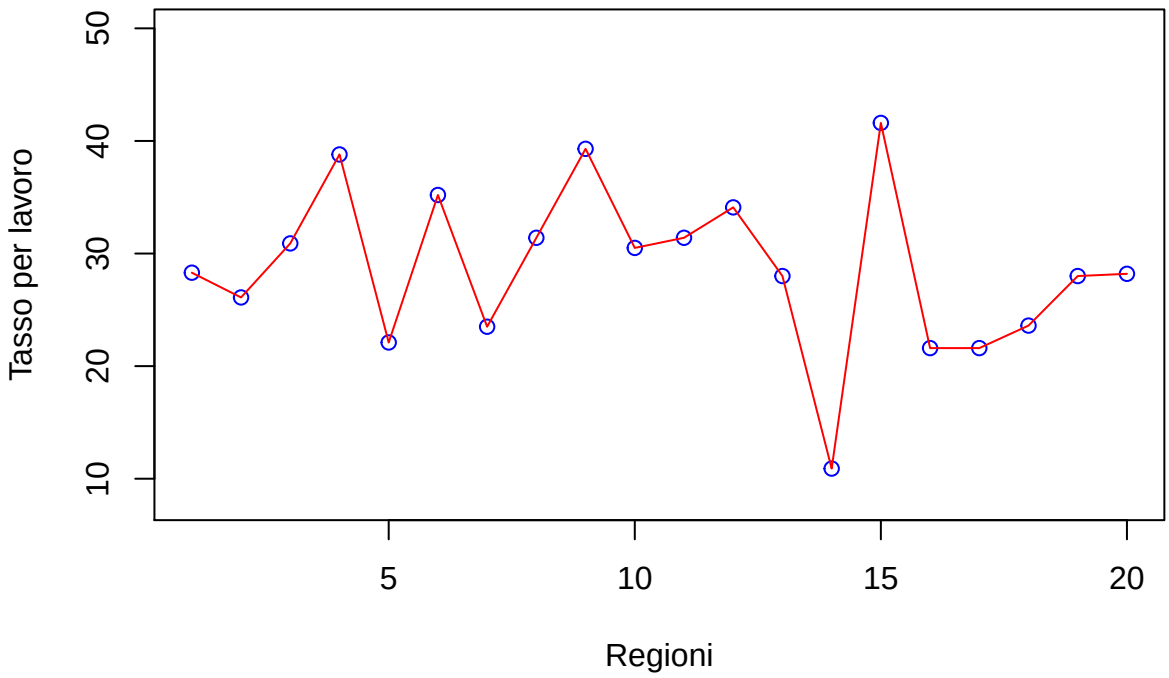
Consideriamo sempre il vettore quantitativo *lavoro*. Attraverso l'utilizzo della funzione *plot()* è possibile visualizzare su un piano cartesiano il valore assunto da ogni regione italiana. Nella seguente riga di codice sono state utilizzate due opzioni, *xlab* e *ylab*, che ci permettono di impostare manualmente il valore dell'etichetta degli assi.

```
plot(lavoro, col="blue", ylim = c(8,50), xlab = "Regioni", ylab="Tasso per lavoro")
```



È possibile connettere i punti del grafico con delle linee utilizzando l'apposito comando *lines()*

```
plot(lavoro, col="blue", ylim = c(8,50), xlab = "Regioni", ylab="Tasso per lavoro")
lines(lavoro, col="red")
```



La creazione di plot non ci permette di dedurre nuove informazioni rispetto a quello che sappiamo già.

Possiamo anche creare un unico grafico che ci permetta di confrontare i valori assunti dalle diverse modalità. Associamo ad ogni modalità un simbolo diverso:

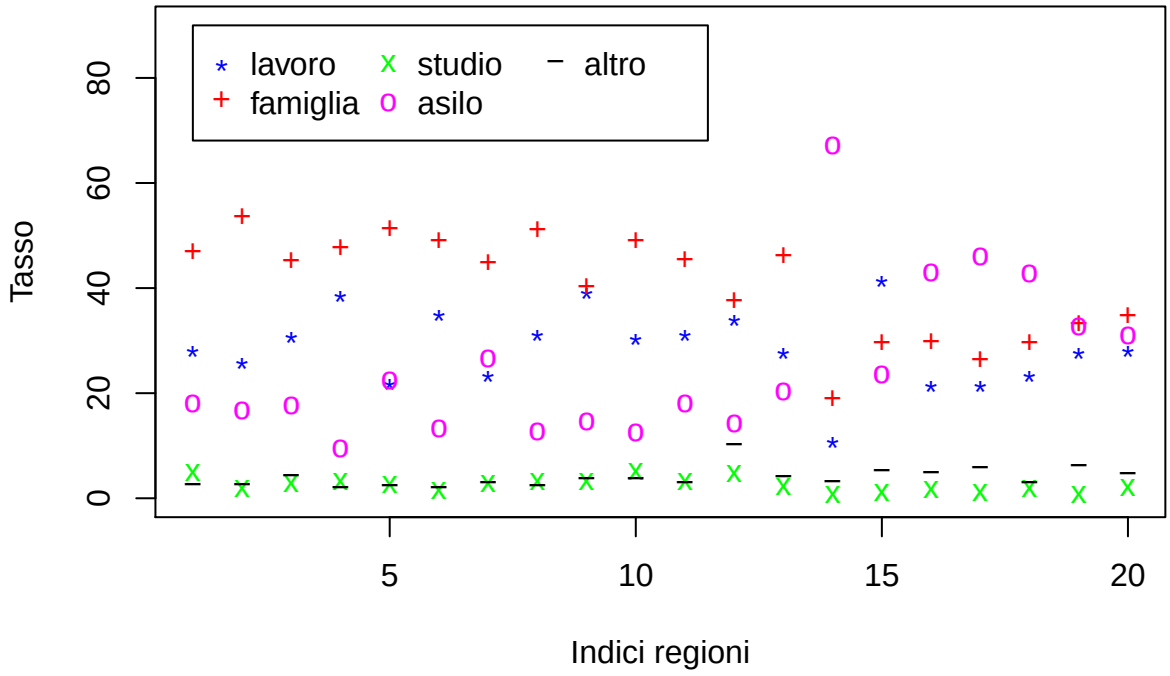
- $*$ per indicare i valori di lavoro
- $+$ per indicare i valori di famiglia
- x per indicare i valori di studio
- o per indicare i valori di asilo
- $-$ per indicare i valori di altro

Useremo la funzione *points()* per aggiungere altri punti al grafico già esistente. Fra i parametri utilizzati compare *pch* che ci permette di indicare quale simbolo utilizzare per rappresentare i valori. La funzione *legend()* serve a creare una legenda all'interno del grafico disegnato. Il parametro *ncol* utilizzato all'interno della funzione *legend()* ci permette di indicare il numero di colonne da utilizzare.

```
plot(lavoro, pch="*", ylim = c(0, 90), col="blue",
     ylab="Tasso", xlab="Indici regioni")
points(famiglia, pch="+", col="red")
points(studio, pch="x", col="green")
```

```r
points(asilo, pch="o", col="magenta")
points(altro, pch="-", col="black")

legend(1, 90, c("lavoro", "famiglia", "studio", "asilo", "altro"),
       pch=c("*","+", "x", "o", "-"),
       col = c("blue", "red", "green", "magenta", "black"),
       ncol = 3)
```

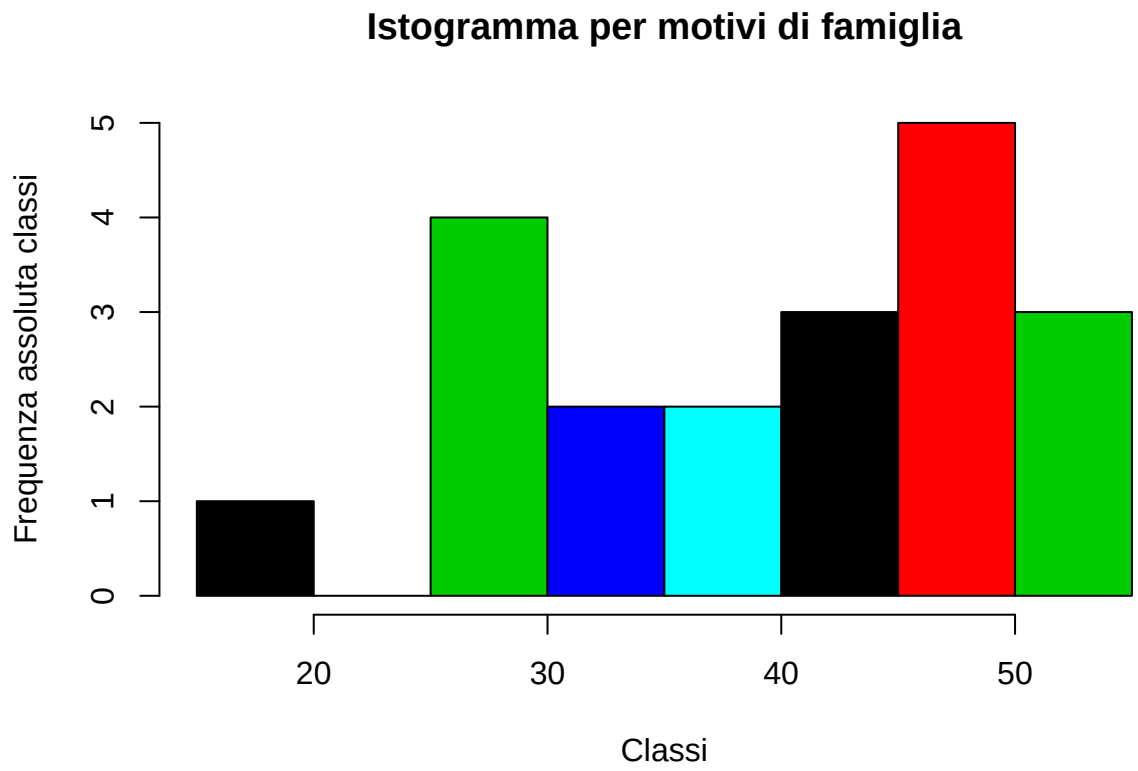


Istogrammi

Gli istogrammi sono una particolare rappresentazione grafica di una distribuzione di frequenza in classi e vengono utilizzati per le variabili *quantitative* Questa tipologia di rappresentazione grafica viene ottenuta mediante rettangoli adiacenti aventi per basi segmenti i cui estremi corrispondono agli estremi delle classi. Fissate le basi, le altezze devono essere tali che l'area di ogni rettangolo risultante sia uguale alla frequenza relativa o assoluta della classe stessa. È possibile creare un'istogramma in R utilizzando la funzione *hist()*. Questo comando prevede diverse opzioni interessanti:

- *freq*, che può valere FALSE indicando così di utilizzare la frequenza relativa, oppure può essere impostato a TRUE per utilizzare la frequenza assoluta. Di default è impostato a TRUE se e solo se le classi sono equidistanti.

```r
h <- hist(famiglia, freq = TRUE, col = 1:5,
          main="Istogramma per motivi di famiglia",
          ylab="Frequenza assoluta classi", xlab = "Classi")
```



Utilizzando il comando *str()* R è in grado di fornirci informazioni aggiuntive sulla creazione dell'istogramma, come la suddivisione in classi utilizzata.

```r
str(h)
```

```
## List of 6
##  $ breaks  : int [1:9] 15 20 25 30 35 40 45 50 55
##  $ counts  : int [1:8] 1 0 4 2 2 3 5 3
##  $ density : num [1:8] 0.01 0 0.04 0.02 0.02 0.03 0.05 0.03
##  $ mids    : num [1:8] 17.5 22.5 27.5 32.5 37.5 42.5 47.5 52.5
##  $ xname   : chr "famiglia"
##  $ equidist: logi TRUE
##  - attr(*, "class")= chr "histogram"
```

Com'è possibile vedere, le informazioni fornite da R riguardo l'istogramma sono diverse:

- *breaks* è un vettore contenente la suddivisione in classi utilizzata

- *counts* è un vettore i quali valori indicano il numero di elementi che ricade in ogni classe
- *density* è il vettore con le densità delle classi stimate
- *mids* vettore contenente i punti centrali delle classi
- *xname* è il nome del vettore di dati su cui si basa l'istogramma
- *equidist* un valore booleano per indicare se la distanza fra i valori di breaks è uguale

Da queste informazioni è possibile ricavare la frequenza relativa moltiplicando il vettore *density* per l'ampiezza delle classi. In questo caso, l'ampiezza delle classi è 5.

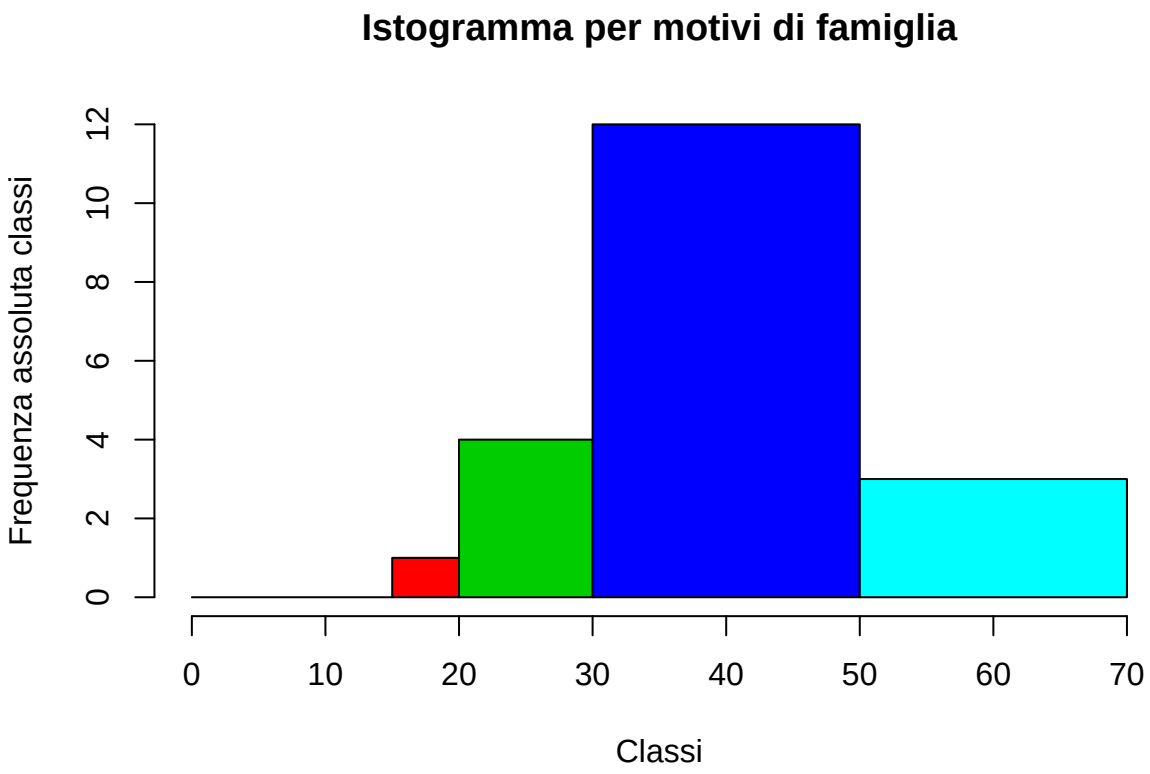
```
fr <- h$density * 5
fr
```

```
## [1] 0.05 0.00 0.20 0.10 0.10 0.15 0.25 0.15
```

Utilizziamo adesso il vettore delle classi definito precedentemente durante il calcolo delle frequenze per definire un nuovo istogramma.

```
h <- hist(famiglia, freq = TRUE, col = 1:5,
          main="Istogramma per motivi di famiglia",
          ylab="Frequenza assoluta classi", xlab = "Classi",
          breaks = classi)
```

```
## Warning in plot.histogram(r, freq = freq1, col = col, border = border,
## angle = angle, : the AREAS in the plot are wrong -- rather use 'freq =
## FALSE'
```



Boxplot

Consideriamo il campione dei valori assunti da una variabile quantitativa "Lavoro". Ordinando il vettore in ordine crescente, è possibile calcolare dei particolari valori chiamati *quartili*. Prende il nome di **primo quartile** e si indica con Q_1, il valore per il quale il 25% dei dati si trova alla sua sinistra e il restante 75% alla sua destra. In maniera speculare, il **terzo quartile**, indicato con Q_3 è il valore per il quale il 25% dei dati sono alla sua destra e il 75% alla sua sinistra. Il **secondo quartile** indicato con Q_2 è il valore che alla sua destra il 50% dei dati e alla sua sinistra il restante 50% ed è chiamato anche **mediana**. I valori Q_0 e Q_1 rappresentano rispettivamente il valore *minimo* e il valore *massimo* del campione. R fornisce le funzioni *quantile()* e *summary()* che ci permettono i valori del minimo (Q_0) e del massimo (Q_4), della mediana (Q_2), del primo e del terzo quartile (Q_1 e Q_3). Riferendoci al vettore *lavoro*:

```
quantile(lavoro)
```

```
##      0%     25%     50%     75%    100%
## 10.900 23.575 28.250 32.075 41.600
```

```r
summary(lavoro)
```

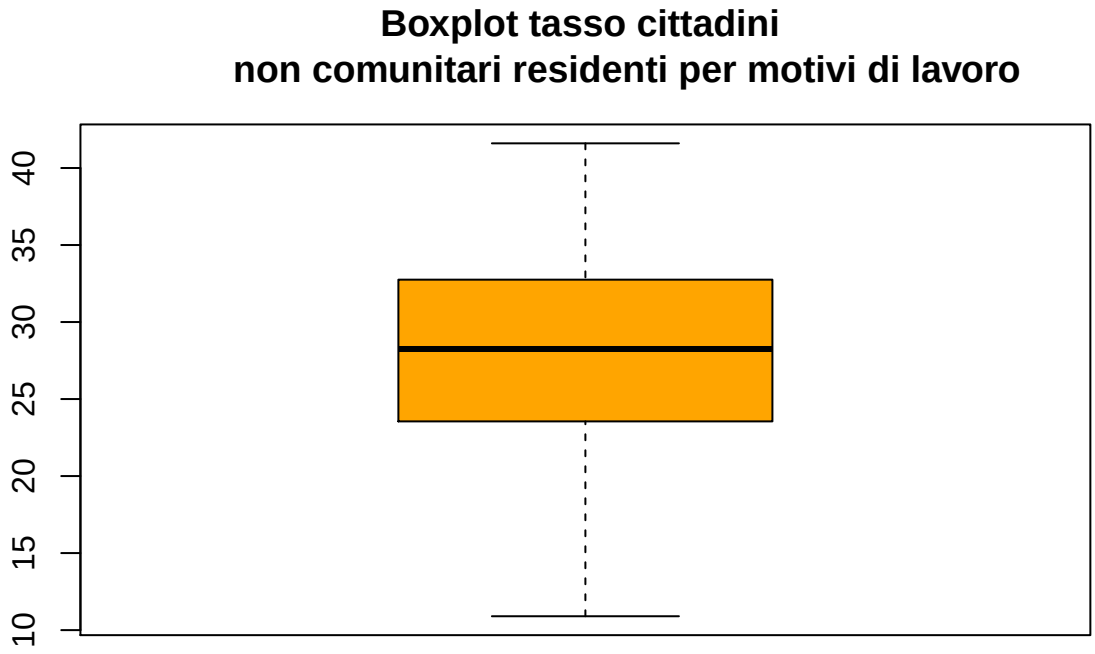
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   10.90   23.57   28.25   28.75   32.08   41.60
```

Calcolati i quartili è possibile definire cos'è un boxplot. Un **boxplot**, detto anche diagramma a scatola con baffi, è un grafico relativo a caratteri quantitativi ottenuto a partire dai quartili. Il suo scopo è quello di descrivere le caratteristiche salienti della distribuzione. Il disegno della scatola ha come estremi Q_1 e Q_3 ed ha una linea orizzontale in corrispondenza di Q_2. Sia in alto che in basso sono presenti due linee orizzontali dette *baffi*. Il baffo inferiore corrisponde al valore più piccolo fra le osservazioni ed è maggiore o uguale del valore Q_1 - $1.5 * (Q_3$ - $Q_1)$, mentre il baffo superiore corrisponde al valore più alto delle osservazioni ed è minore o uguale del valore $Q_3 + 1.5 * (Q_3$ - $Q_1)$. La distanza tra il primo e il terzo quartile è detta **scarto interquartile** o **intervallo interquartile**. Se tutti i valori cadono all'interno dell'intervallo (a,b) prendiamo il baffo inferiore come minimo e il baffo superiore come massimo. I numeri che non cadono nell'intervallo e risultano quindi essere fuori sono detti *valori anomali* o *outlier* e necessitano di uno studio per capirne le origini. Lo scopo del boxplot è quello di illustrare alcune caratteristiche di una distribuzione di frequenza come:

- la centralità
- la forma
- la dispersione
- la presenza di valori anomali

La centralità viene espressa dalla mediana. È possibile verificare la simmetria della distribuzione confrontando le differenze fra Q_3 - Q_2 e Q_2 - Q_1: se queste infatti risultano essere uguali o vicine allora la mediana si trova più o meno al centro della distribuzione, garantendo la simmetria dei dati. I valori dei dati che sono al di sopra del baffo superior o al di sotto del baffo inferiore sono outliers e nel grafico vengano rappresentati sottoforma di punti. Per la creazione di boxplot, R fornisce la funzione *boxplot()* . Il grafico può essere creato sia con orientamento verticale che orizzontale utilizzando il parametro *horizontal* che di default è impostato a FALSE, indicante orientamento verticale.

```r
boxplot(lavoro, col = "orange", main="Boxplot tasso cittadini
        non comunitari residenti per motivi di lavoro")
```



Come descritto precedentemente, gli estremi della scatola nel grafico sono rappresentati da $Q_3 = 32.08$ e da $Q_1 = 23.57$ mentre la linea orizzontale, la mediana, è in corrispondenza di $Q_2 = 28.25$. Il baffo inferiore è pari a 10.90 ovvero al primo numero maggiore o uguale a Q_1 - 1.5 $*$ (Q_3 - Q_1), 23.7 - 1.5 $*$ (32.08 - 23.57) = 10.93. Il baffo superiore invece è 41.60 ovvero al primo valore del campione minore o uguale di Q_3 + 1.5 $*$ (Q_3 - Q_1), ossia 32.08 + 1.5 $*$ (32.08 - 23.57) = 44.845. Tutti i valori ricadono fra il minimo e il massimo quindi non abbiamo anomalie. Per quanto riguarda la simmetria abbiamo che Q_3 - Q_2, 32.08 - 28.25 = 3.5, e Q_2 - Q_1, 28.25 - 23.57 = 4.68. I valori sono diversi ma molto vicini.

I boxplot per le restanti modalità sono:

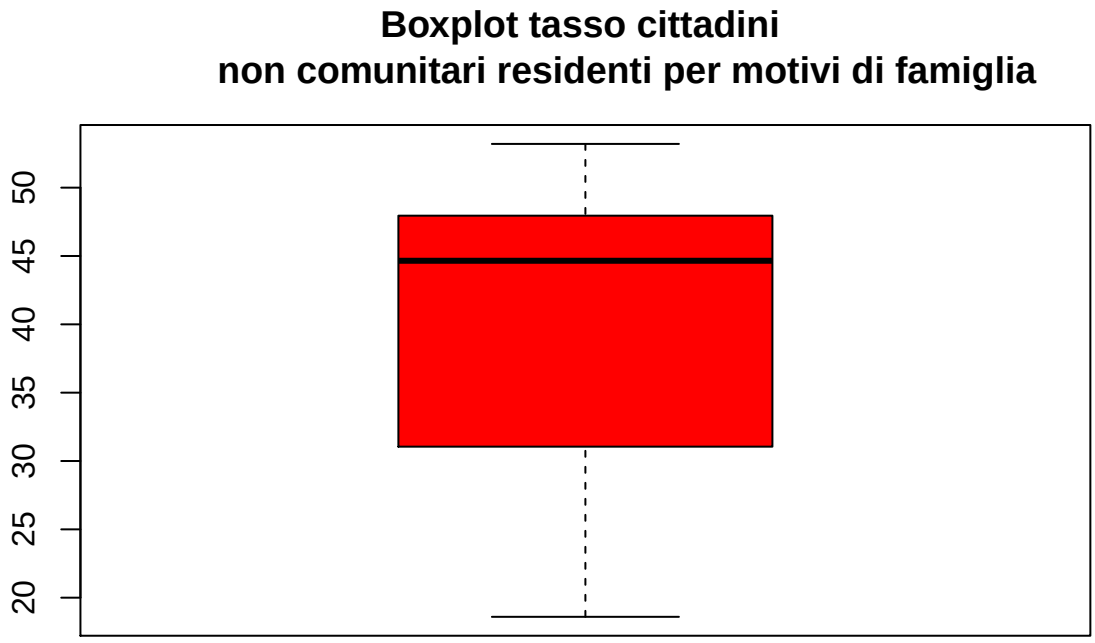
```
quantile(famiglia)
```

```
##      0%     25%     50%     75%    100%
## 18.600 31.925 44.650 47.625 53.200
```

```
summary(famiglia)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   18.60   31.93   44.65   40.10   47.62   53.20
```

```
boxplot(famiglia, col = "red", main="Boxplot tasso cittadini
        non comunitari residenti per motivi di famiglia")
```



In questo boxplot la mediana è spostata più verso l'alto, fattore che ci dice che la distribuzione non è simmetrica.

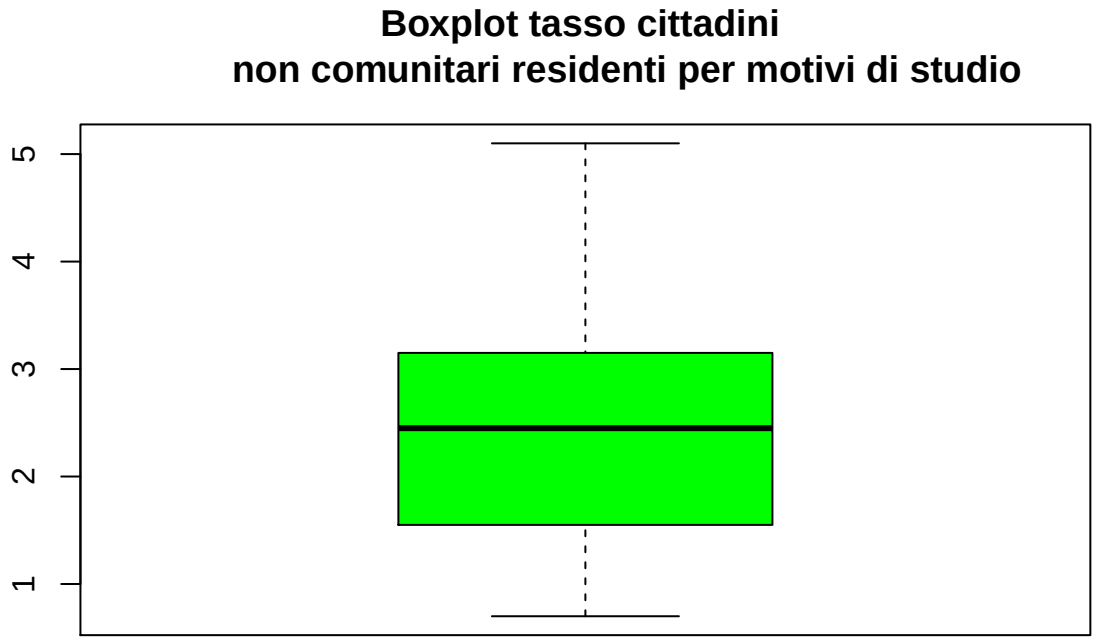
```
quantile(studio)
```

```
##    0%   25%   50%   75%  100%
## 0.700 1.575 2.450 3.125 5.100
```

```
summary(studio)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.700   1.575   2.450   2.505   3.125   5.100
```

```
boxplot(studio, col = "green", main="Boxplot tasso cittadini
        non comunitari residenti per motivi di studio")
```



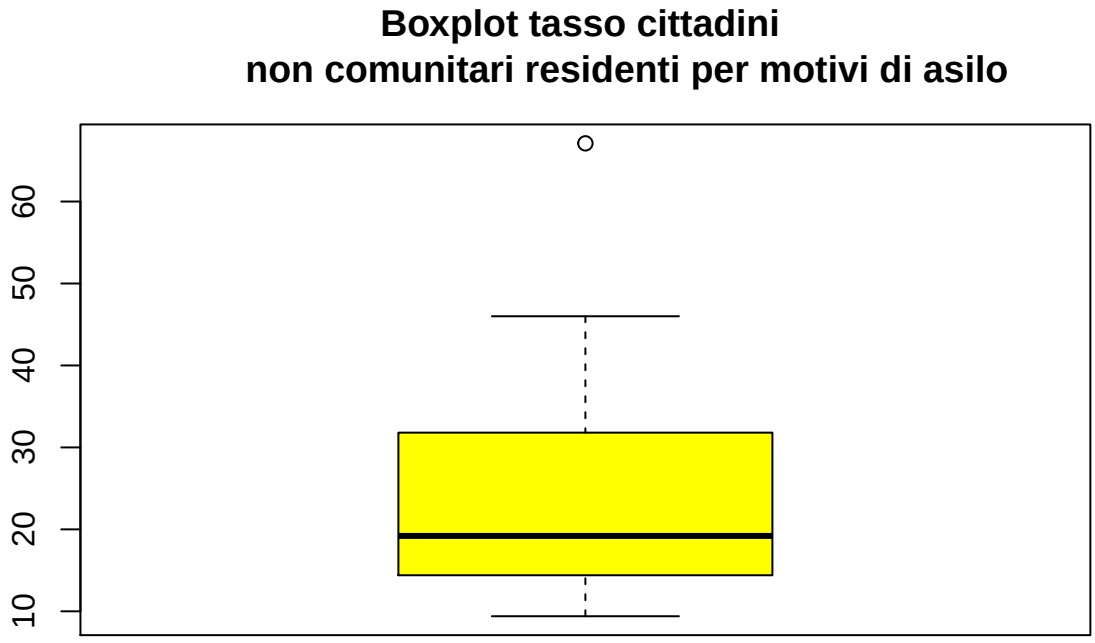

```
quantile(asilo)
```

```
##    0%   25%   50%   75%  100%
##  9.40 14.45 19.20 31.40 67.10
```

```
summary(asilo)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    9.40   14.45   19.20   25.12   31.40   67.10
```

```
boxplot(asilo, col = "yellow",main="Boxplot tasso cittadini
        non comunitari residenti per motivi di asilo")
```



In questo boxplot è presente un valore anomale e, riferendoci alla tabella, scopriamo che è il Molise con 67.1. I baffo superiore è molto più grande di quello inferiore, dicendoci che la disperisione dei dati è maggiore. Anche la mediana non risulta essere al centro ma tende verso il basso incidendo sulla simmetria.

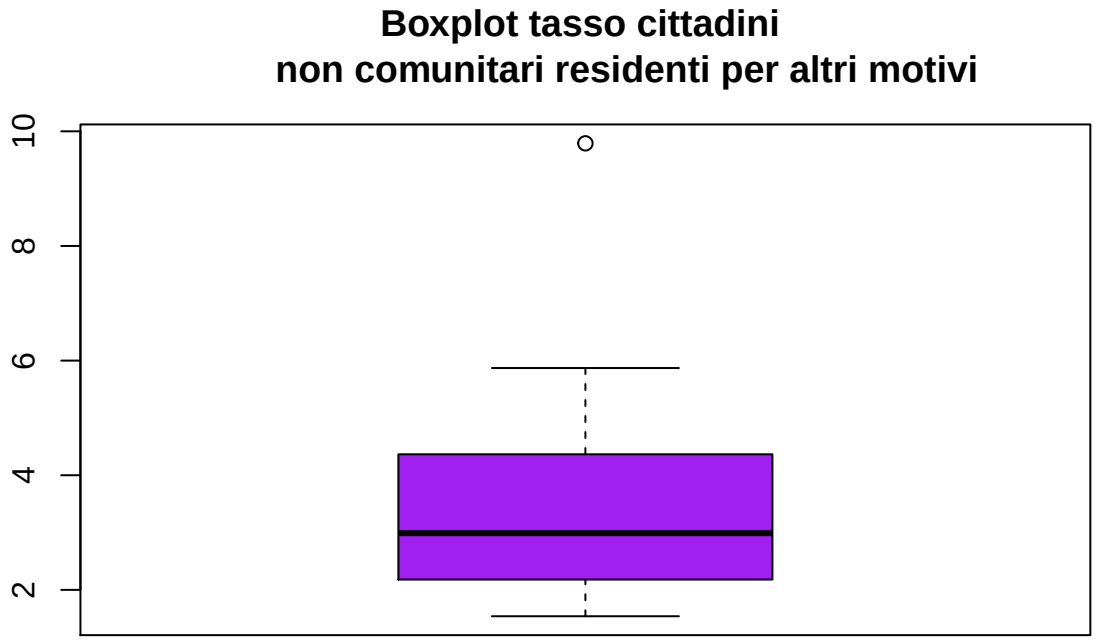
```
quantile(altro)
```

```
##     0%    25%    50%    75%   100%
## 1.5400 2.1900 2.9900 4.3475 9.7900
```

```
summary(altro)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.540   2.190   2.990   3.526   4.348   9.790
```

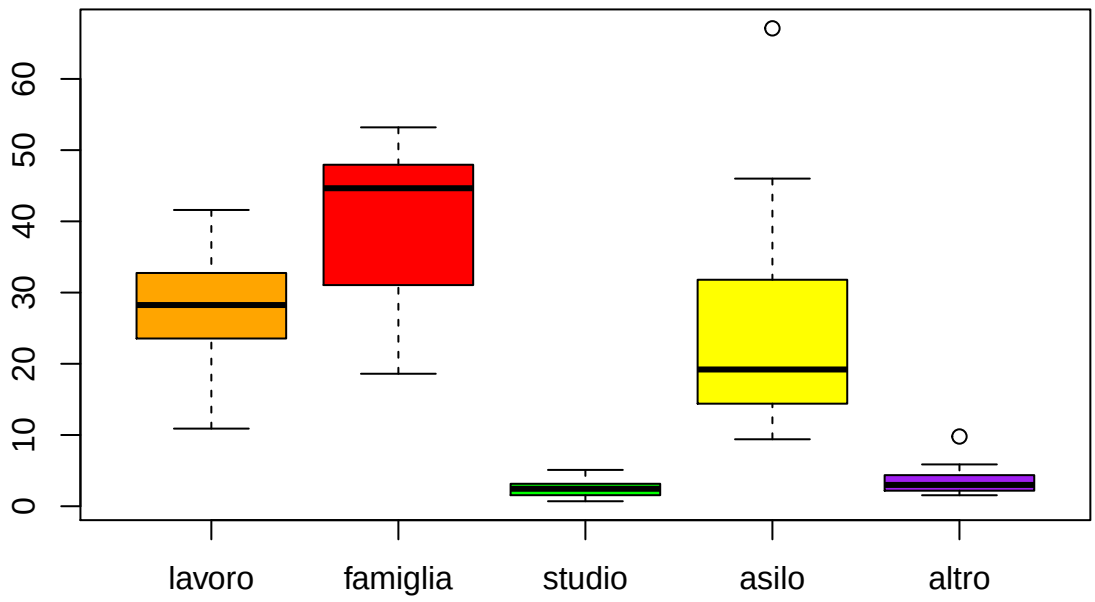
```
boxplot(altro, col = "purple", main="Boxplot tasso cittadini
        non comunitari residenti per altri motivi")
```



Anche in quest'ultimo boxplot è presente un valore anomale che esce al di fuori dell'intervallo. In questo caso, la regione è il Lazio con 9.79. La mediana risulta essere più in basso rispetto al centro.

Possiamo anche confrontare i diversi boxplot mettendoli tutti insieme in un unico grafico.

```
boxplot(lavoro, famiglia, studio, asilo, altro,
        col = c("orange","red","green","yellow", "purple"),
        names = c("lavoro", "famiglia", "studio", "asilo", "altro"))
```



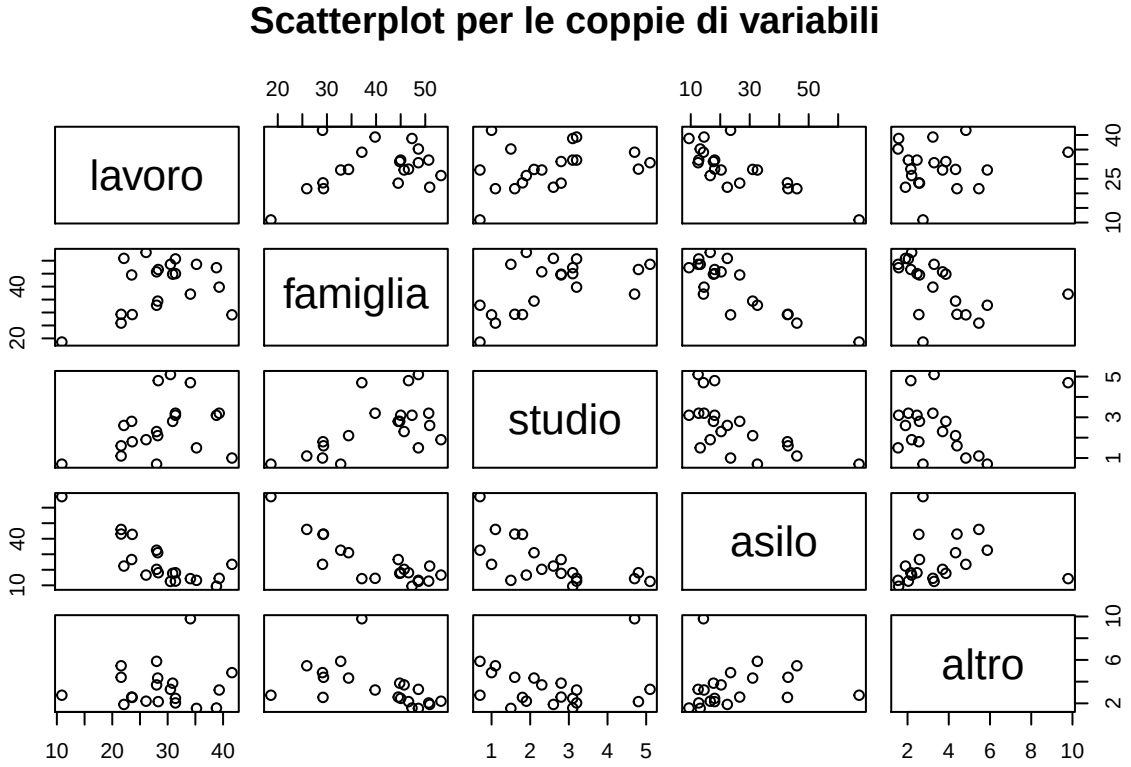
Scatterplot

Gli **scatterplot** o **diagrammi di dispersione** consistono in rappresentazioni grafiche delle relazioni tra variabili quantitative. Quello che si fa è fissare la variabile da porre sull'asse delle ascisse e quella da porre sull'asse delle ordinate. Queste variabili vengono chiamate rispettivamente variabile **indipendente** e variabile **dipendente**. Iniziamo col creare quindi un data frame, un tipo di struttura dati a disposizione in R. All'interno di questo data frame inseriamo i vettori delle diverse modalità.

```
motivi <- data.frame(lavoro, famiglia, studio, asilo, altro)
```

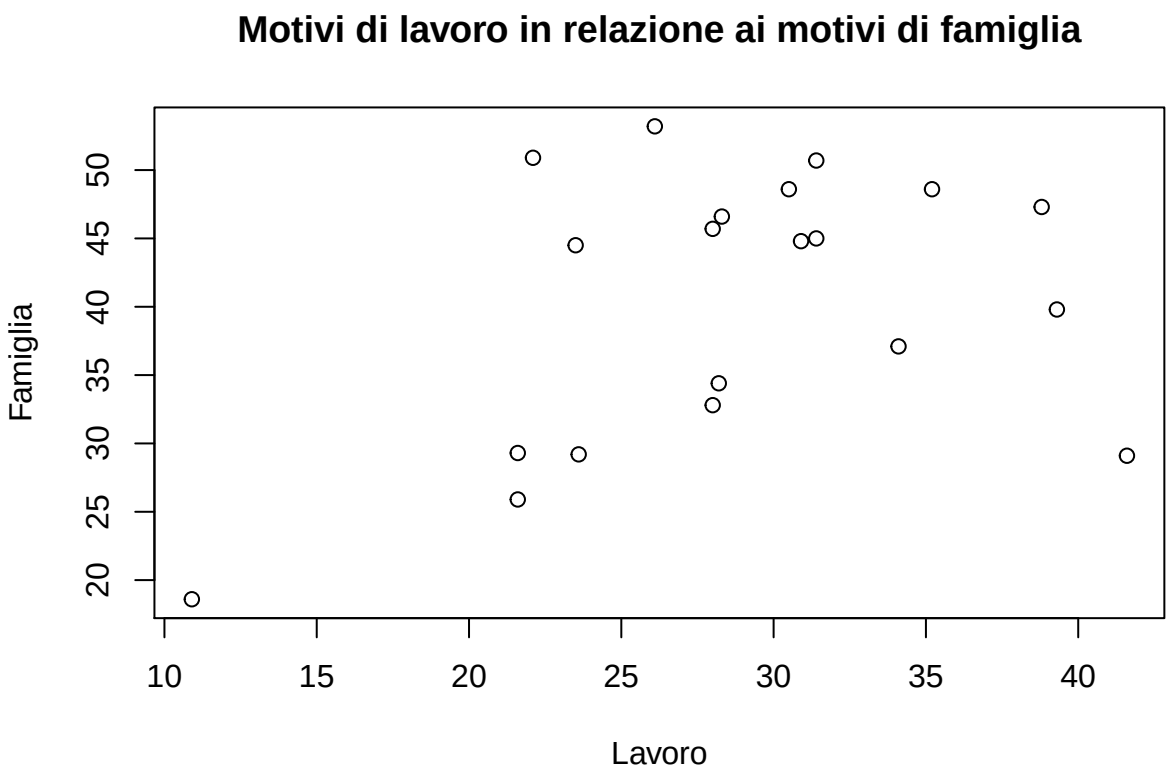
Per realizzare uno scatterplot basta utilizzare la funzione *pairs()* darle in input il data frame appena creato.

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



Le immagini mostrano le nuvole di punti ottenute considerando tutte le coppie di variabili. Almeno nella metà delle coppie, è presente una qualche relazione, anche accennata. Alcune invece, come il plot che mette in relazione lavoro e famiglia, non sembrano essere in relazione. È possibile anche stampare uno solo grafico alla volta utilizzando la funzione *plot()* dandole in input singoli vettori del data frame.

```
plot(motivi$lavoro, motivi$famiglia, xlab="Lavoro",
     ylab="Famiglia",
     main="Motivi di lavoro in relazione ai motivi di famiglia")
```


Motivi di lavoro in relazione ai motivi di famiglia
Famiglia
20
25
30
35
40
45
50
10
15
20
25
30
35
40
Lavoro

Statistica descrittiva univariata

Per i fenomeni quantitativi è utile definire la funzione di distribuzione empirica. Abbiamo due tipi di funzione di distribuzione empirica:

- funzione di distribuzione empirica **discreta**
- funzione di distribuzione empirica **continua**

Funzione di distribuzione empirica discreta

Una **funzione di distribuzione empirica discreta** è una funzione utilizzata quando la nostra variabile assume valori discreti. Il fatto che i valori della variabile siano discreti, porta la funzione ad assumere una forma a gradini. Utilizziamo le frequenze relative $f_i = \frac{n_i}{n}$ con i = 1, 2, . . . , k e le frequenze relative cumulate $F_i = f_1 + f_2 + \ldots + f_k$ dove k è il numero discreto di valori che il nostro campione di dati può assumere. La funzione di distribuzione empirica discreta è così definita:

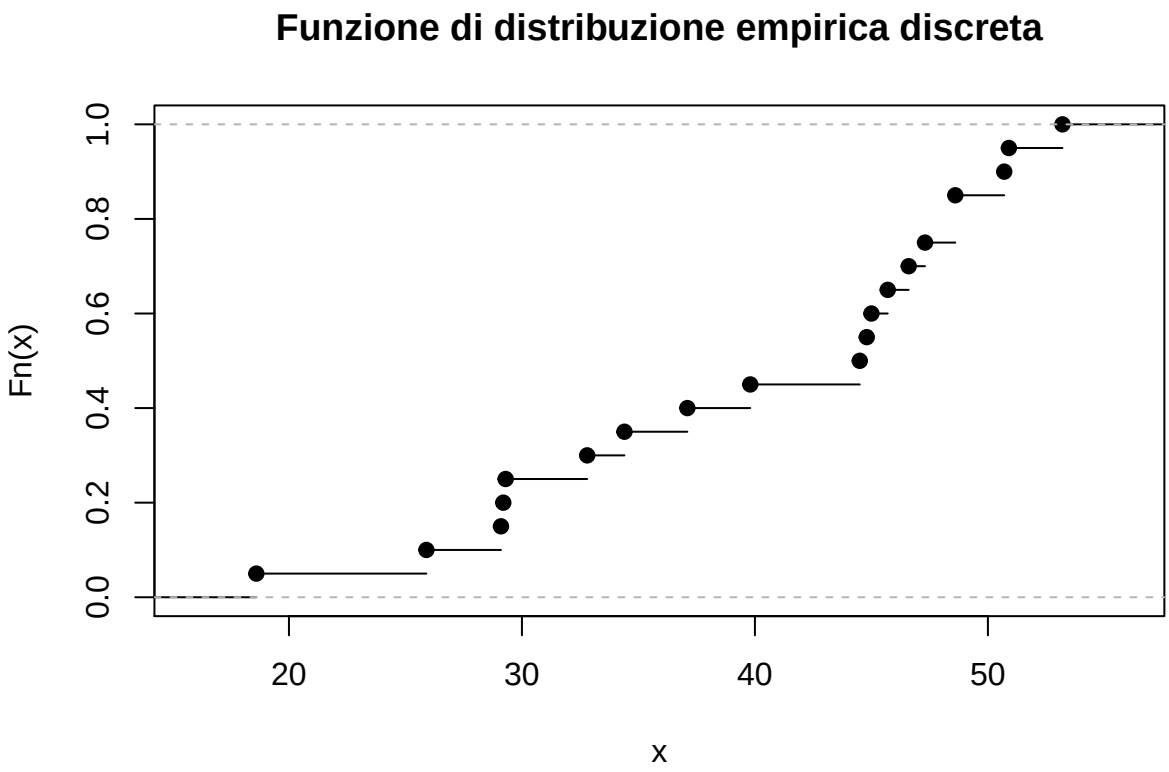
$$F(x) = \frac{\#\{x_i \leq x, i = 1, 2, ..., n\}}{n} = \begin{cases} 0, & x < z_1 \\ F_1, & z_1 \leq x < z_2 \\ ... & \\ F_i, & z_i \leq x < z_{i+1} \\ ... & \\ 1, & x \geq x_k \end{cases}$$

dove:

- $\#$ è la cardinalità dell'insieme
- gli z_i rappresentano i possibili valori assunti dal campione
- gli x_i i valori del campione
- gli F_i la proporzione dei dati del campione minori o uguali di z_i

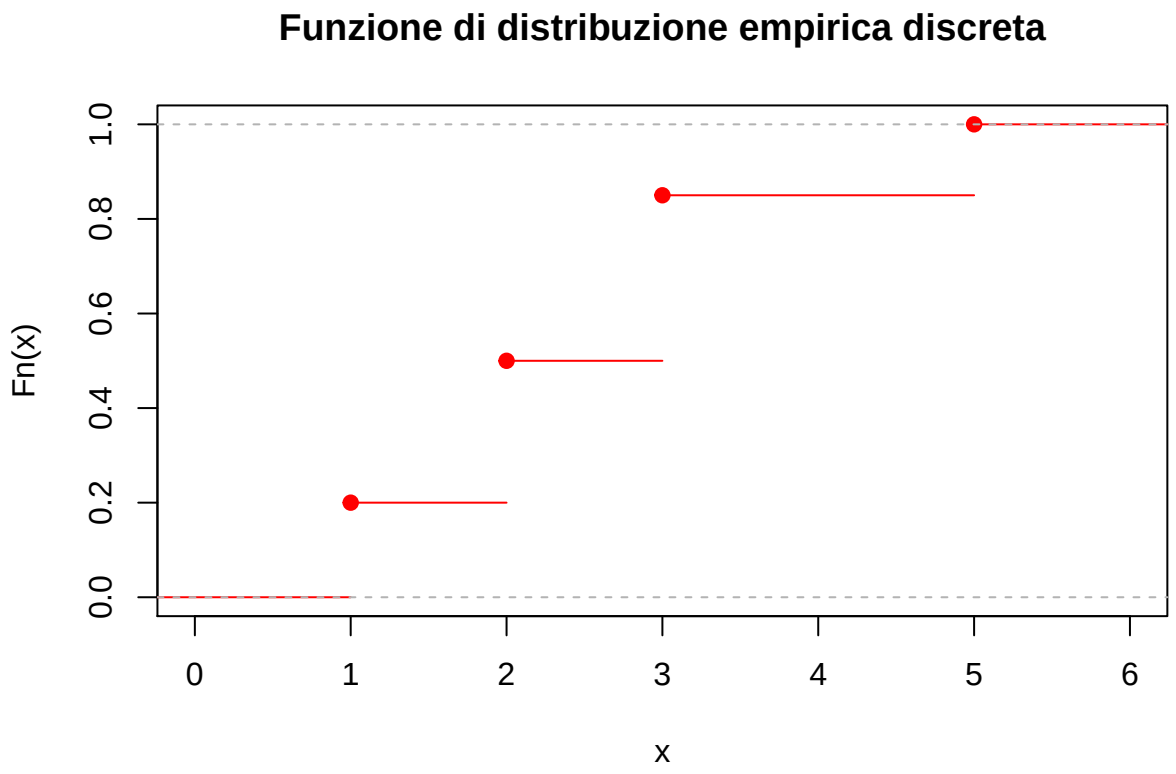
La funzione di distribuzione empirica discreta F(x) è definita per ogni x reale. R dispone della classe *stepfun* che implementa una serie di metodi per trattare funzioni a gradino. In particolare, la funzione *ecdf()*, acronimo di *empirical cumulative distribution function*, permette di disegnare il grafico della funzione di distribuzione empirica per le variabili quantitative discrete.

```
plot(ecdf(famiglia), main = "Funzione di distribuzione empirica discreta")
```



I gradini sono tanti quanto i valori assunti dalla variabile e, siccome in questo vettore i numeri sono tutti differenti tranne che per una coppia, ha poco senso creare questo grafico. Prendendo il vettore studio però ed eliminando la parte decimale è possibile visualizzare un esempio di questo grafico.

```
plot(ecdf(round(studio, digits = 0)),
     main="Funzione di distribuzione empirica discreta", col="red")
```



La funzione *round()* serve ad arrotondare un numero alla cifra specificata con il parametro *digits*.

Funzione di distribuzione empirica continua

Per fenomeni quantitativi continui, la funzione di distribuzione empirica è una funzione **continua**, necessaria se i dati vengono raccolti in classi. Se i dati vengono quindi raccolti in k distinte classi $C_1 = [z_1, z_2)$, $C_2 = [z_3, z_4)$, ..., $C_k = [z_k, z_{k+1}]$, la funzione di distribuzione empirica è così definita:

$$F(x) = \begin{cases} 0, & x < z_1 \\ \dots & \\ F_i, & x = z_i \\ \frac{F_{i+1}-F_i}{z_{i+1}-z_i}x + \frac{z_{i+1}F_i - z_i F_{i+1}}{z_{i+1}-z_i}, & z_i < x < z_{i+1}, \quad z_i < x < z_{i+1} \\ F_{i+1}, & x = z_{i+1} \\ \dots & \\ 1, & x \geq z_{k+1} \end{cases}$$

Si noti che F(x) = 0 per $x < z_1$, F(x) = 1 per $x \leq z_{k+1}$ mentre se $z_i < x < z_{i+1}$ la funzione empirica coincide con il segmento che passa per i punti (z_i, F_i) e (z_{i+1}, F_{i+1})

La suddivisione in classi adottata è la seguente: [0, 15), [15, 20), [20, 25)

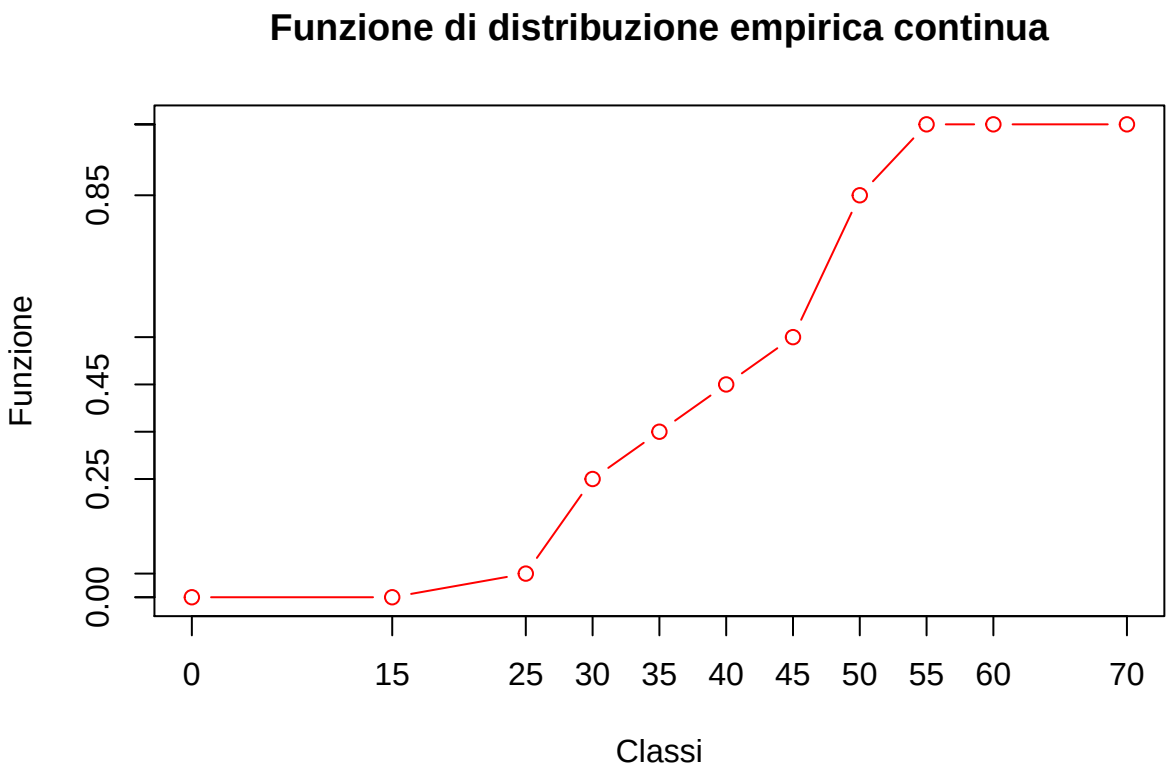
```
classi_empiriche <- c(0,15,25,30,35,40,45,50,55,60,70)

Ffamiglia <- cumsum(table
                    (cut
                      (famiglia, breaks=classi_empiriche, right =FALSE)))/length(famigli
Ffamiglia
```

```
##  [0,15) [15,25) [25,30) [30,35) [35,40) [40,45) [45,50) [50,55) [55,60)
##    0.00    0.05    0.25    0.35    0.45    0.55    0.85    1.00    1.00
## [60,70)
##    1.00
```

Il vettore Fi contiene le frequenze relative cumulate dei tassi delle varie regioni associati alle varie classi.

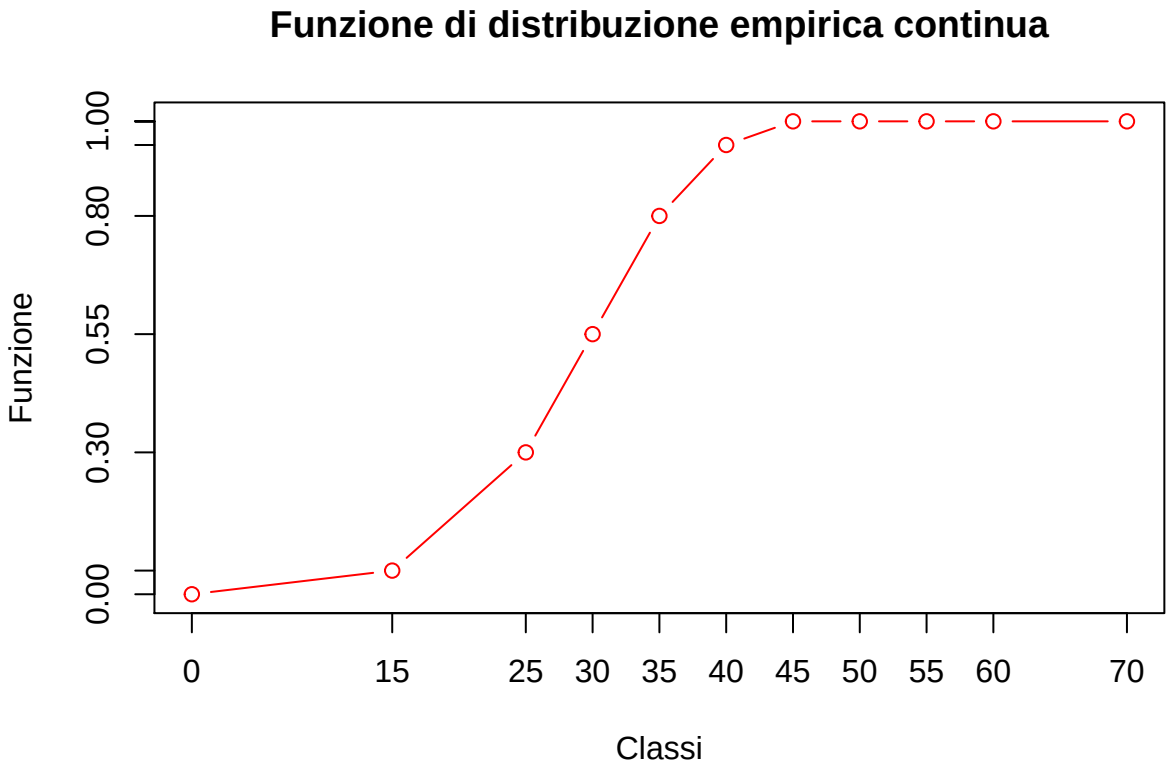
```
Ffamiglia <- c(0, Ffamiglia)
plot(classi_empiriche, Ffamiglia, type = "b",
     axes = FALSE, main = "Funzione di distribuzione empirica continua",
     ylab ="Funzione", xlab = "Classi", col = "red")
axis(1, classi_empiriche)
axis(2, format(Ffamiglia, digits=2))
box()
```



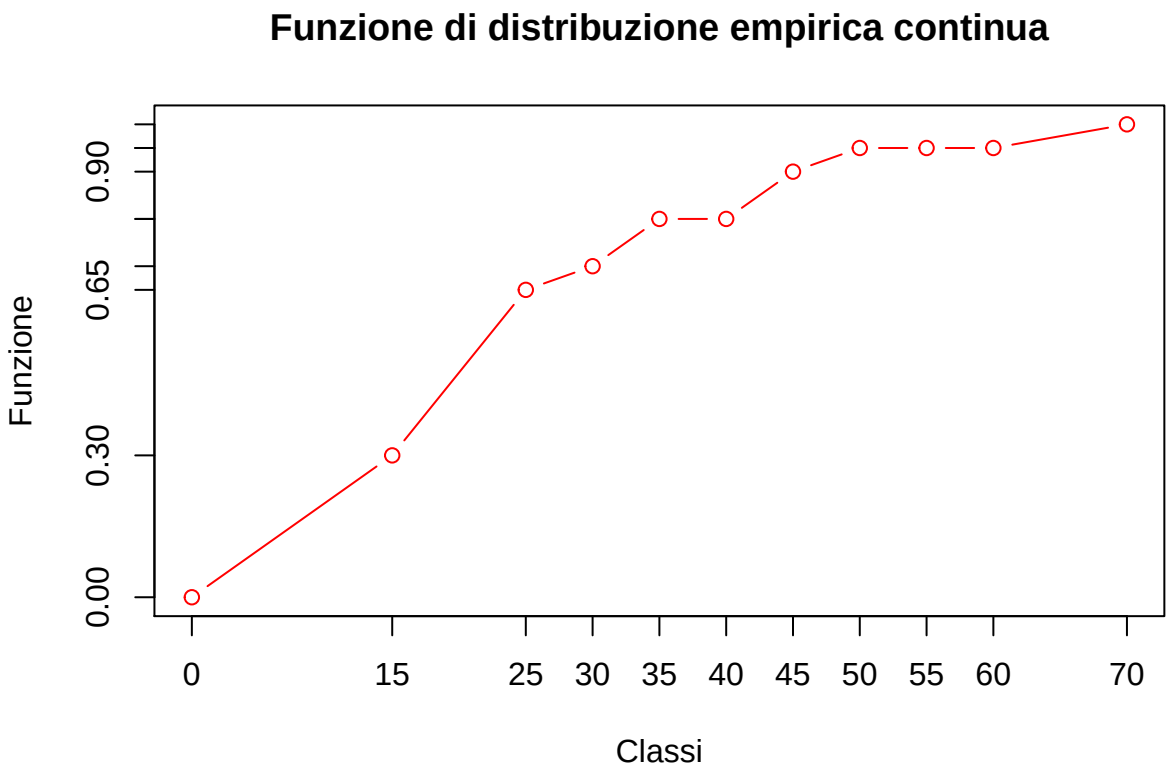
Utilizziamo $c(0, F_i)$ per aggiungere uno zero all'inizio del vettore delle frequenze relative cumulate. Nella funzione *plot()* abbiamo utilizzato l'opzione *type* = *b* che ci consente di congiungere i punti successivi del grafico mediante delle linee continue ai cui estremi troviamo dei piccoli cerchi. L'opzione *axes* = *FALSE* consente di non tracciare gli assi. Mediante l'opzione *axis(1, classi)* si disegna l'asse orizzontale in basso e con *axis(2, format(Fi, digits=2))* si ottiene l'asse verticale di sinistra con una formattazione opportuna dei numeri. Con *box()* racchiudiamo tutto in un rettangolo.

Utilizzando la stessa suddivisione in classi, creiamo il grafico per i vettori asilo e lavoro:

```
Flavoro <- cumsum(table
                  (cut
                    (lavoro, breaks=classi_empiriche, right =FALSE)))/length(lavoro)
Flavoro <- c(0, Flavoro)
plot(classi_empiriche, Flavoro, type = "b",
     axes = FALSE, main = "Funzione di distribuzione empirica continua",
     ylab ="Funzione", xlab = "Classi", col = "red")
axis(1, classi_empiriche)
axis(2, format(Flavoro, digits=2))
box()
```

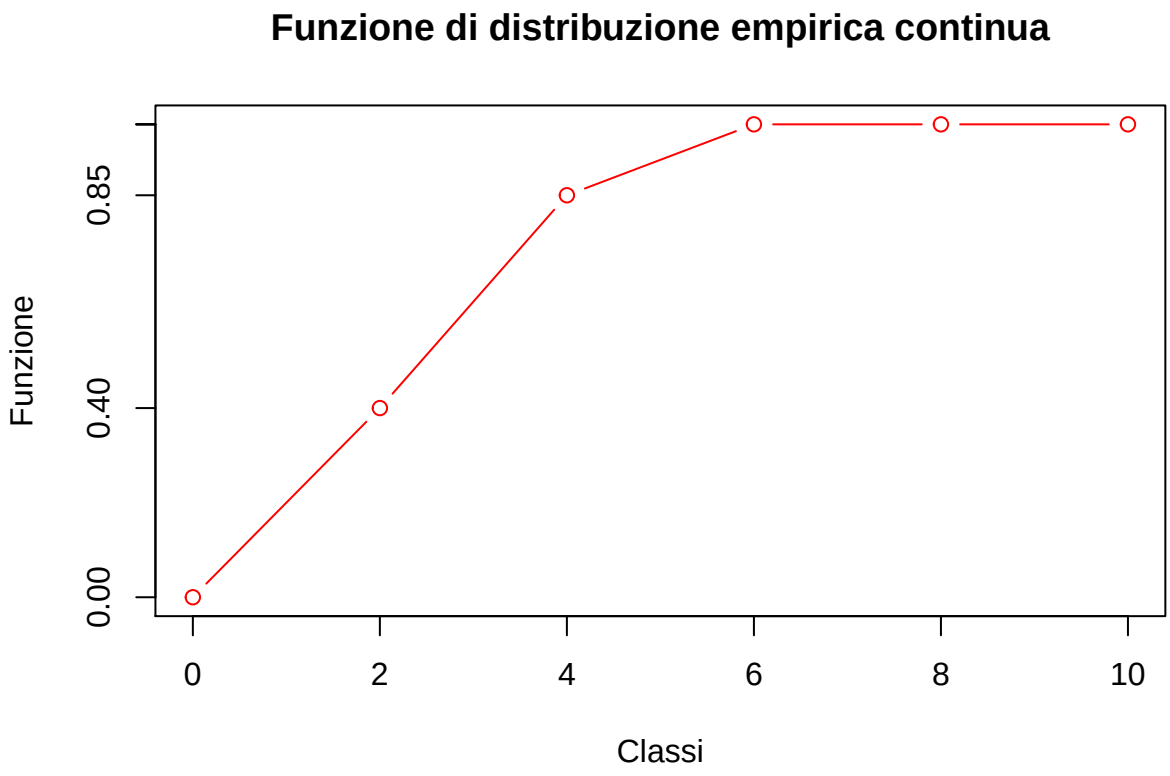



```
Fasilo <- cumsum(table
                 (cut
                   (asilo, breaks=classi_empiriche, right =FALSE)))/length(asilo)
Fasilo <- c(0, Fasilo)
plot(classi_empiriche, Fasilo, type = "b",
     axes = FALSE, main = "Funzione di distribuzione empirica continua",
     ylab ="Funzione", xlab = "Classi", col = "red")
axis(1, classi_empiriche)
axis(2, format(Fasilo, digits=2))
box()
```

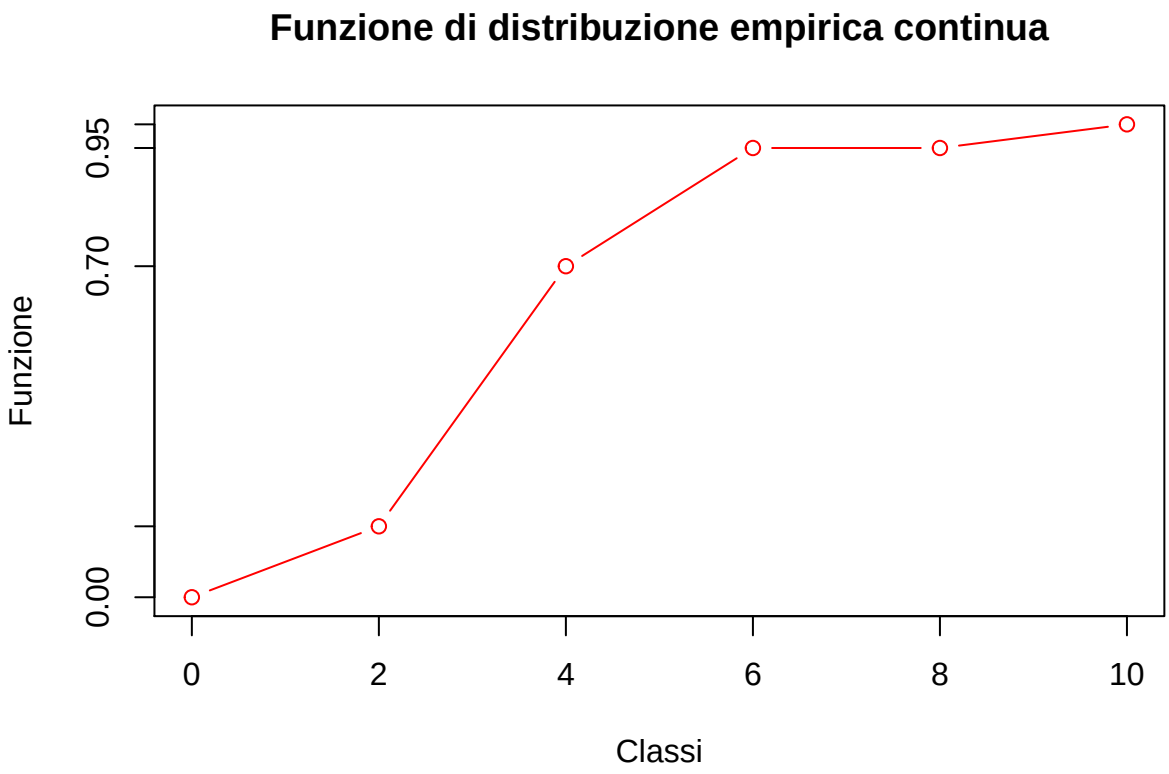


Siccome i vettori studio e altro risultano contenere valori decisamente piccoli, utilizziamo una suddivisione in classi diversa:

```
classi_empiriche1 <- c(0,2,4,6,8,10)
Fstudio <- cumsum(table
                  (cut
                    (studio, breaks=classi_empiriche1, right =FALSE)))/length(studio)
Fstudio <- c(0, Fstudio)
plot(classi_empiriche1, Fstudio, type = "b",
     axes = FALSE, main = "Funzione di distribuzione empirica continua",
     ylab ="Funzione", xlab = "Classi", col = "red")
axis(1, classi_empiriche1)
axis(2, format(Fstudio, digits=2))
box()
```




```
Faltro <- cumsum(table
                 (cut
                   (altro, breaks=classi_empiriche1, right =FALSE)))/length(altro)
Faltro <- c(0, Faltro)
plot(classi_empiriche1, Faltro, type = "b",
     axes = FALSE, main = "Funzione di distribuzione empirica continua",
     ylab ="Funzione", xlab = "Classi", col = "red")
axis(1, classi_empiriche1)
axis(2, format(Faltro, digits=2))
box()
```



Indici di sintesi

Come visto prima, attraverso i boxplot è possibile rappresentare in maniera semplice la distribuzione di frequenza di un campione di dati numerico. Quello che faremo adesso è introdurre degli indici che servono a misurare quantitativamente alcune delle caratteristiche osservate nei boxplot e nei grafici di distribuzioni di frequenza. Gli indici di **sintesi**, detti anche *statistiche*, sono utili nel descrivere dati numerici. In particolare questi indici sono: *media*, *moda*, *mediana*, *varianza*, *deviazione standard* e *coefficiente di variazione*. Le prime tre, media, mediana e moda, sono *misure di centralità* mentre le restanti tre misurano la *dispersione dei dati*.

Media, mediana e moda

Media

Supponiamo di avere a disposizione un insieme $x_1, x_2, ..., x_n$ di n valori, chiamato *campione di ampiezza n*. La **media campionaria**, indicata con $\bar{x}$ è la media aritmetica di questi valori.

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

Questa misura gode della proprietà di linearità. Inoltre, è possibile vedere la media come una media pesata dei valori distinti assunti dai dati. Ogni valore distinto usa come peso la sua frequenza relativa, ovvero la frazione dei dati uguale al suo valore.

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

$s_i = x_i - \bar{x}$ con $(i = 1, 2, ..., n)$

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

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

Va sottolineato che la media campionaria è sensibile ai valori anomali.

Per il calcolo della media campionaria, R fornisce la funzione *mean()*

```
mean(lavoro)
```

```
## [1] 28.755
```

```
mean(famiglia)
```

```
## [1] 40.105
```

```
mean(studio)
```

```
## [1] 2.505
```

```
mean(asilo)
```

```
## [1] 25.12
```

```
mean(altro)
```

```
## [1] 3.5255
```

Mediana

Una seconda statistica che indica la centralità di un insieme di dati è la **mediana campionaria**. Assegnato un insieme di dati di ampiezza n, lo si ordini dal minore al maggiore. Se n è dispari, si definisce mediana campionaria il valore che è in posizione $\frac{(n+1)}{2}$, mentre se n risulta essere pari la mediana campionaria è definita come la media aritmetica dei valori che occupano le posizioni $\frac{n}{2}$ e $\frac{(n+1)}{2}$. Questa definizione di mediana campionaria bipartisce le osservazioni in due gruppi di uguale numerosità. Il linguaggio R ci mette a disposizione la funzione *median()* per calcolare la mediana di un vettore contenente delle osservazioni. Di seguito, la funzione *median()* viene utilizzata per calcolare la mediana dei vettori a nostra disposizione:

```
median(lavoro)
```

```
## [1] 28.25
```

```
median(famiglia)
```

```
## [1] 44.65
```

```
median(studio)
```

```
## [1] 2.45
```

```
median(asilo)
```

```
## [1] 19.2
```

```
median(altro)
```

```
## [1] 2.99
```

Sia la media campionaria che la mediana campionaria risultano essere statistiche utili per descrivere la centralità dei dati. A dispetto però della media campionaria, la mediana dipende solo da uno o due dei valori centrali dei dati e non risente dei valori estremi (estremamente bassi o alti). Lo svantaggio di utilizzare la mediana è che bisogna prima ordinare i dati a dispozione prima di poterla calcolare.

Moda

La terza statistica utilizzata per descrivere la centralità dei dati è la **moda campionaria**. La moda campionaria di un insieme di dati, se esiste, è la modalità a cui è associata la frequenza (assoluta o relativa) più elevata. Se esistono più modalità con frequenza massima, ciascuna di esse è detta **valore modale**. La moda quindi nient'altro è che il valore che si presenta con frequenza maggiore nell'insieme dei dati. Per definizione, è possibile utilizzare la moda anche per dati di tipo qualitativo, cosa che non è possibile fare con media e mediana. La moda può non esistere oppure non essere unica: quando non è unica, la distribuzione è detta *unimodale* mentre quando ci sono due o più mode diverse è detta *bimodale* o *multimodale*. Non esiste in R una funzione per estrarre la moda da una distribuzione di dati poichè essa risulta facilmente estraibile osservando il grafico delle frequenze assolute. Si è optato lo stesso per definire una funzione in grado di ricavarla:

```
moda <- function(v){
  y<-table(v)
  z<-which(y==max(y))
  return(c(z))
}
```

Applichiamo poi la moda a ciascun vettore della matrice:

```
moda(lavoro)
```

```
## 21.6   28 31.4
##    2    7   12
```

```
moda(famiglia)
```

```
## 48.6
##   16
```

```
moda(studio)
```

```
## 0.7 2.8 3.1 3.2
##   1  11  12  13
```

```
moda(asilo)
```

```
## 18.1
##    9
```

```
moda(altro)
```

```
## 1.54 1.57  1.9 2.04 2.16  2.2 2.46 2.55 2.58 2.75 3.23 3.29  3.7 3.86 4.33
##    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15
##  4.4 4.83 5.46 5.87 9.79
##   16   17   18   19   20
```

In quasi tutti i vettori considerati, la moda è risultata essere una buona statistica per misurare la centralità tranne nell'ultimo caso, col vettore *altro*, dove i dati sono molto diversi fra loro.

Dopo aver calcolato queste tre statistiche, è possibile descrivere la forma di una distribuzione confrontando la media campionaria e la mediana campionaria. Se queste due statistiche risultano essere uguali allora la distribuzione di frequenza tende ad essere simmetrica; se la media campionaria è maggiore della mediana campionaria allora la distribuzione sarà sbilanciata verso destra; se invece la media campionaria è sensibilimente minore della mediana campionaria allora la distribuzione di frequenza è più sbilanciata verso sinistra.

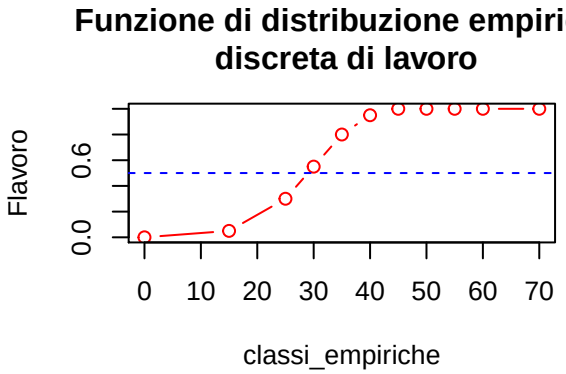
Mediana per distribuzione di frequenza

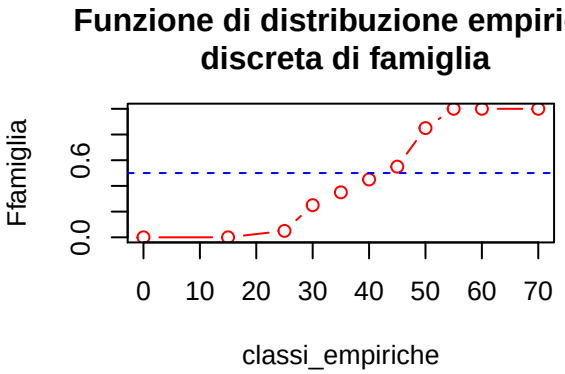
Un modo di procedere diverso per definire la mediana è quello di considerare le frequenze relative cumulate. Sia X una variabile quantitativa e siano $z_1 < z_2 < ... < z_k$. Considerato un campione $(x_1, x_2, ..., x_n)$, siano $F_i = f_1 + f_2 + ... + f_i$ con $(i = 1, 2, ..., k)$ le frequenze relative cumulate.

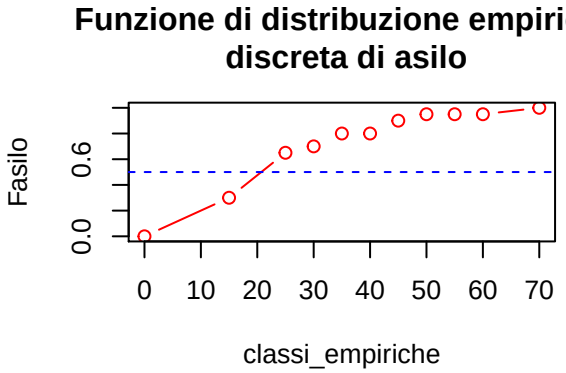
La **mediana per una distribuzione di frequenze** è definita come la modalità i-esima $(i = 1, 2, ..., k)$ che soddisfa la doppia disuguaglianza: $F_{i-1} < 0.5$, $F_i \geq 0.5$

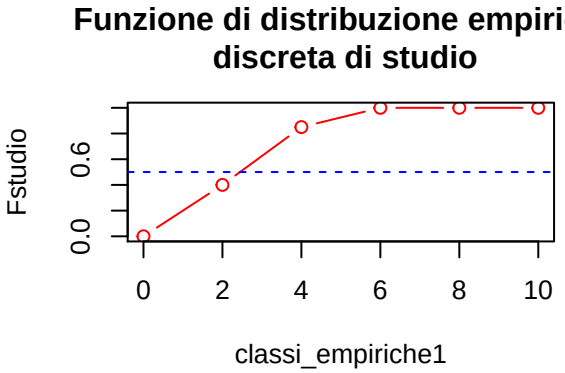
Come si evince dalla definzione, la mediana di una distribuzione di frequenza è un valore di sintesi che indica un punto centrale intorno al quale si dispone la distribuzione di frequenza.

```
par(mfrow = c(2,2))
plot(classi_empiriche, Flavoro, type="b",
     main="Funzione di distribuzione empirica \n discreta di lavoro", col="red")
abline(h=0.5, lty=2, col="blue")
plot(classi_empiriche, Ffamiglia, type="b",
     main="Funzione di distribuzione empirica \n discreta di famiglia", col="red")
abline(h=0.5, lty=2, col="blue")
plot(classi_empiriche, Fasilo, type="b",
     main="Funzione di distribuzione empirica \n discreta di asilo", col="red")
abline(h=0.5, lty=2, col="blue")
plot(classi_empiriche1, Fstudio, type="b",
     main="Funzione di distribuzione empirica \n discreta di studio", col="red")
abline(h=0.5, lty=2, col="blue")
```

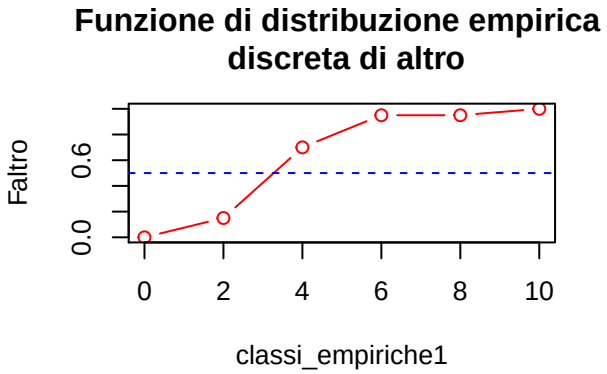









```
plot(classi_empiriche1, Faltro, type="b",
     main="Funzione di distribuzione empirica \n discreta di altro", col="red")
abline(h=0.5, lty=2, col="blue")
```



Per ottenere il valore della mediana per la distribuzione di frequenze utilizziamo la funzione *quantile()*.

```
quantile(lavoro, 0.5, type=1)
```

```
##  50%
## 28.2
```

```
quantile(famiglia, 0.5, type=1)
```

```
##  50%
## 44.5
```

```
quantile(studio, 0.5, type=1)
```

```
## 50%
## 2.3
```

```r
quantile(asilo, 0.5, type=1)
```

```
##  50%
## 18.1
```

```r
quantile(altro, 0.5, type=1)
```

```
##  50%
## 2.75
```

Quantili, percentili, decili e quartili

Oltre la mediana è possibile definire altri indici di posizione detti **quantili**, che dividono l'insieme dei dati ordinati in un numero di parti uguali. Supponiamo di avere a disposizione una variabile quantitativa con un certo numero di osservazioni disposte in ordine crescente. Possiamo dividere i dati in α gruppi, ognuno contenente circa lo stesso numero di elementi. Gli $\alpha - 1$ numeri sono detti *quantili* di ordine α. Solitamente possiamo dividere i dati in $\alpha = 4$ gruppi da 3 quantili detti *quartili*, oppure $\alpha = 10$ e 9 quantili detti *decili* o ancora $\alpha = 100$ e 99 quantili detti *percentili*. I quantili (percentili) sono indici di posizione non centrali utilizzati per insiemi numerosi di dati. Per il calcolo dei quantili R ha a disposizione la funzione *quantile(v, probs = , type =)*. Oltre al vettore, la funzione prende in input anche due ulteriori parametri:

- *probs* che rappresenta il vettore delle probabilità
- *type* a cui è possibile assegnare valori da 1 a 9 e indica l'algoritmo da utilzzare per il calcolo dei quantili. Infatti, R possiede per 9 algoritmi per il calcolo dei quantili e quello di default è il 7, basato su tecniche di interpolazione tra i punti.

```r
quantile(lavoro)
```

```
##     0%    25%    50%    75%   100%
## 10.900 23.575 28.250 32.075 41.600
```

```r
quantile(famiglia)
```

```
##     0%    25%    50%    75%   100%
## 18.600 31.925 44.650 47.625 53.200
```

```r
quantile(studio)
```

```
##    0%   25%   50%   75%  100%
## 0.700 1.575 2.450 3.125 5.100
```

```r
quantile(asilo)
```

```
##    0%   25%   50%   75%  100%
##  9.40 14.45 19.20 31.40 67.10
```

```r
quantile(altro)
```

```
##     0%    25%    50%    75%   100%
## 1.5400 2.1900 2.9900 4.3475 9.7900
```

Abbiamo detto che R è in grado di utilizzare diversi algoritmi per il calcolo dei quantili. Mostriamo ora lavorando sul vettore famiglia, la differenza fra i vari algoritmi

```r
tipiquartili <- function(x){
  y<-numeric(0)
  for(i in 1:9){
    y<-rbind(y, c(quantile(x, 0, type=i), quantile(x, 0.25, type=i),
                  quantile(x, 0.5, type=i), quantile(x, 0.75, type=i),
                  quantile(x, 1, type=i)))
  }
  rownames(y) <- paste("type", 1:9)
  y
}
tipiquartili(famiglia)
```

```
##          0%      25%   50%      75% 100%
## type 1 18.6 29.30000 44.50 47.30000 53.2
## type 2 18.6 31.05000 44.65 47.95000 53.2
## type 3 18.6 29.30000 44.50 47.30000 53.2
## type 4 18.6 29.30000 44.50 47.30000 53.2
## type 5 18.6 31.05000 44.65 47.95000 53.2
## type 6 18.6 30.17500 44.65 48.27500 53.2
## type 7 18.6 31.92500 44.65 47.62500 53.2
## type 8 18.6 30.75833 44.65 48.05833 53.2
## type 9 18.6 30.83125 44.65 48.03125 53.2
```

Calcolando i quantili sul vettore famiglia con i differenti algoritmi è possibile notare che i risultati restituiti differiscono.

Varianza, deviazione standard e coefficiente di variazione

Siccome gli indici di posizione non tengono conto della variabilità dei dati, abbiamo bisogno di introdurre due nuovi indici: *varianza campionaria* e *deviazione standard*. Questi due indici ci permettono di permettono di misurare la variabilità di una distribuzione di frequenza. Diamo le definizioni di queste due misure:

Assegnato un insieme di dati numerici $x_1, x_2, ..., x_n$, si definisce **varianza campionaria**, indicata con s^2, la quantità:

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

con n=2,3,...

dove $\bar{x}$ è la media campionaria.

Definiamo come **devianzione standard**, indicata con s, la radice quadrata della varianza campionaria

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

con n=2,3,...

Questi due indici sono detti **indici di dispersione** in quanto più i dati risultanto discostarsi dalla media più essi risultano essere grandi.

In R è possibile calcolare la varianza campionaria di un vettore numerico utilizzando la funzione *var()*:

```
var(lavoro)
```

```
## [1] 52.48471
```

```
var(famiglia)
```

```
## [1] 97.43313
```

```
var(studio)
```

```
## [1] 1.701553
```

```
var(asilo)
```

```
## [1] 216.5964
```

```r
var(altro)
```

```
## [1] 3.773721
```

È possibile ricavare la devizione standard campionaria utilizzando la funzione *sd()*:

```r
sd(lavoro)
```

```
## [1] 7.244633
```

```r
sd(famiglia)
```

```
## [1] 9.870822
```

```r
sd(studio)
```

```
## [1] 1.304436
```

```r
sd(asilo)
```

```
## [1] 14.71722
```

```r
sd(altro)
```

```
## [1] 1.942607
```

La media campionaria e la varianza campionaria sono, rispettivamente, gli indici di posizione e di dispersione più utilizzati. Per confrontare le variazioni esistenti tra i diversi campioni di dati è possibile utilizzare il **coefficiente di variazione**.

Assegnato un insieme di dati numerici $x_1, x_2, ..., x_n$ si definisce *coefficiente di variazione* il rapporto tra la deviazione standard e il modulo della media campionaria:

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

In R non è presente una funzione per calcolare il coefficiente di variazione, quindi la definiamo noi:

```r
cv <- function(x){
  sd(x)/abs(mean(x))
}
```

Applicandola ai vettori a nostra disposizione:

```r
cv(lavoro)
```

```
## [1] 0.2519434
```

```r
cv(famiglia)
```

```
## [1] 0.2461245
```

```r
cv(studio)
```

```
## [1] 0.5207328
```

```r
cv(asilo)
```

```
## [1] 0.5858764
```

```r
cv(altro)
```

```
## [1] 0.5510159
```

Tra i coefficienti di variazione calcolati, gli ultimi tre risultano essere quelli con valore maggiore, fatto che indica che la dispersione dei dati è maggiore in questi vettori.

Forma di una distribuzione di frequenza

Con gli indici descritti finora è possibile comprendere la forma delle distribuzioni di frequenza ed eventuali sbilanciamenti verso destra o sinistra. Esistono indici statistici che permettono di misurare quando una distribuzione di frequenza presenta simmetria o asimmetria oppure se è più o meno piccata. Un indice che permette di misurare la simmetria di una distribuzione di frequenza è la **skewness campionaria**.

Assegnato un insieme di dati numerici $x_1, x_2, ..., x_n$ si definisce *skewness campionaria* il valore:

$$\gamma_1 = \frac{m_3}{m_2^{3/2}}$$

dove m_3 denota il momento centrato campionario di ordine 3. In generale, il momento centrato campionario di ordine j è definito come segue:

$$m_j = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^j \qquad j = 1, 2, ...$$

Se $\gamma_1 = 0$ allora la distribuzione di frequenza è simmetrica; Se $\gamma_1 > 0$ allora l'asimmetria è positiva (la coda di destra è più allungata); Se $\gamma_1 < 0$ l'asimmetria è negativa (la coda di sinistra è più allungata).

Non è presente una funzione per il calcolo della skewness campionaria in R, quindi abbiamo provveduto ad implementarlo:

```
skw <- function(x){
  n <- length(x)
  m2 <- (n-1)*var(x)/n
  m3 <- (sum((x-mean(x))^3))/n
  m3/(m2^1.5)
}
```

Riferendoci ai dati dei vettori in analisi,

```
skw(lavoro)
```

```
## [1] -0.3185947
```

Qui abbiamo un'asimmetria negativa in quanto la skewness è minore di 0.

```
skw(famiglia)
```

```
## [1] -0.5760205
```

Anche qui abbiamo un'asimmetria negativa in quanto la skewness è minore di 0.

```
skw(studio)
```

```
## [1] 0.5093442
```

```
skw(asilo)
```

```
## [1] 1.374309
```

```r
skw(altro)
```

```
## [1] 1.754433
```

Con questi ultimi tre vettori, l'asimmetria risulta essere positiva in quanto la skewness è maggiore di 0.

Un indice per misurare la densità dei dati intorno alla media è la **curtosi campionaria**

Assegnato un insieme di dati numerici $x_{1,x_2,\ldots,x_n}$ si definisce *curtosi campionaria* il valore:

$$\gamma_2 = \beta_2 - 3$$

dove

$$\beta_2 = \frac{m_4}{m_2^2}$$

dove m_4 rappresenta il momento campionario di ordine 4.

Questi due indici β_2 e γ_2 permettono di confrontare la distribuzione di frequenza dei nostri dati con una densità di probabilità normale.

Se $\beta_2 < 3$ e $\gamma_2 < 0$ la distribuzione di frequenza si dice *platicurtica*, ovvero risulta essere più piatta di una normale. Se $\beta_2 > 3$ e $\gamma_2 > 0$ la distribuzione di frequenza si dice *leptocurtica*, ovvero risulta essere più piccata di una normale. Se $\beta_2 = 3$ e $\gamma_2 = 0$ la distribuzione di frequenza si dice *normocurtica*, ovvero piatta come una normale.

Implementiamo in R il calcolo della curtosi campionaria come segue

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

Riferendoci ai vettori in analisi

```r
curt(lavoro)
```

```
## [1] 0.2741569
```

```r
curt(famiglia)
```

```
## [1] -0.8179971
```

```r
curt(studio)
```

```
## [1] -0.524069
```

```r
curt(asilo)
```

```
## [1] 1.33479
```

```r
curt(altro)
```

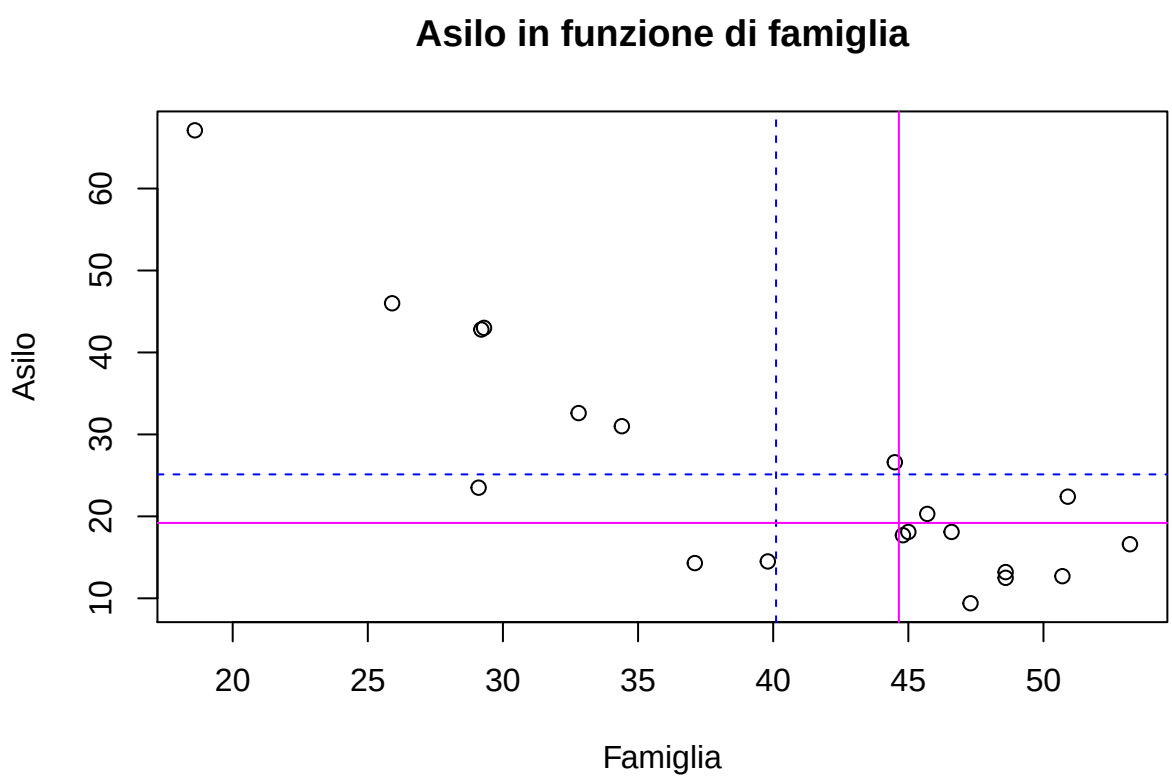
```
## [1] 3.380733
```

Nei vettori *lavoro*, *asilo* e *altro* la curtosi è positiva indicando quindi che la distribuzione di frequenza è più piccata di una normale e si dice in questo caso leptocurtica. Per quanto riguarda i vettori *famiglia* e *studio* invece, la curtosi è negativa, ovvero la distribuzione è più piatta di una normale, quindi platicurtica.

Statistica descrittiva bivariata

La statistica descrittiva bivariata è la branca della statistica che si occupa di definire i metodi grafici e statistici per descrivere le relazioni fra due variabili. Le relazioni fra variabili quantitative possono essere rappresentate graficamente mediante scatterplot in cui, ogni coppia di osservazioni viene rappresentata con un punto o un cerchio sul piano euclideo. Quello che si fa è scegliere una variabile da porre sull'asse delle ascisse, che prende il nome di *variabili indipendente*, e una da porre sull'asse delle ordinate chiamata *variabile dipendente* e si disegnano i punti in corrispondenza delle coppie. Creando questo grafico vogliamo capire se le coppie di punti presentano una certa regolarità e se esiste una relazione fra le variabili.

```r
plot(famiglia, asilo, main="Asilo in funzione di famiglia",
     xlab="Famiglia", ylab = "Asilo")
abline(v=median(famiglia), lty=1, col="magenta")
abline(v=mean(famiglia), lty=2, col="blue")
abline(h=median(asilo), lty=1, col="magenta")
abline(h=mean(asilo), lty=2, col="blue")
```



Dallo scatterplot prodotto possiamo notare che i dati sembrano essere sparsi intorno ad una retta discendente, mostrando così una correlazione lineare negativa fra le variabili.

Covarianza e correlazione campionaria

È possibile misurare la correlazione tra le variabili utilizzando la **covarianza campionaria**.

Assegnato un campione bivariato (x_1, y_1), (x_2, y_2), ..., (x_n, y_n) di una variabile quantitativa bidimensionale (X,Y), siano $\bar{x}$ e $\bar{y}$ le medie campionarie di $x_1, x_2, ..., x_n$ e di $y_1, y_2, ..., y_n$. La covarianza campionaria tra le due variabili X e T è definita come segue:

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

Quando $C_{xy} > 0$ si dice che le variabili sono *correlate positivamente*. Quando $C_{xy} < 0$ si dice che le variabili sono *correlate negativamente*. Quando $C_{xy} = 0$ si dice che le variabili sono *non correlate*.

Inoltre dividiamo per $n - 1$ perchè in caso x e y siano uguali la covarianza campionaria è uguale alla varianza campionaria.

È possibile calcolare la covarianza campionaria in R utilizzando la funzione *cov()*

```
cov(famiglia, asilo)
```

```
## [1] -123.6491
```

Com'è possibile vedere, la covarianza campionaria fra le due variabili considerate è negativa, fatto che ci dice che le due variabili sono correlate negativamente.

È possibile ottenere una misura quantitativa della correlazione fra le variabili introducendo il **coefficiente di correlazione campionaria**

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

dove s_x e s_y corrisponde rispettivamente alla deviazione standard di x e di y.

Questo coefficiente di correlazione ha lo stesso segno della covarianza e, in base al suo valore, diciamo che le variabili sono correlate positivamente, negativamente o non correlate.

Il coefficiente di correlazione gode di diverse proprietà:

- $-1 \leq r_{xy} \leq 1$;
- Se $r_{xy} = 1$ allora abbiamo una correlazione perfetta positiva e tutti i punti sono allineati su una linea retta ascendente;
- Se $0 < r_{xy} < 1$ allora i punti sono posizionati in una nuvola attorno ad una linea retta interpolante ascendente;
- Se $r_{xy} = 0$ allora i punti sono dispersi in una nuvola che non presenta alcuna evidente direzione di natura lineare;
- Se $-1 < r_{xy} < 0$ allora i punti sono posizionati in una nuvola attorno ad una linea retta interpolante discendente;
- Se $r_{xy} = -1$ abbiamo una correlazione perfetta negativa e tutti i punti sono allineati su una retta discendente.

In R per calcolare il coefficiente di correlazione si utilizza la funzione *cor()*

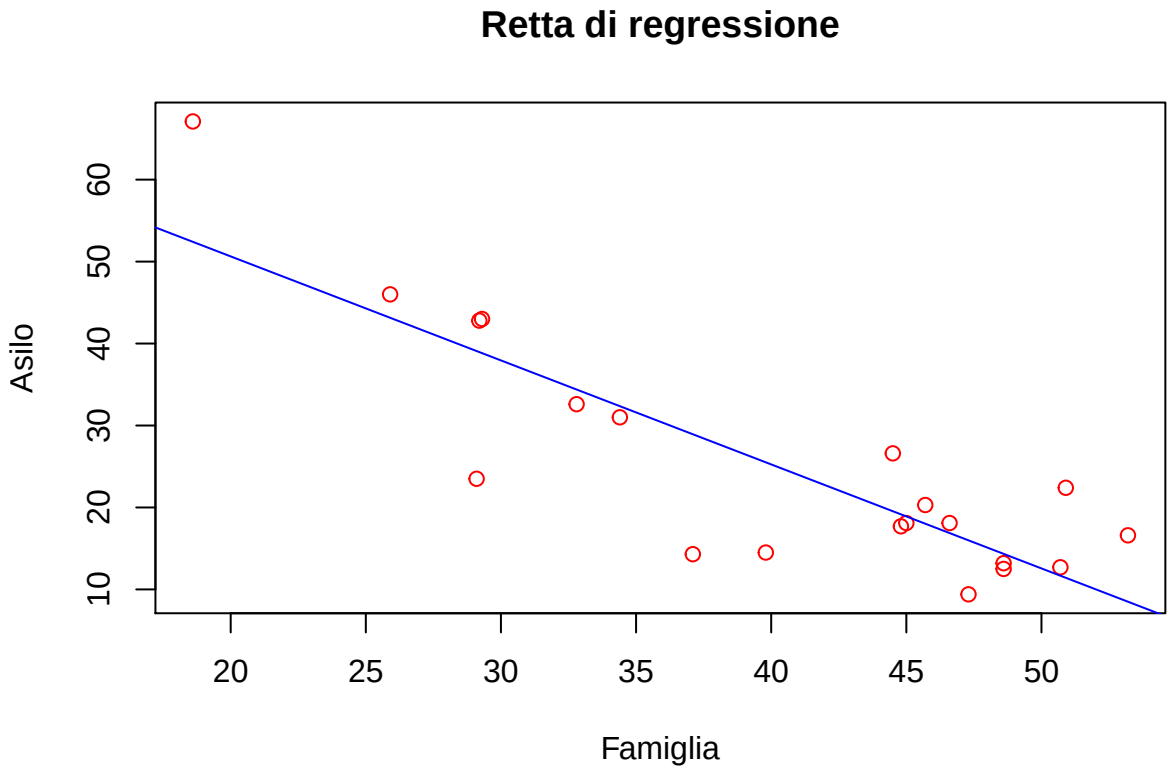
```
cor(famiglia, asilo)
```

```
## [1] -0.8511612
```

Misurando la correlazione fra le due variabili che stiamo considerando otteniamo un valore negativo piuttosto vicino a -1, fatto che indica la correlazione negativa fra le due variabili.

È possibile aggiungere allo scatterplot precedente la retta interpolante stimata.

```r
plot(famiglia, asilo, main = "Retta di regressione",
     xlab="Famiglia", ylab="Asilo", col="red")
abline(lm(asilo~famiglia), col="blue")
```



La funzione *abline(lm(asilo~famiglia)* è quella che permette di aggiungere la retta allo scatterplot, dove *lm()* sta per linear model ed è utilizzata per eseguire le analisi di regressioni lineari.

Il modello lineare viene utilizzato per spiegare o prevedere un andamento futuro sulla base della relazione si instaura tra una variabile Y dipendente e una o più variabili indipendenti $X_1, X_2, ..., X_p$. Nel caso in cui $p = 1$, l'analisi prende il nome di **regressione semplice** mentre se $p > 1$ allora prende il nome di **regressione multipla**.

Regressione lineare semplice

Questo modello di regressione è esprimibile attraverso l'equazione di una retta che interpola al meglio la nuvola di punti dello scatterplot. L'equazione è la seguente:

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

dove:

- α è l'intercetta;
- β il coefficiente angolare.

Il coefficiente angolare indica la pendenza della retta mentre l'intercetta corrisponde all'ordinata del punto di intersezione della retta interpolante con l'asse delle ordinate. L'identificazione di questa retta viene ottenuta applicando il *metodo dei minimi quadrati*. I **coefficienti di regressione** sono i valori α e β per i quali la somma Q dei quadrati degli errori è **minima**:

$$Q = \sum_{i=1}^{n} [y_i - (\alpha + \beta x_i)]^2$$

dove:

- n è il numero di osservazioni;
- $(x_1, x_2, ..., x_n)$ sono i valori osservati della variabile X;
- $(y_1, y_2, ..., y_n)$ sono i valori osservati della variabile Y.

Il metodo dei minimi quadrati conduce alla fine a:

$$\beta = \frac{s_y}{s_x} * r_{xy} \qquad \alpha = \bar{y} - \beta \bar{x}$$

Riferendoci ai vettori da noi presi in considerazione abbiamo che:

```
beta <- (sd(asilo)/sd(famiglia)) * cor(famiglia,asilo)
alpha <- mean(asilo) - beta*mean(famiglia)

c(alpha,beta)
```

```
## [1] 76.015883 -1.269066
```

Il fatto che β risulta essere negativo, ci conferma che la retta è discendente. Per poter eseguire le analisi di regressione lineare utilizziamo la funzione *lm(y~x)* (l'argomento indica che y dipende da x).

```
linearmodel <- lm(asilo~famiglia)
linearmodel
```

```
##
## Call:
## lm(formula = asilo ~ famiglia)
##
## Coefficients:
## (Intercept)     famiglia
##      76.016       -1.269
```

Vediamo che i coefficienti coincidono con quelli calcolati precedentemente. La retta di regressione con i vettori da noi presi in esame ha come equazione:

$$y = 76.016 - 1.269 * x$$

Residui

Calcolati i coefficienti di regressione e con essi l'equazione della retta di regressione, è possibile osservare quanto questa retta si discosta dai valori osservati. Esistono quindi degli scostamenti, chiamati **residui**, tra le ordinate dei punti y_i che sono i valori osservati e i corrispondenti valori stimati ottenuti mediante la retta di regressione.

$$\hat{y} = \alpha + \beta x_i \qquad (i = 1, 2, ..., n)$$

I residui verranno quindi calcolati nel seguente modo:

$$E_i = y_i - \hat{y}_i \qquad (i = 1, 2, ..., n)$$

È possibile calcolare i valori stimati utilizzando la funzione *fitted()* dandole in input *lm(y~x)*

```
fitted(lm(asilo~famiglia))
```

```
##         1         2         3         4         5         6         7
## 16.877418  8.501584 19.161736 15.989072 11.420435 14.339286 19.542456
##         8         9        10        11        12        13        14
## 11.674248 25.507065 14.339286 18.907923 28.933543 18.019577 52.411259
##        15        16        17        18        19        20
## 39.086069 38.832256 43.147079 38.959162 34.390525 32.360020
```

Per calcolare i residui invece esiste la funzione *resid()* che prende in input sempre *lm(y~x)*

```
resid(lm(asilo~famiglia))
```

```
##          1          2          3          4          5          6
##   1.222582   8.098416  -1.461736  -6.589072  10.979565  -1.139286
##          7          8          9         10         11         12
##   7.057544   1.025752 -11.007065  -1.839286  -0.807923 -14.633543
##         13         14         15         16         17         18
##   2.280423  14.688741 -15.586069   4.167744   2.852921   3.840838
##         19         20
##  -1.790525  -1.360020
```

I residui sono anche uno degli attributi fornitoci dalla funzione *lm()* ed è possibile accedervi nella seguente maniera:

```
linearmodel$residuals
```

```
##          1          2          3          4          5          6
##   1.222582   8.098416  -1.461736  -6.589072  10.979565  -1.139286
##          7          8          9         10         11         12
##   7.057544   1.025752 -11.007065  -1.839286  -0.807923 -14.633543
##         13         14         15         16         17         18
##   2.280423  14.688741 -15.586069   4.167744   2.852921   3.840838
##         19         20
##  -1.790525  -1.360020
```

È inutile calcolare la media campionaria dei residui $\bar{E}$ in quanto essa è sempre nulla perchè gli scostamenti positivi e negativi si compensano a vicenda. Per quanto riguarda invece mediana, varianza campionaria e deviazione standard campionaria:

```
median(linearmodel$residuals)
```

```
## [1] 0.1089144
```

```
var(linearmodel$residuals)
```

```
## [1] 59.67764
```

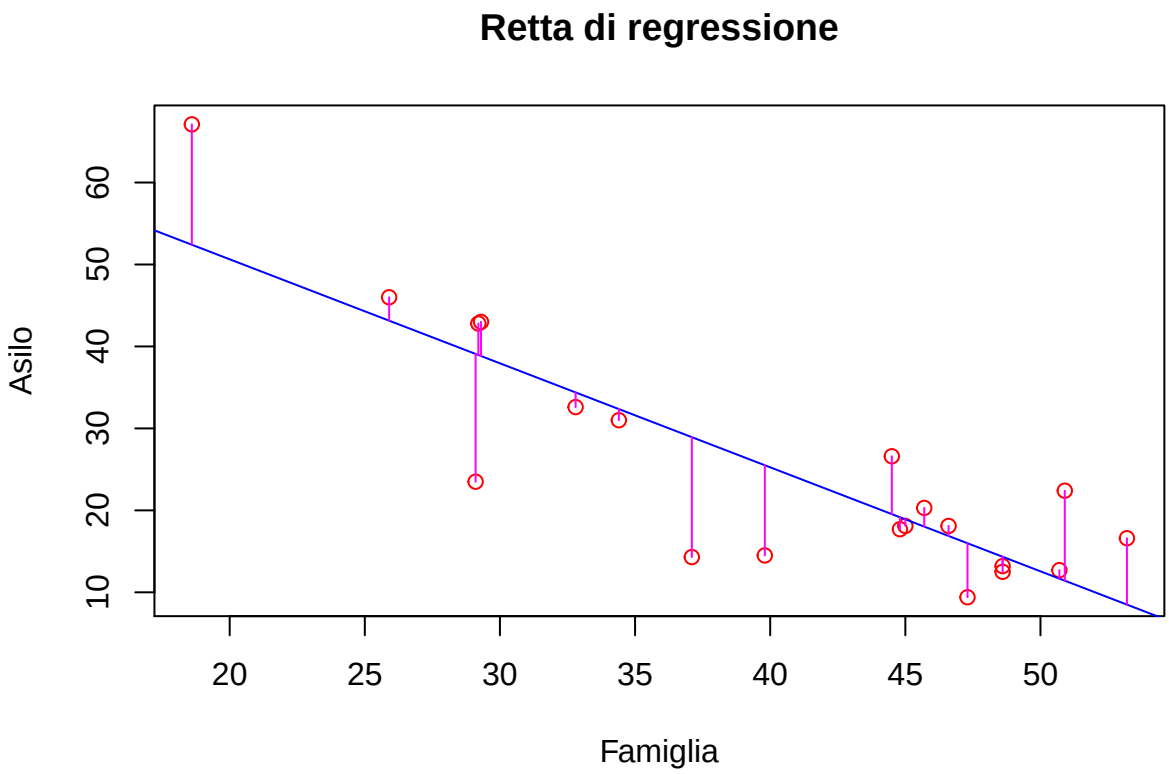
```
sd(linearmodel$residuals)
```

```
## [1] 7.72513
```

Il coefficiente di variazione è impossibile da ottenere in quanto abbiamo detto che la media campionaria dei residui risulta essere nulla. È possibile visualizzare graficamente i residui attraverso diversi grafici:

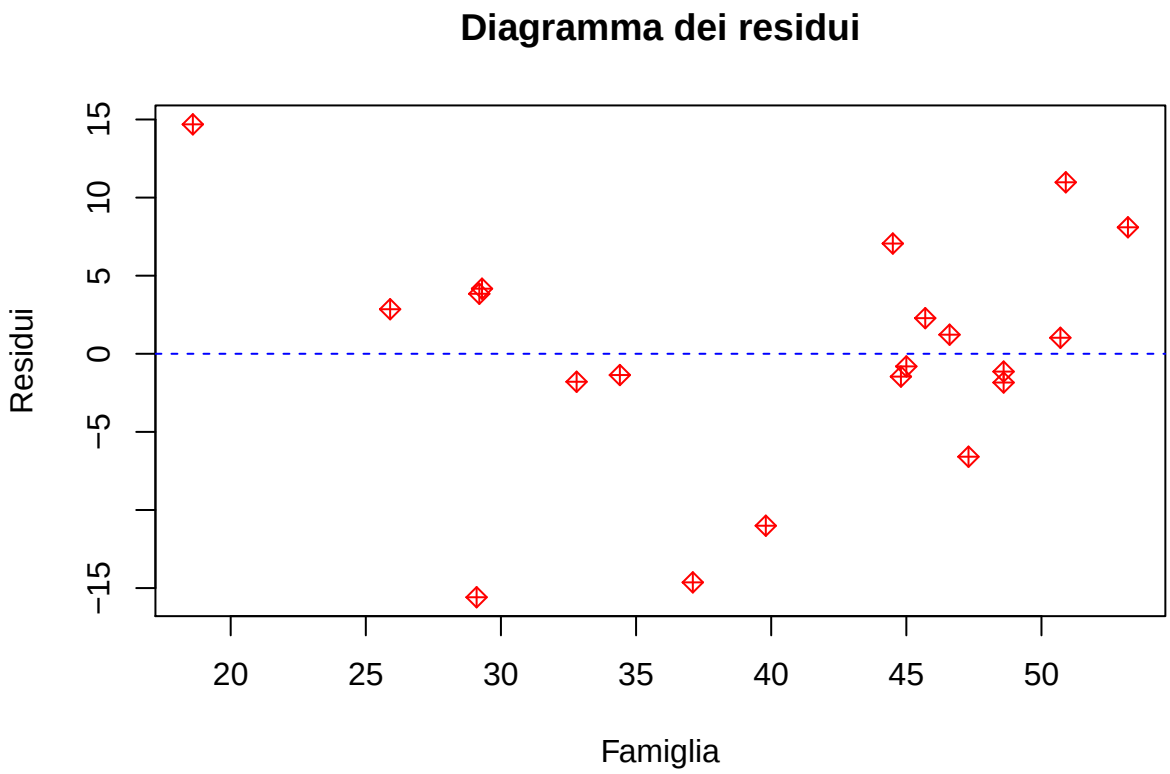
- Tracciando dei segmenti che congiungono i valori stimati e i valori osservati utilizzando la funzione *segments()*

```
plot(famiglia, asilo, main = "Retta di regressione",
     xlab="Famiglia", ylab="Asilo", col="red")
abline(lm(asilo~famiglia), col="blue")
stime <- fitted(lm(asilo~famiglia))
segments(famiglia, stime, famiglia, asilo, col="magenta")
```



- Rappresentando i valori dei residui rispetto alle osservazioni della variabile indipendente Il **diagramma dei residui** è un grafico in cui i valori dei residui sono posti sull'asse delle ordinate e quelli della variabile indipendente sull'asse delle ascisse.

```
residui <- resid(lm(asilo~famiglia))
plot(famiglia, residui, main = "Diagramma dei residui",
     xlab="Famiglia", ylab="Residui", pch=9, col="red")
abline(h=0, col="blue", lty=2)
```



Questo diagramma ci aiuta a capire come si adatta la retta di regressione rispetto ai dati.

Calcola i valori dei residui standardizzati rispetto ai valori stimati È possibile calcolare i *residui standardizzati* nella seguente maniera:

$$E_i^{(s)} = \frac{E_i - E}{s_E}$$

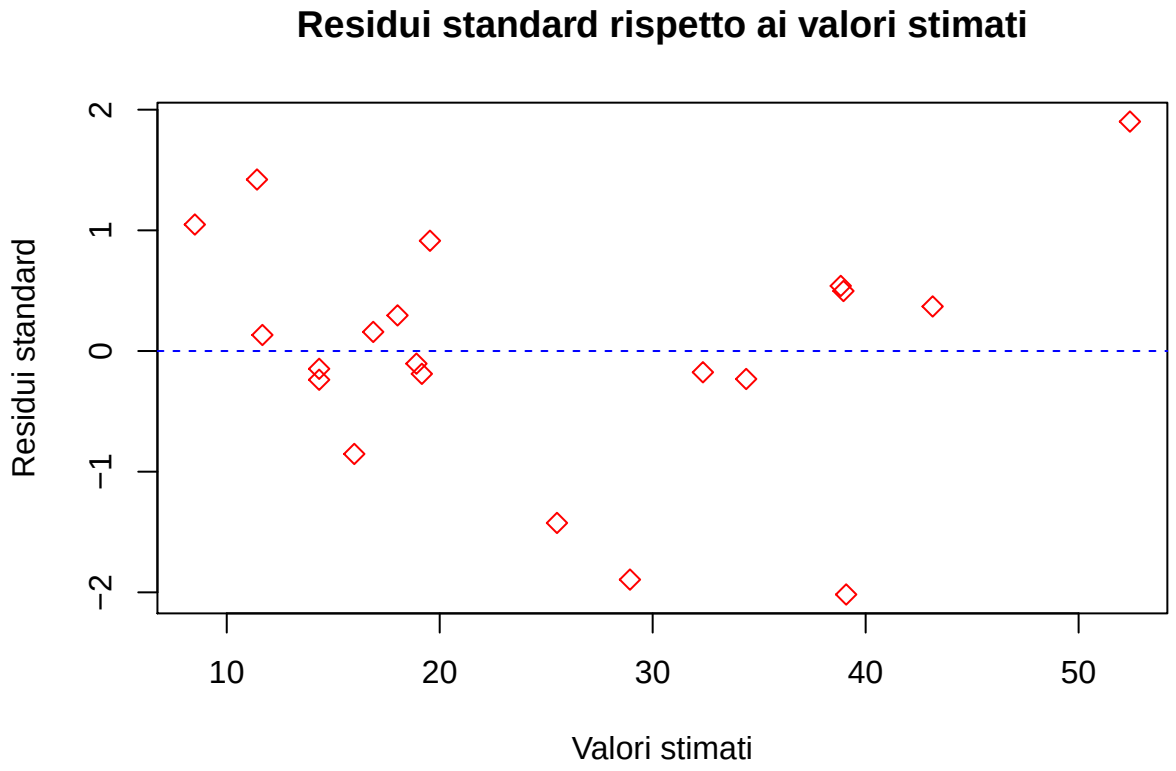
che hanno media campionaria nulla e varianza unitaria.

```
residuistandard <- residui/sd(residui)
residuistandard
```

```
##          1          2          3          4          5          6
##  0.1582604  1.0483210 -0.1892183 -0.8529399  1.4212789 -0.1474779
##          7          8          9         10         11         12
##  0.9135825  0.1327812 -1.4248387 -0.2380913 -0.1045837 -1.8942777
##         13         14         15         16         17         18
##  0.2951954  1.9014230 -2.0175800  0.5395047  0.3693039  0.4971874
##         19         20
## -0.2317793 -0.1760514
```

È possibile rappresentare questi residui standardizzati mediante un grafico:

```
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)
```



Coefficiente di determinazione

Quello che ci interessa capire è quanto la retta si adatti ai dati e questo lo possiamo fare attraverso il **coefficiente di determinazione** definito nella seguente maniera:

$$D^2 = \frac{\sum_{i=1}^{n}(\hat{y}_i - \bar{y})^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

Il coefficiente di determinazione è il rapporto tra *la varianza dei valori stimati tramite la retta di regressione* e *la varianza dei valori osservati.*

Nel caso di regressione lineare semplice esso è pari al quadrato del coefficiente di correlazione, ovvero:

$$D^2 = r_{xy}^2$$

Se il coefficiente di determinazione è vicino ad 1 allora fra le due variabili esiste una forte correlazione, se invece questo coefficiente risulta vicino allo 0 non esiste correlazione fra le due variabili. In questo caso, per calcolare il coefficiente di determinazione, possiamo utilizzare il coefficiente di correlazione ed elevarlo al quadrato:

```
(cor(famiglia, asilo))^2
```

```
## [1] 0.7244754
```

In alternativa possiamo utilizzare la funzione *summary(lm(y~x))$r.square*:

```
summary(lm(asilo~famiglia))$r.square
```

```
## [1] 0.7244754
```

Com'é possibile vedere, il coefficiente di determinazione fra le due variabili prese in esame risulta essere abbastanza vicino ad 1, indicando una correlazione fra le variabili.

Regressione lineare multipla

Molte applicazioni affrontano situazioni nelle quali vi è più di una variabile indipendente. Il modello di regressione lineare multipla serve a spiegare la relazione fra una variabile quantitativa dipendente Y e diverse variabili quantitative indipendenti $X_1, X_2, ..., X_p$. Come variabile dipendente scegliamo "Lavoro" e ne valutiamo la relazione con tutte le altre variabili a nostra disposizione. Iniziamo con l'applicare le funzioni *cov()* e *cor()* sul dataframe "motivi" definito in precedenza, nella parte riguardante gli scatterplot, per calcolare la covarianza e le correlazioni fra le coppie di variabili. Gli output delle funzioni saranno due matrici simmetriche.

```
cov(motivi)
```

```
##              lavoro    famiglia     studio       asilo     altro
## lavoro    52.484711   27.536026   3.174447  -84.309053  1.294682
## famiglia  27.536026   97.433132   7.321026 -123.649053 -8.615503
## studio     3.174447    7.321026   1.701553  -12.275895  0.076550
## asilo    -84.309053 -123.649053 -12.275895  216.596421  3.453884
## altro      1.294682   -8.615503   0.076550    3.453884  3.773721
```

```r
cor(motivi)
```

```
##               lavoro   famiglia      studio      asilo       altro
## lavoro    1.00000000  0.3850628  0.33591472 -0.7907372  0.09199446
## famiglia  0.38506277  1.0000000  0.56858571 -0.8511612 -0.44930619
## studio    0.33591472  0.5685857  1.00000000 -0.6394474  0.03020909
## asilo    -0.79073720 -0.8511612 -0.63944740  1.0000000  0.12080843
## altro     0.09199446 -0.4493062  0.03020909  0.1208084  1.00000000
```

Dalle matrici restituiteci come risultato possiamo notare che vi è una correlazione negativa molto vicina ad 1 fra la variabile "Lavoro" e "Asilo" e, come visto prima, anche fra "Famiglia" e "Asilo" e "Studio" e "Asilo".

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

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

dove:

- α rappresenta l'intercetta, ovvero il valore di Y quando $X_1 = X_2 = ... = X_p = 0$
- $\beta_1, \beta_2, ..., \beta_p$ sono i regressori. Abbiamo che β_1 rappresenta l'inclinazione di Y rispetto alla variabile X_1 tenendo costanti le variabili $X_2, X_3, ..., X_p$, β_2 rappresenta l'inclinazione... e così fino a β_p.

Utilizziamo la funzione $lm(y \sim x_1 + x_2 + ... + x_p)$ per eseguire l'analisi di regressioni lineari multivariate:

```r
multiplelinearmodel <- lm(lavoro ~ famiglia + studio + asilo + altro)
multiplelinearmodel$coefficients
```

```
## (Intercept)    famiglia      studio       asilo       altro
##  100.462590   -1.006580   -1.006555   -1.004737   -1.014970
```

I segni di tutti i regressori sono negativi, questo vuol dire che i valori di "Famiglia", "Studio", "Asilo"" e "Altro" hanno un effetto negativo sul valore di "Lavoro", in altre parole al crescere dei valori "Famiglia", "Studio", "Asilo"" e "Altro" diminuisce quello di "Lavoro".

Residui

Calcolati i coefficienti passiamo al calcolo dei residui. Prima di calcolare i residui bisogna calcolare i valori stimati che risultano essere uguali a:

$$\hat{y}_i = \alpha + \beta_1 x_{i,1} + \beta_2 x_{i,2} + ... + \beta_p x_{i,p} \qquad (i = 1, 2, ..., n)$$

Ricordiamo che la media campionaria dei residui $\bar{E}$ è sempre nulla. I residui li calcoliamo esattamente come prima:

$$E_i = y_i - \hat{y}_i$$

Utilizziamo anche qui la funzione *fitted()* per calcolare i valori stimati tramite regressione lineare multipla:

```
stimemult <- fitted(lm(lavoro ~ famiglia + studio + asilo + altro))
stimemult
```

```
##        1        2        3        4        5        6        7        8
## 28.34642 26.08850 30.84782 38.69300 22.17606 35.20738 23.50679 31.37730
##        9       10       11       12       13       14       15       16
## 39.33268 30.51090 31.36360 34.08336 27.99525 10.82657 41.65093 21.68974
##       17       18       19       20
## 21.52531 23.66772 28.02987 28.18080
```

```
residuimult <- resid(lm(lavoro ~ famiglia + studio + asilo + altro))
residuimult
```

```
##            1            2            3            4            5
## -0.046415472  0.011495272  0.052183935  0.106999953 -0.076064888
##            6            7            8            9           10
## -0.007381926 -0.006789891  0.022696499 -0.032684485 -0.010901486
##           11           12           13           14           15
##  0.036403466  0.016637996  0.004750038  0.073425725 -0.050925030
##           16           17           18           19           20
## -0.089736047  0.074694203 -0.067724913 -0.029867500  0.019204553
```

Per quanto riguarda mediana, varianza campionaria e deviazione standard dei residui abbiamo che:

```
median(multiplelinearmodel$residuals)
```

```
## [1] -0.001019927
```

```
var(multiplelinearmodel$residuals)
```

```
## [1] 0.002796633
```

```
sd(multiplelinearmodel$residuals)
```

```
## [1] 0.0528832
```

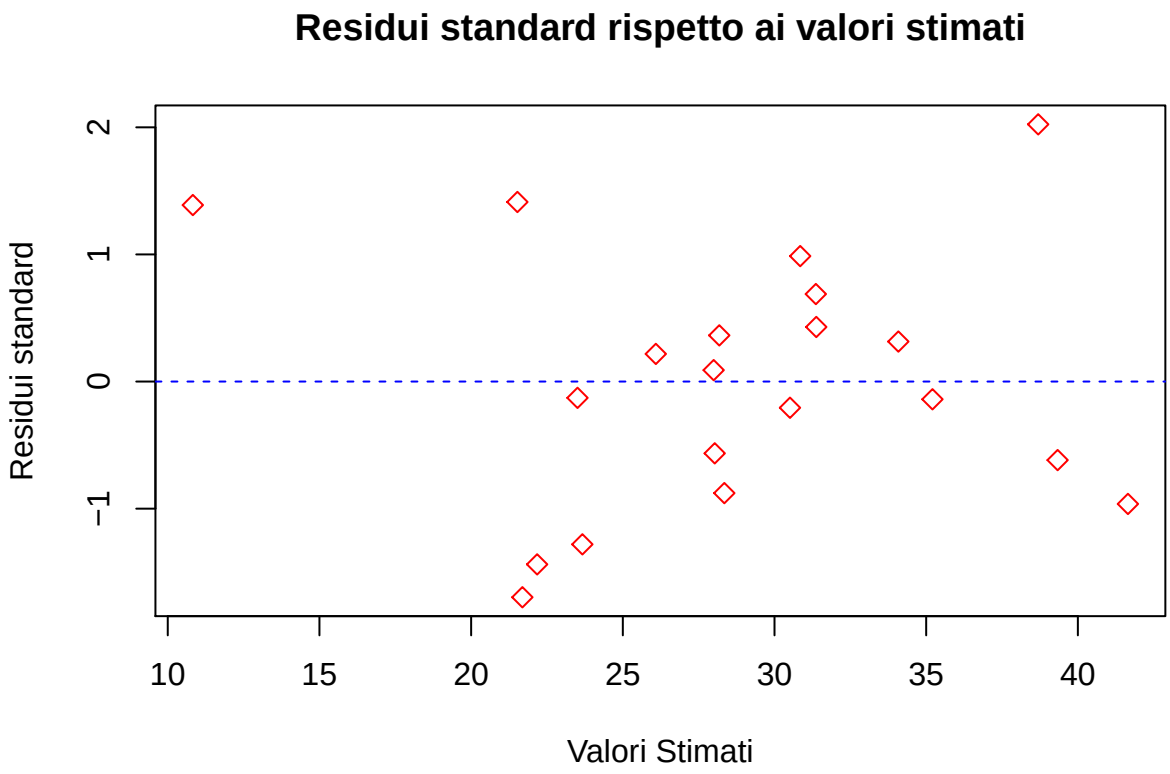
Anche nel caso multivariato è interessante calcolare i residui standardizzati e lo facciamo attraverso le seguenti linee di codice:

```
residuimultstandard <- residuimult/sd(residuimult)
residuimultstandard
```

```
##           1           2           3           4           5           6
## -0.87769787  0.21737095  0.98677718  2.02332599 -1.43835638 -0.13958925
##           7           8           9          10          11          12
## -0.12839411  0.42918164 -0.61805043 -0.20614271  0.68837487  0.31461780
##          13          14          15          16          17          18
##  0.08982131  1.38845087 -0.96297179 -1.69687249  1.41243726 -1.28065081
##          19          20
## -0.56478238  0.36315035
```

Realizziamo quindi un grafico in cui i residui standardizzati vengono disegnati in funzione dei valori stimati

```
plot(stimemult,residuimultstandard,
     main="Residui standard rispetto ai valori stimati",
     ylab="Residui standard", xlab="Valori Stimati",
     pch=5, col="red")
abline(h=0, col="blue", lty=2)
```



I punti risultano essere disposti anche in questo caso in maniera del tutto casuale e non è possibile notare nessuna tendenza.

Coefficiente di determinazione

Come in precedenza, il coefficiente di determinazione è uguale a:

$$D^2 = \frac{\sum_{i=1}^{n}(\hat{y}_i - \bar{y})\check{s}}{\sum_{i=1}^{n}(y_i - \bar{y})\check{s}}$$

Ed esattamente come prima, se $D^2 = 0$ allora il modello di regressione multipla usato non spiega per nulla i dati, se $D^2 = 1$ allora il modello approssima perfettamente i dati.

Utilizziamo la funzione *summary(lm(y~x_1+x_2+...+x_p))$r.square* per calcolarlo. Utilizzando i dati a nostra disposizione otteniamo:

```
summary(lm(lavoro ~ famiglia+studio+asilo+altro))$r.square
```

```
## [1] 0.9999467
```

Essendo il valore del coefficiente decisamente vicino ad 1, il modello di regressione multipla utilizzato spiega in maniera ottima i dati.

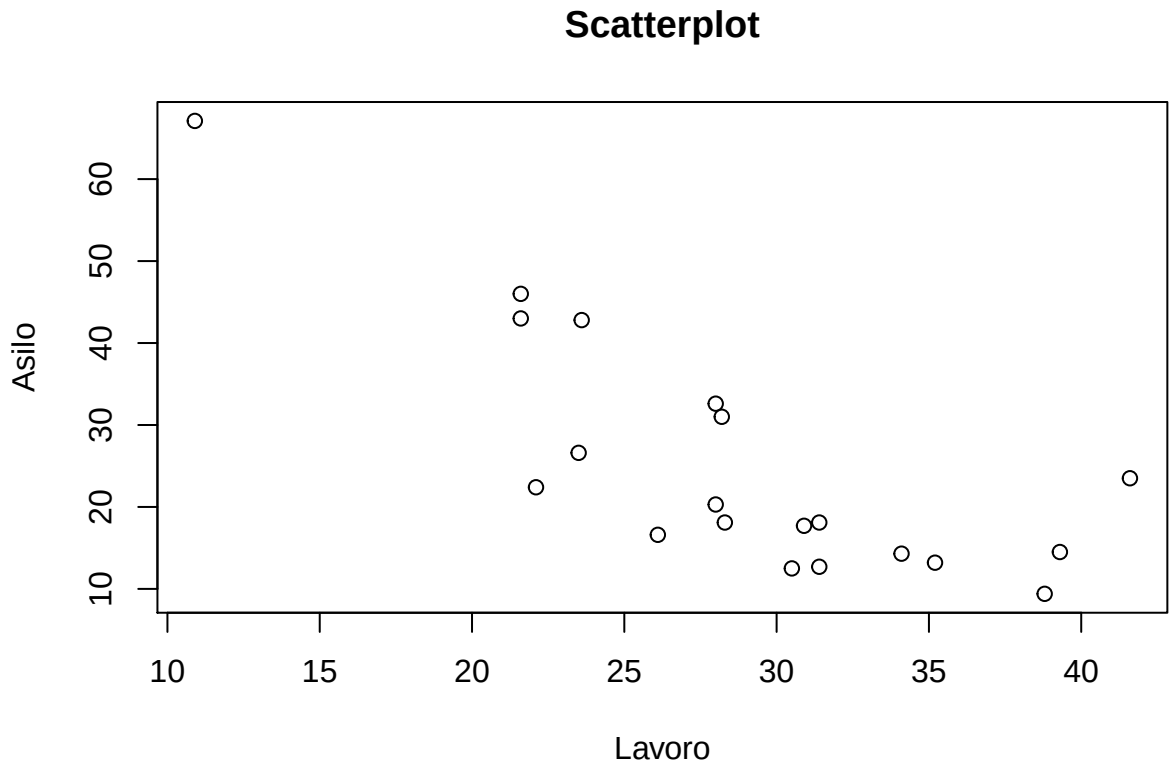
Regressione non lineare

Molto spesso la linearità non è un modello accettabile e lo notiamo quando il coefficiente di determinazione è prossimo allo 0. Il modello lineare più semplice è il **modello di regressione polinomiale** che calcoliamo nella seguente maniera:

$$Y = \alpha + \beta X + \gamma X^2$$

Degli scatterplot creati in precedenza, in gran parte sembra che la linearità non basti. Esaminiamo nel dettagli la coppia "Lavoro" e "Asilo"

```
plot(lavoro, asilo, main="Scatterplot", xlab="Lavoro", ylab="Asilo")
```



Calcolando il coefficiente di determinazione esso risulta pari a:

```
summary(lm(asilo ~ lavoro))$r.square
```

```
## [1] 0.6252653
```

Un valore che si trova nel mezzo ma che, utilizzando un modello di regressione non lineare possiamo far aumentare e approssimare meglio.

Per la stima dei parametri $\alpha, \beta e \gamma$ è possibile ricorrere alla regressione multipla:

$$Y = \alpha + \beta X_1 + \gamma X_2$$

con regressori $X_1 = X$ e $X_2 = X^2$.

Utilizzando R possiamo stimare α, β e γ con la funzione *lm(y~x + I(x^2))* dove *I()* è un *identificatore di variabile* e viene inserito quando vi è la necessità di effettuare operazione matematiche nelle variabili della regressione, in questo caso, l'elevazione a potenza.

Procediamo all'approssimazione polinomiale

```
pol <- lm(asilo~lavoro+I(lavoro^2))
pol
```

```
##
## Call:
## lm(formula = asilo ~ lavoro + I(lavoro^2))
##
## Coefficients:
## (Intercept)       lavoro  I(lavoro^2)
##   124.38170     -5.72361      0.07451
```

```
alphapol <- pol$coefficients[[1]]
betapol <- pol$coefficients[[2]]
gammapol <- pol$coefficients[[3]]

c(alphapol, betapol, gammapol)
```

```
## [1] 124.38169775  -5.72360516   0.07450643
```

Il modello polinomiale costruito ha quindi come valori $\alpha = 124.3816$, $\beta = -5.7236$ e come $\gamma = 0.0745$. Procediamo col calcolo del coefficiente di determinazione attraverso la funzione *summary()*:

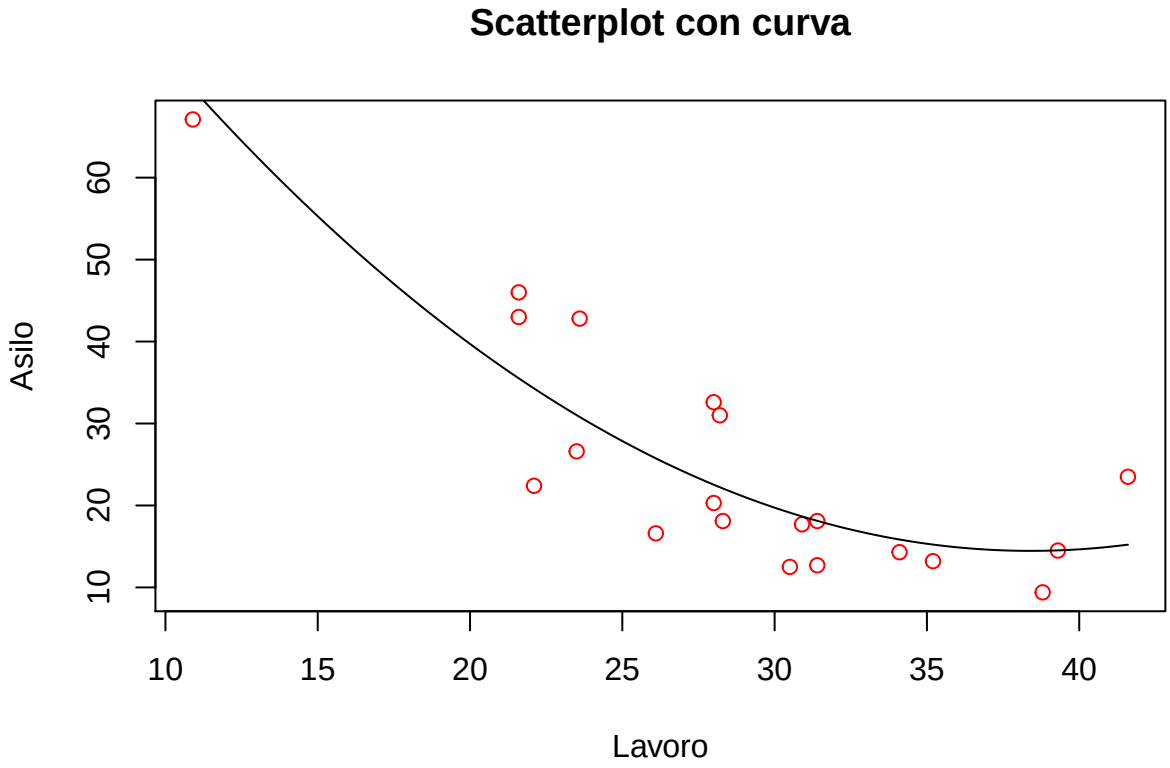
```
summary(lm(asilo~lavoro+I(lavoro^2)))$r.square
```

```
## [1] 0.7709843
```

Com'è possibile notare, il coefficiente di determinazione ottenuto col modello di regressione non lineare polinomia dal valore di 0.7709 è maggiore di quello di regressione lineare dal valore di 0.6252, risultando quindi nettamente migliore.

Andiamo adesso ad aggiungere la curva appena descritta allo scatterplot:

```r
plot(lavoro, asilo, main="Scatterplot con curva",
     xlab="Lavoro", ylab="Asilo", col = "red")
curve(alphapol+betapol*x+gammapol*x^2, add = TRUE)
```



Analisi dei cluster

L'analisi dei cluster è una metodologia che permette di raggruppare in sottoinsiemi entità appartenenti ad un insieme più ampio. Questi sottoinsiemi sono chiamati **cluster**. Le diverse tecniche per effettuare l'analisi dei cluster hanno come unico obiettivo quello di ottenere raggruppamenti in base alla somiglianza in modo che gli elementi di uno stesso gruppo siano tra loro più simili possibili e gli elementi appartenenti a gruppi diversi siano più diversi possibili. Sia n il numero di individui, il problema dell'analisi dei cluster sta nel determinare m sottoinsiemi, i cluster, di individui in I, con m minore intero di n, tali che I_i appartenga soltanto ad un unico sottoinsieme. Gli individui assegnati allo stesso cluster sono detti **simili**, mentre quelli assegnati a cluster diversi vengono chiamati **dissimili**. Molte delle tecniche di analisi dei cluster prevedono la standardizzazione di ogni variabile usanto dal media campionaria e la deviazione standard campionaria, questo per far si che tutti i valori osservati siano privi di unità di misura. Questa operazione è detta *scalamento* e, oltre

a rendere puri i valori osservati, ci permette di ridurne la grandezza. Preso un elemento x_{ij}, il corrispondente valore scalato è pari a:

$$\frac{x_{ij} - \bar{x}}{s_j}$$

Nel nostro caso, i valori risultano essere in percentuali, quindi non vi è bisogno di effettuare scalamento, di seguito mostreremo solo gli effetti dello scalamento sulla nostra tabella.

R fornisce la funzione *scale()* che, data una matrice in input, permette di effettuare uno scalamento. Questa funzione prevede due attributi, *center* che di default è uguale a TRUE e indica se dagli elementi di ogni colonna della matrice si sottrae il valore medio della corrispondente colonna e *scale*, che di default è posto anch'esso a TRUE e indica se bisogna dividere gli elementi centrati di ogni colonna della matrice per la deviazione standard della rispettiva colonna.

Mostriamo i la nostra tabella:

```
tabellaScalata <- scale(tabella)
tabellaScalata
```

```
##                 lavoro    famiglia      studio      asilo       altro
## Piemonte    -0.06280511  0.65799990  1.75938140 -0.4769924 -0.70292149
## Valle d'Aos -0.36647818  1.32663720 -0.46380207 -0.5789139 -0.68233060
## Liguria      0.29608124  0.47564427  0.22615142 -0.5041715  0.17219131
## Lombardia    1.38654362  0.72891597  0.45613592 -1.0681369 -1.00663711
## T-A Adige   -0.91861103  1.09362723  0.07282842 -0.1848176 -0.83676227
## Veneto       0.88962406  0.86061726 -0.77044806 -0.8099358 -1.02208028
## F-V Giulia  -0.72536453  0.44525166  0.22615142  0.1005625 -0.48671715
## Em-Romag     0.36509785  1.07336550  0.53279742 -0.8439097 -0.76469416
## Toscana      1.45556023 -0.03089915  0.53279742 -0.7216039 -0.15211520
## Umbria       0.24086796  0.86061726  1.98936590 -0.8574992 -0.12122886
## Marche       0.36509785  0.49590600  0.45613592 -0.4769924 -0.54848982
## Lazio        0.73778752 -0.30443259  1.68271990 -0.7351934  3.22479069
## Abruzzo     -0.10421508  0.56682208 -0.15715607 -0.3275076  0.08982776
## Molise      -2.46458302 -2.17864323 -1.38374006  2.8524418 -0.39920587
## Campania     1.77303662 -1.11490206 -1.15375556 -0.1100752  0.67152039
## Puglia      -0.98762764 -1.09464032 -0.69378657  1.2149038  0.45016832
## Basilicata  -0.98762764 -1.43908984 -1.07709406  1.4187467  0.99582690
## Calabria    -0.71156121 -1.10477119 -0.54046357  1.2013142 -0.50216032
## Sicilia     -0.10421508 -0.74005993 -1.38374006  0.5082483  1.20688351
## Sardegna    -0.07660843 -0.57796604 -0.31047907  0.3995321  0.41413427
## attr(,"scaled:center")
##   lavoro famiglia   studio    asilo    altro
##  28.7550  40.1050   2.5050  25.1200   3.5255
## attr(,"scaled:scale")
```

```
##    lavoro  famiglia    studio     asilo     altro
##  7.244633  9.870822  1.304436 14.717215  1.942607
```

Distanza e similarità

Le misure metriche di somiglianza sono basate soprattutto sulle *funzioni distanza* tra i vettori delle caratteristiche. La definizione di questa funzione è la seguente:

1. $d(X_i, X_j) = 0$ se e solo se $ X_i = X_j$, con X_i e X_j in E_p;
2. $d(X_i, X_j) \geq 0$ per ogni X_i e X_j in E_p;
3. $d(X_i, X_j) = d(X_j, X_i)$ per ogni X_i e X_j in E_p;
4. $d(X_i, X_j) \leq d(X_i, X_k) + d(X_k, X_j)$ per ogni X_i ,X_j e X_k in E_p.

Queste proprietà rappresentano le caratteristiche che deve possedere una funzione di distanza. Le distanze calcolate fra tutte le possibili coppie vengono inserite in una matrice simmetrica di cardinalità *nxn*, con ovviamente tutti 0 sulla diagonale, ed è quindi possibile considerare anche solo la matrice triangolare superiore. R ha a disposizione la funzione *dist(X, method=, diag=FALSE, upper=FALSE)* per calcolare la matrice delle distanze secondo un metodo impostato dall'utente attraverso l'attributo *method*.

I possibili valori di method, rappresentano i metodi disponibili per il calcolo della matrice delle distanze e, in particolare abbiamo:

1. metrica euclidea (**euclidean**)
2. metrica del valore assoluto o metrica di Manhattan (**manhattan**)
3. metrica del massimo o metrica di Chebycev (**maximum**)
4. metrica di Minkowski (**minkowski**)
5. distanza di Canberra (**canberra**)
6. distanza di Jaccard (**binary**)

Metrica euclidea

La metrica più diffusa e familiare è quella **euclidea**, definita come segue:

$$d_2(X_i, X_j) = \sum_{k=1}^{p} [(x_{ik} - x_{jk})^2]^{\frac{1}{2}}$$

La metrica euclidea usata su tutti i dati, è fortemente influenzata dall'unità di misura base alla quale è valutata ciascuna delle p caratteristiche. Per usare questa metrica in R basta settare *"euclidean"* come valore dell'attributo *method* della funzione *dist()*.

Per i dati a nostra disposizione abbiamo che:

```r
dist(tabella, method="euclidean", diag=FALSE, upper=FALSE)
```

```
##              Piemonte Valle d'Aos   Liguria Lombardia T-A Adige    Veneto
## Valle d'Aos  7.685155
## Liguria      4.129165    9.918447
## Lombardia   13.772004   15.804332 11.953414
## T-A Adige    8.962567    7.450503 11.858398 21.475775
## Veneto       9.321717   10.776159  7.762242  5.625913 16.213562
## F-V Giulia  10.191977   13.542688 11.648966 23.213791  7.814243 18.331165
## Em-Romag     7.625903    7.160000  7.970721  8.800051 13.453609  4.715930
## Toscana     13.561154   18.998971 10.312948  9.234479 21.990655 10.083953
## Umbria       6.447240    8.292050  6.874220  9.335331 13.492298  6.213091
## Marche       3.892300    9.954275  1.581139 11.684695 11.847092  7.403810
## Lazio       14.019875   19.848126 10.948283 14.841442 21.603521 14.593577
## Abruzzo      3.789670    8.715503  4.031823 15.594451  8.342062 10.768733
## Molise      59.202349   63.088767 59.434351 70.274764 56.312987 66.315942
## Campania    23.105603   29.604001 19.970000 23.514413 29.459207 23.202890
## Puglia      31.296128   35.963176 31.121722 41.941136 29.973655 38.127150
## Basilicata  35.726321   40.511327 35.355905 45.961855 34.599618 42.327490
## Calabria    30.726082   35.620535 30.485670 40.949486 29.838943 37.258423
## Sicilia     20.769066   26.280771 19.567067 29.822475 22.041572 26.405092
## Sardegna    18.090299   23.870209 17.118730 27.458106 19.737652 23.992167
##             F-V Giulia  Em-Romag   Toscana    Umbria    Marche     Lazio
## Valle d'Aos
## Liguria
## Lombardia
## T-A Adige
## Veneto
## F-V Giulia
## Em-Romag     17.161340
## Toscana      20.462710 13.633639
## Umbria       16.444273  3.229938 12.747298
## Marche       11.619570  7.863612 10.149527  7.058250
## Lazio        19.339186 16.035040  8.924887 13.815209 11.850270
## Abruzzo       7.929968  9.893715 14.041756  9.138277  4.366646 13.784705
## Molise       49.741923 66.458815 63.476298 65.459694 59.363744 61.105577
## Campania     24.138817 26.454945 14.428444 25.370093 19.900676 15.594281
## Puglia       22.546893 38.473622 35.209642 37.355483 31.121594 32.854864
## Basilicata   27.149851 42.849462 38.834687 41.682837 35.387286 36.308248
## Calabria     22.305625 37.832791 34.090503 36.789912 30.368867 32.337866
## Sicilia      14.435169 27.366017 22.749057 26.190006 19.698683 20.541577
## Sardegna     12.124871 24.845002 20.664946 23.648712 17.132043 18.909299
##               Abruzzo    Molise  Campania    Puglia Basilicata  Calabria
## Valle d'Aos
```

```
## Liguria
## Lombardia
## T-A Adige
## Veneto
## F-V Giulia
## Em-Romag
## Toscana
## Umbria
## Marche
## Lazio
## Abruzzo
## Molise      56.749648
## Campania    21.765268 54.388569
## Puglia      28.743521 28.518810 27.943423
## Basilicata  33.136500 24.909719 30.280305  4.683332
## Calabria    28.274237 29.417512 26.501668  2.740894   5.838502
## Sicilia     18.026894 41.158649 16.811651 12.819551  16.384691 12.189848
## Sardegna    15.577448 43.088240 16.289874 14.622753  18.522605 13.809359
##               Sicilia
## Valle d'Aos
## Liguria
## Lombardia
## T-A Adige
## Veneto
## F-V Giulia
## Em-Romag
## Toscana
## Umbria
## Marche
## Lazio
## Abruzzo
## Molise
## Campania
## Puglia
## Basilicata
## Calabria
## Sicilia
## Sardegna     3.080844
```

Metrica di Manhattan

La metrica di Manhattan è così definita:

$$d_1(X_i, X_j) = \sum_{k=1}^{p} |x_{ik} - x_{jk}|$$

Calcoleremo queste altre metriche utilizzate solo su una parte delle regioni, giusto per mostrare la differenza nei risultati.

Per utilizzare questa metrica, bisogna dare *"manhattan"* come valore all'attributo *method* della funzione *dist()*.

```
x <- tabella[1:5, 1:5]
x
```

```
##             lavoro famiglia studio asilo altro
## Piemonte      28.3     46.6    4.8  18.1  2.16
## Valle d'Aos   26.1     53.2    1.9  16.6  2.20
## Liguria       30.9     44.8    2.8  17.7  3.86
## Lombardia     38.8     47.3    3.1   9.4  1.57
## T-A Adige     22.1     50.9    2.6  22.4  1.90
```

```
dist(x, method = "manhattan", diag = FALSE, upper=FALSE)
```

```
##             Piemonte Valle d'Aos Liguria Lombardia
## Valle d'Aos    13.24
## Liguria         8.50       16.86
## Lombardia      22.19       27.63   21.29
## T-A Adige      17.26       13.10   21.76     34.13
```

Metrica di Chebycev o del massimo

La metrica di Chebycev è definita nella seguente maniera:

$$d_\infty(X_i, X_j) = max_{k=1,...,p}|X_{ik} - X_{jk}|$$

Per utilizzare questa metrica, bisogna dare *"maximum"* come valore all'attributo *method* della funzione *dist()*.

```
dist(x, method = "maximum", diag = FALSE, upper=FALSE)
```

```
##             Piemonte Valle d'Aos Liguria Lombardia
## Valle d'Aos      6.6
## Liguria          2.6        8.4
## Lombardia       10.5       12.7     8.3
## T-A Adige        6.2        5.8     8.8      16.7
```

Metrica di Minkowski

La metrica di Minkowsi è definita come segue:

$$d_r(X_i, X_j) = [\sum_{k=1}^{p} |x_{ik} - x_{jk}|^r]^{\frac{1}{r}}$$

con $r \geq 1$. Se $r = 2$ si ottiene la metrica euclidea, se invece $r = 1$ allora si ottiene la metrica del valore assoluto mentre se $r = \infty$ si ottiene la metrica di Chebycev. Per utilizzare questa metrica, bisogna dare *"minkowski"* come valore all'attributo *method* della funzione *dist()* e bisogna fornire il parametro r. Nell'esempio seguente abbiamo posto $r = 4$

```
dist(x, method = "minkowski", 4, diag = FALSE, upper=FALSE)
```

```
##              Piemonte Valle d'Aos   Liguria Lombardia
## Valle d'Aos  6.684634
## Liguria      2.996038    8.619268
## Lombardia   11.565630   13.151498  9.659892
## T-A Adige    6.836819    6.134183  9.423024 18.065344
```

Metrica di Canberra

La metrica di Canberra è definita come segue:

$$d_c(X_i, X_j) = \sum_{k=1}^{p} \frac{|x_{ik} - x_{jk}|}{|x_{ik} + x_{jk}|}$$

Per utilizzare questa metrica, bisogna dare *"canberra"* come valore all'attributo *method* della funzione *dist()*.

```
dist(x, method = "canberra", diag = FALSE, upper=FALSE)
```

```
##              Piemonte Valle d'Aos   Liguria Lombardia
## Valle d'Aos 0.5918112
## Liguria     0.6203357   0.6674115
## Lombardia   0.8536681   0.9384240 0.9193390
## T-A Adige   0.6346280   0.4825259 0.7243004 0.9025051
```

Metrica di Jaccard

In R è disponibile la distanza di Jaccard solo per vettori binari, diversi da quelli a nostra disposizione. Non è quindi possibile calcolarla.

Misure di similarità

In alternativa alla matrice delle distanze, è possibile utilizzare una matrice delle similarità. Una **misura di similarità** fornisce un valore numerico compreso tra 0 e 1 e permette di definire quantitativamente la somiglianza o differenza fra due individui. In generale, una funzione a valori reali $s_{ij} = s(X_i, X_j)$ è detta misura di similarità se e soltanto se essa sosddisfa le seguenti condizioni:

1. $s(X_i, X_i) = 1$;
2. $0 \leq s(X_i, X_j) \leq 1$;
3. $s(X_i, X_j) = s(X_j, X_i)$ per ogni X_i e X_j

La quantità s_{ij} è detta **coefficiente di similarità**. La principale differenza fra una misura di distanza e una di similarità è che la prima è sempre maggiore o uguale di 0 mentre la seconda risulta essere compresa fra -1 e 1. È sempre possibile trasformare una misura di distanza in una misura di similarità nella seguente maniera:

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

Misura di non omogeneità totale

Consideriamo un insieme di n individui e supponiamo esista un insieme di p caratteristiche osservabili e possedute da ogni individuo. Alla matrice delle misure è possibile associare una matrice W_x di cardinalità $p*p$ detta **matrice delle varianze e covarianze** dove il generico elemento w_{rl} è definito nella seguente maniera:

$$w_{rl} = \frac{1}{n-1} \sum_{i=1}^{n} (x_{ir} - \bar{x_r})(x_{il} - \bar{x_l}) \qquad (r, l = 1, 2, ..., p)$$

Se $r = l$ allora l'elemento w_{rl} sarà uguale alla varianza campionaria relativa alla caratteristica r-esima effettuata su tutti gli n individui, mentre se $r \neq l$ l'elemento w_{rl} è uguale alla covarianza campionaria tra la caratteristica r-esima e la caratteristica l-esima effettuata su tutti gli n individui.

Dalla matrice delle varianze e covarianze è possibile ricavare una matrice chiamata **matrice statistica di non omogeneità**, indicata con H_I, per l'insieme I di individui di cardinalità $p * p$ dove il generico elemento è definito nella seguente maniera:

$$h_{rl} = \sum_{i=1}^{n} (x_{ir} - \bar{x_r})(x_{il} - \bar{x_l}) = (n-1)w_{rl} \qquad (r, l = 1, 2, ..., p)$$

In questa matrice, quando $r = l$ l'elemento h_{rr} corrisponde a n-1 volte la varianza campionaria della caratteristica r-esima su tutti gli individui. Una **misura di non omogeneità**

statistica dell'insieme I di individui corrisponde alla **traccia** delle matrice H_I ed è definita come segue:

$$trH_I = \sum_{r=1}^{p} h_{rr} = (n-1)\sum_{r=1}^{p} s_r^2$$

oppure in alternativa:

$$trH_I = \sum_{i=1}^{n} d_2^2(X_i, \bar{X})$$

dove d_2 indica la distanza euclidea e $\bar{X} = (\bar{x_1}, \bar{x_2}, ..., \bar{x_p})$ è un vettore dove il generico elemento $\bar{x_j}$ rappresenta la media campionaria della caratteristca j-esima sugli n individui.

Misure di non omogeneità tra cluster

Oltre a misure di non omogeneità relative all'insieme totale, vanno definite anche misure di non omogeneità all'interno dei cluster e tra cluster distinti. Infatti, al termine della classificazione, gli individui che appartengono allo stesso cluster dovrebbero essere il più omogenei tra loro e il più differenti da quelli appartenenti a cluster diversi.

È quindi possibile definire una **misura di non omogeneità totale** che tenga conto della non omogeneità interna ai cluster (*within*) e della non omogeneità fra cluster (*between*). Denotiamo questa misura di non omogeneità totale con T e la definiamo come:

$$T = S + B$$

dove S corrisponde alla somma delle matrici di non omogeneità statistica relative ai singoli cluster ed B la matrice di non omogeneità statistica tra i vari cluster considerati. Le matrici T, S e B risultano avere tutte e tre la stessa cardinalità. Per ogni fissata matrice di dati, anche la matrice T è fissata mentre invece, le matrici S e B dipendono strettamente dalla partizione in cluster considerata. Per ogni partizione dell'insieme degli individui in un numero fissato di cluster, otteniamo che:

$$trT = trS + trB$$

o, in maniera equivalente:

$$1 = \frac{trS}{trT} + \frac{trB}{trT}$$

e siccome abbiamo detto che trT è fissata allora i cluster dovrebbero essere individuati in modo da **minimizzare*** la misura di non omogeneità statistica all'interno dei cluster* e **massimizzare** *la misura di non omogeneità statistica fra cluster.*

Metodi di ottimizzazione

Scelta la misura di distanza (o la similarità in alternativa) bisogna scegliere un algoritmo di raggruppamento delle unità osservate. I metodi di raggruppamento si suddividono in tre tipi:

- **Metodo di enumerazione completa**;
- **Metodi gerarchici**;
- **Metodi non gerarchici**.

Il primo metodo elencato, quello di enumerazione completa (di ottimizzazione), è computazionalmente oneroso da applicare in quanto bisogna calcolare la funzione obiettivo per ogni partizione dell'insieme totale e questo va bene solo quando abbiamo un insieme di dati piccolo. Vengono per questo adottati metodi di raggruppamento gerarchici e non gerarchici che operano su sottoclassi delle partizioni degli individui in cluster.

Metodi non gerarchici

Si è scelto di effettuare prima la suddivisione in cluster utilizzando dei metodi non gerarchici in modo da stabilire il numero di partizioni ottimale, sfruttandolo successivamente nell'analisi mediante metodi gerarchici. I metodi non gerarchici consentono di riallocare gli individui già classificati precedentemente nell'analisi e l'obiettivo finale di queste tecniche è quello di ottenere un'unica partizione degli individui di partenza. Per quanto riguarda il numero di cluster da individuare, in molti metodi non gerarchici si assume che esso sia fissato da chi sta facendo l'analisi mentre in altri questo numero è determinato durante l'analisi stessa. Il metodo non gerarchico più utilizzato è quello del **k-means** (Hartigan e Wong, 1979) nel quale il numero di cluster è specificato a priori e fornisce in output un'unica partizione. I passi dell'algoritmo sono i seguenti:

1. Fissare il numero m dei cluster specificando m punti iniziali (scegliendo alcuni individui o prendendo una configurazione determinata con una tecnica gerarchica) che inducono una prima partizione provvisoria;
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 (centroide) di ognuno degli m gruppi così ottenuti. Tali centroidi costituiscono i punti di riferimento per i nuovi cluster;
4. Valutare ogni distanza di ogni unità dal centroide ottenuto al punto precedente;
5. Ricalcolare i centroidi dei k gruppi così ottenuti;
6. Ripeto il procedimento dal punto 4 fino ad ottenere una partzione che risulta essere stabile oppure dopo aver raggiunto un determinato numero di iterazioni.

Per garantire la convergenza della procedura iterativa, viene utilizzata la distanza euclidea come misura di distanza fra i vettori delle caratteristiche e i centroidi e viene considerata *la matrice contenente i quadrati delle distanze euclidee*. Il metodo k-means risulta essere molto veloce nei calcoli e lascia estrema libertà agli individui di allontanarsi o raggrupparsi. Uno svantaggio riguarda il fatto che la classificazione finale può essere influenzata dalla scelta iniziale degli m vettori delle caratteristiche come punti di riferimento. Infatti vi è il rischio di cadere in un ottimo locale e non in uno globale.

Per utilizzare il metodo k-means in R bisogna utilizzare la funzione *kmeans(X, centers, iter.max=N, nstart=M)*. Per la nostra analisi utilizzeremo un valore di centers inizialmente pari a 2, un numero di iterazioni pari a 25 e un valore di nstart uguale a 10.

```
km <- kmeans(tabella, centers = 2, iter.max = 25, nstart = 10)
km
```

```
## K-means clustering with 2 clusters of sizes 6, 14
##
## Cluster means:
##     lavoro famiglia   studio    asilo    altro
## 1 22.31667 28.36667 1.333333 43.75000 4.226667
## 2 31.51429 45.13571 3.007143 17.13571 3.225000
##
## Clustering vector:
##    Piemonte Valle d'Aos     Liguria   Lombardia   T-A Adige      Veneto
##           2           2           2           2           2           2
##  F-V Giulia    Em-Romag     Toscana      Umbria      Marche       Lazio
##           2           2           2           2           2           2
##     Abruzzo      Molise    Campania      Puglia  Basilicata    Calabria
##           2           1           2           1           1           1
##     Sicilia    Sardegna
##           1           1
##
## Within cluster sum of squares by cluster:
## [1] 1208.644 1331.883
##  (between_SS / total_SS =  64.1 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
## [5] "tot.withinss" "betweenss"    "size"         "iter"
## [9] "ifault"
```

La suddivisione in cluster individuata è la seguente:

- $C_1 = Piemonte, Valled'Aosta, Liguria, Lombardia, Trentino - AltoAdige, Veneto, Friuli - Venez$

- $C_2 = Molise, Puglia, Basilicata, Calabria, Sicilia, Sardegna$

```
km$totss
```

```
## [1] 7067.801
```

```
km$tot.withinss
```

```
## [1] 2540.527
```

```
km$betweenss
```

```
## [1] 4527.274
```

```
km$betweenss/km$totss
```

```
## [1] 0.6405491
```

Possiamo notare che la misura di non omogeneità totale è pari a 7067.801, mentre la somma delle misure di non omogeneità totale interne è uguale a 2540.527, la misura di non omogeneità tra cluster diversi è 4527.274 e il rapporto fra $\frac{trB}{trT}$ risulta essere pari a 0.6405, un valore non sufficiente. Proviamo quindi ad aumentare il numero di centri a 3.

```
km <- kmeans(tabella, centers = 3, iter.max = 25, nstart = 10)
km
```

```
## K-means clustering with 3 clusters of sizes 1, 14, 5
##
## Cluster means:
##     lavoro famiglia   studio    asilo altro
## 1 10.90000 18.60000 0.700000 67.10000 2.750
## 2 31.51429 45.13571 3.007143 17.13571 3.225
## 3 24.60000 30.32000 1.460000 39.08000 4.522
##
## Clustering vector:
##    Piemonte Valle d'Aos     Liguria   Lombardia   T-A Adige      Veneto
##           2           2           2           2           2           2
##  F-V Giulia    Em-Romag     Toscana      Umbria      Marche       Lazio
##           2           2           2           2           2           2
##     Abruzzo      Molise    Campania      Puglia  Basilicata    Calabria
##           2           1           2           3           3           3
```

```
##     Sicilia    Sardegna
##           3           3
##
## Within cluster sum of squares by cluster:
## [1]    0.0000 1331.8833  280.4055
##  (between_SS / total_SS =  77.2 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
## [5] "tot.withinss" "betweenss"    "size"         "iter"
## [9] "ifault"
```

La suddivisione in cluster individuata è la seguente:

- $C_1 = Piemonte, Valled'Aosta, Liguria, Lombardia, Trentino - AltoAdige, Veneto, Friuli - Venez$
- $C_2 = Molise$
- $C_3 = Puglia, Basilicata, Calabria, Sicilia, Sardegna$

```
km$totss
```

```
## [1] 7067.801
```

```
km$tot.withinss
```

```
## [1] 1612.289
```

```
km$betweenss
```

```
## [1] 5455.512
```

```
km$betweenss/km$totss
```

```
## [1] 0.7718826
```

Possiamo notare che la misura di non omogeneità totale è pari a 7067.801, mentre la somma delle misure di non omogeneità totale interne è uguale a 1612.28, la misura di non omogeneità tra cluster diversi è 5455.51 e il rapporto fra $\frac{trB}{trT}$ risulta essere pari a 0.771, un valore più che sufficiente considerando il numero di centri utilizzato.

Nell'analisi che effettueremo con i metodi gerarchici utilizzeremo lo stesso numero di cluster.

È possibile anche scegliere i centroidi come punti di riferimento:

```r
d<-dist(tabella, method="euclidean", diag = TRUE, upper = TRUE)
d2 <- d^2
tree <- hclust(d2, method = "centroid")
taglio <- cutree(tree, k = 3, h = NULL)
tagliolist <- list(taglio)
centroidiIniziali <- aggregate(tabella, tagliolist, mean)[,-1]
kmeans(tabella, centers = centroidiIniziali, iter.max = 25)
```

```
## K-means clustering with 3 clusters of sizes 14, 5, 1
##
## Cluster means:
##     lavoro famiglia   studio    asilo altro
## 1 31.51429 45.13571 3.007143 17.13571 3.225
## 2 24.60000 30.32000 1.460000 39.08000 4.522
## 3 10.90000 18.60000 0.700000 67.10000 2.750
##
## Clustering vector:
##    Piemonte Valle d'Aos     Liguria   Lombardia   T-A Adige      Veneto
##           1           1           1           1           1           1
##  F-V Giulia    Em-Romag     Toscana      Umbria      Marche       Lazio
##           1           1           1           1           1           1
##     Abruzzo      Molise    Campania      Puglia  Basilicata    Calabria
##           1           3           1           2           2           2
##     Sicilia    Sardegna
##           2           2
##
## Within cluster sum of squares by cluster:
## [1] 1331.8833  280.4055    0.0000
##  (between_SS / total_SS =  77.2 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
## [5] "tot.withinss" "betweenss"    "size"         "iter"
## [9] "ifault"
```

Com'è possibile vedere anche utilizzando i centroidi abbiamo esattamente la stessa suddivisione in cluster presentata precedentemente.

Metodi gerarchici

I metodi gerarchici si suddividono in: **agglomerativi** e **divisivi**. In quelli di tipo agglomerativo si parte da una situazione in cui si hanno tanti cluster quanti gli individui per poi

giungere ad un unico cluster dove sono riuniti tutti. Nei metodi gerarchici di tipo divisivo invece, si parte da una situazione in cui si ha un solo cluster con tutti gli individui per poi raggiungere, attraverso divisioni, una situazione in cui si hanno tanti cluster quanti sono gli individui. L'obiettivo finale dei metodi gerarchici è quello di ottenere una sequenza di partizioni che possono essere rappresentate graficamente mediante una struttura chiamata **dendrogramma**, sul quale asse delle ordinate sono riportati i livelli di distanza e su quello delle ascisse i singoli elementi. Ad ogni livello corrisponde una partizione.

In molti metodi gerarchici di tipo agglomerativo l'algoritmo ha una struttura comune:

1. A partire dalla matrice dei dati, dobbiamo costruire la matrice delle distanze D (oppure quella delle similiarità S) tra individui considerati come singoli cluster distinti.
2. Individuare la coppia di cluster meno distanti (o più somiglianti) e raggruppare in un unico cluster i due cluster meno distanti (o più somiglianti); calcolare inoltre la distanza (o similarità) di questo cluster originato dall'agglomerazione da tutti gli altri gruppi già esistenti.
3. Costruire una nuova matrice di distanza (o similarità) che risulterà ridotta di una riga e di una colonna rispetto a quella che la precede (questo perchè le due righe e le due colonne dei singoli cluster sono state agglomerate in una singola rigoa e una singola colonna con le nuove distanze).
4. Operare sulla nuova matrice a partire dallo step 2 fino ad esaurire tutte le possibilità di raggruppamento (fino a quando non si ottiene una matrice 2x2).
5. Rappresentare graficamente il processo di agglomerazione attraverso un dendrogramma.

Quello che cambia nei diversi metodi è il passo 1 e il passo 2. Nel primo bisogna scegliere fra la similarità o la distanza mentre le tecniche adottate nel secondo passo sono quelle che differenzieranno di molto i vari metodi e da cui dipende il nome del metodo stesso.

L'analisi gerarchica di tipo agglomerativo viene effettuata in R attraverso la funzione *hclust(d, method =)* dove:

- **d** è rappresenta un oggetto creato tramite la funzione *dist()*;
- **method** seleziona il metodo gerarchico agglomerativo. Il valore di default è *complete*.

Abbiamo 5 diversi metodi fra cui scegliere:

1. Metodo del legame singolo (*single*);
2. Metodo del legame completo (*complete*);
3. Metodo del legame medio (*average*);
4. Metodo del centroide (*centroid*);
5. Metodo della mediana (*median*)

Metodo del legame singolo

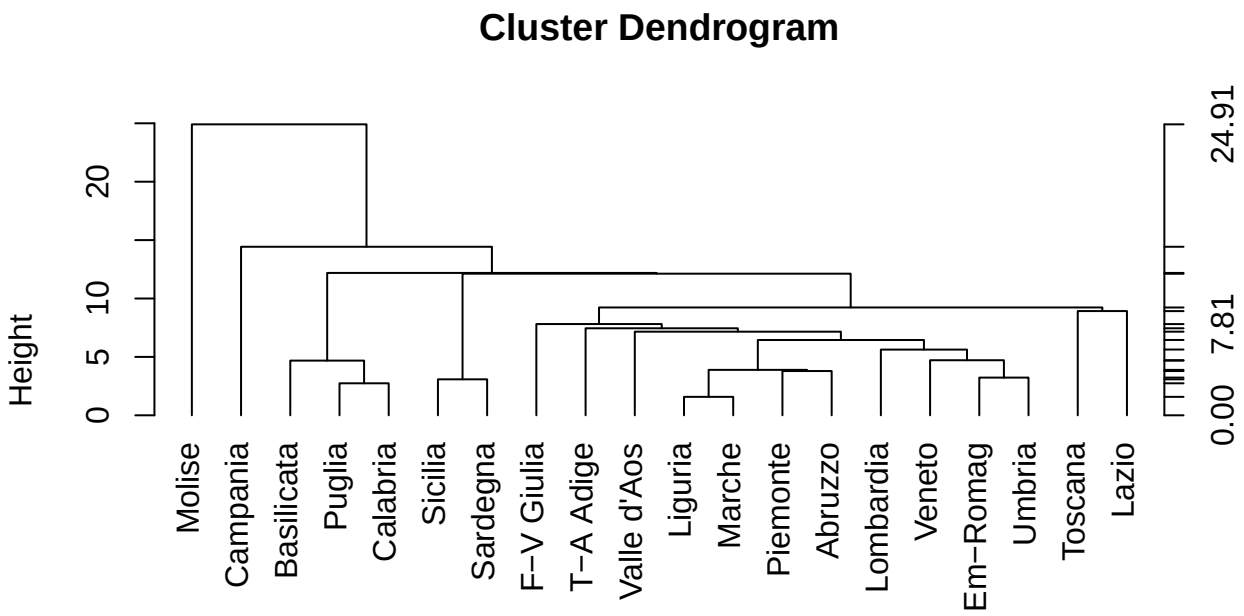
In questo metodo la distanza fra due gruppi G_1 e G_2 contenenti rispettivamente n_1 e n_2 individui è definita come la minima fra tutte le distanze che si possono calcolare tra ogni individuo di G_1 e ogni individuo di G_2. Vengono poi agglomerate i gruppi con distanza minore come descritto precedentemente. Il metodo del legame singolo ha come vantaggio quello di consentire di individuare gruppi di qualsiasi forma. Attraverso R adesso procediamo all'analisi dei cluster attraverso questo metodo:

```
d <- dist(tabella, method = "euclidean", diag = TRUE, upper = FALSE)
hls <- hclust(d, method = "single")
str(hls)
```

```
## List of 7
##  $ merge      : int [1:19, 1:2] -3 -16 -19 -8 -1 1 -17 -6 -4 6 ...
##  $ height     : num [1:19] 1.58 2.74 3.08 3.23 3.79 ...
##  $ order      : int [1:20] 14 15 17 16 18 19 20 7 5 2 ...
##  $ labels     : chr [1:20] "Piemonte" "Valle d'Aos" "Liguria" "Lombardia" ...
##  $ method     : chr "single"
##  $ call       : language hclust(d = d, method = "single")
##  $ dist.method: chr "euclidean"
##  - attr(*, "class")= chr "hclust"
```

Procediamo adesso alla stampa del dendrogramma:

```
plot(hls, hang = -1, xlab = "Metodo gerarchico agglomerativo",
     sub="del legame singolo")
axis(side = 4, at= round(c(0, hls$height), 2))
```



Metodo gerarchico agglomerativo
del legame singolo

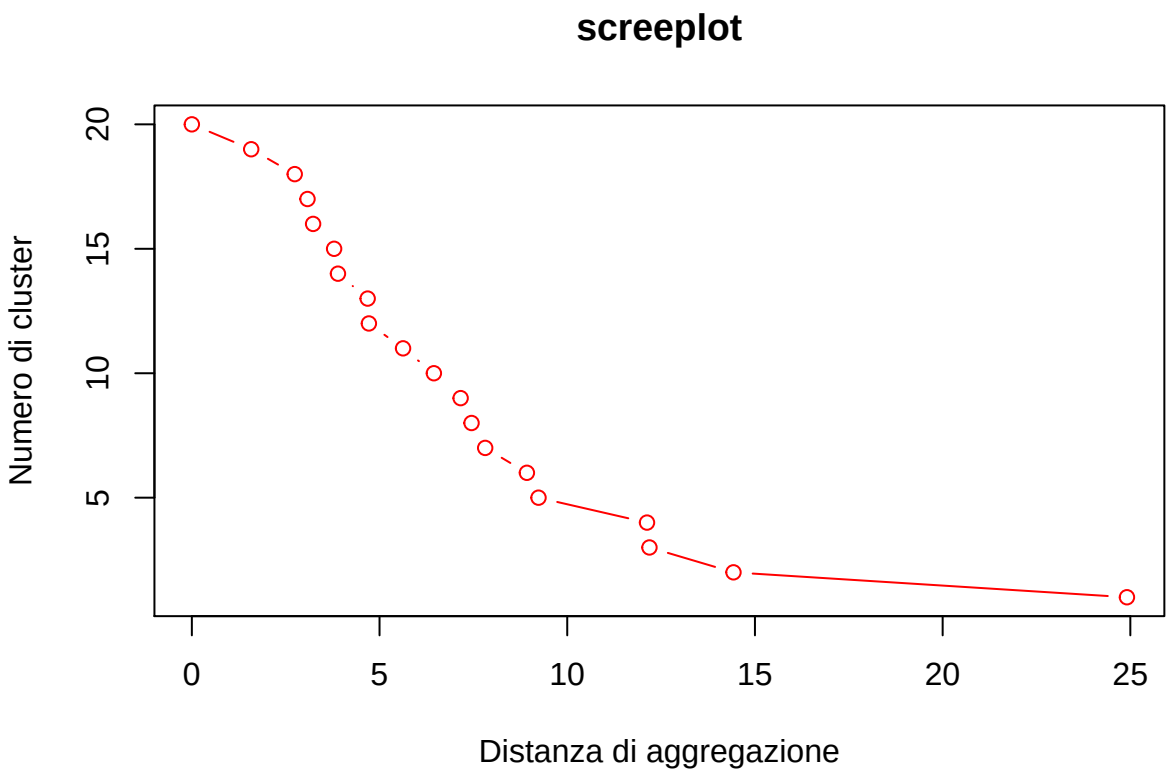
La funzione *axis(side = 4, at= round(c(0, hls$height), 2))* ci permette di costruire l'asse delle altezze alla destra del grafico, arrotondando alla seconda cifra decimale.

Per scegliere una buona partizione del dendrogramma, consideriamo un grafico detto **screeplot**, sul quale asse delle ordinate vengono posti i numeri di gruppi ottenibili con il meotodo gerarchico e sull'asse ascisse vengono le distanze a cui avvengono le successive aggregazioni fra gruppi. Al fine di fornire misure quantitative degli incrementi riscontrati tra le distanze a cui avvengono le successive agglomerazioni, consideriamo la seguente quantità:

$$\delta_k = d_{k-1} - d_k \qquad (k = 2, ..., n)$$

dove d_k rappresenta il livello di distanza a cui è stata effettuata l'agglomerazione in k gruppi e n è il numero iniziale di individui. Quando δ_k risulta sufficientemente elevato allora i gruppi sono dissimili fra loro ed è possibile tagliare il dendrogramma all'altezza corrispondente alla partizione in k gruppi. È possibile utilizzare lo screeplot solo per i metodi gerarchici del legame singolo, medio e completo in quanto con i metodi del centroide e della mediana, le successive agglomerazioni potrebbero verificarsi ad un livello di distanza minore o uguale rispetto alle precedenti. Questa procedura però non fornisce sempre la suddivisione in cluster più adeguata, infatti è sempre preferibile utilizzare le misure di non omogeneità statistiche.

```
plot(c(0, hls$height), seq(20, 1), type = "b", main = "screeplot",
     xlab = "Distanza di aggregazione", ylab = "Numero di cluster", col = "red")
```




```
hls$height
```

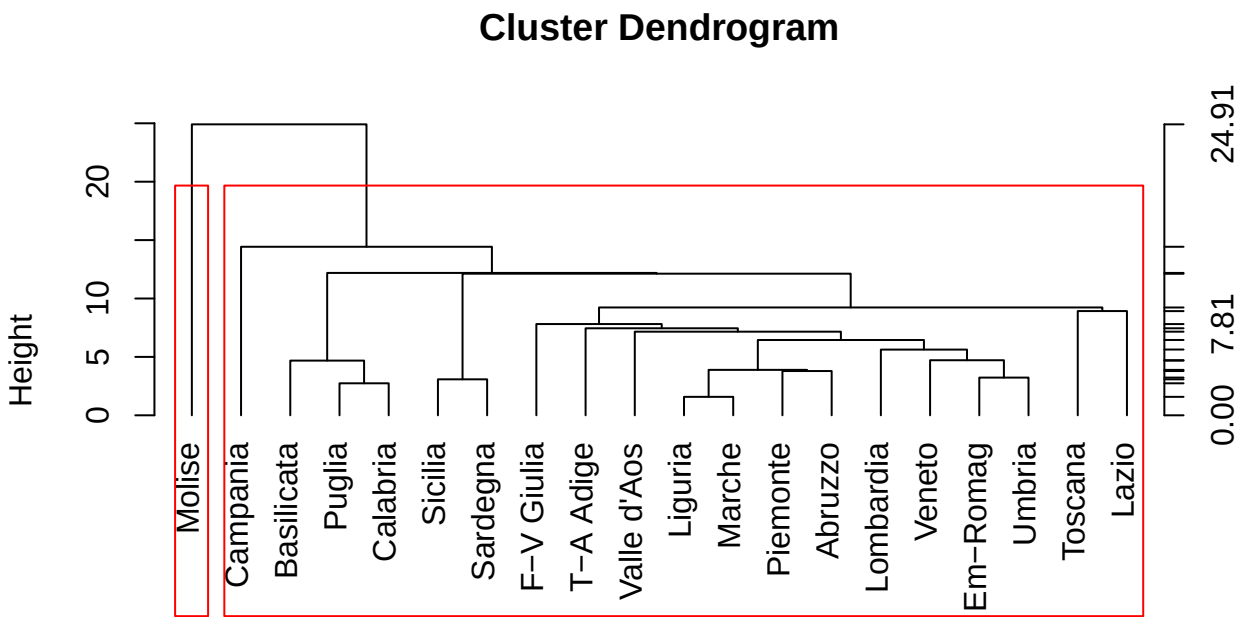
```
##  [1]  1.581139  2.740894  3.080844  3.229938  3.789670  3.892300  4.683332
##  [8]  4.715930  5.625913  6.447240  7.160000  7.450503  7.814243  8.924887
## [15]  9.234479 12.124871 12.189848 14.428444 24.909719
```

In questo caso il max lo abbiamo fra:

$$\delta_2 = h_1 - h_2 = 24.909719 - 14.428444 = 10.48127$$

Quindi il valore di k per il quale δ_k è massimo si ha con $k = 2$. Applichiamo la suddivisione al dendrogramma utilizzando la funzione *rect.hclust()*:

```
plot(hls, hang = -1, xlab = "Metodo gerarchico agglomerativo",
     sub="del legame singolo")
axis(side = 4, at= round(c(0, hls$height), 2))
rect.hclust(hls, k=2)
```



Metodo gerarchico agglomerativo
del legame singolo

Per ottenere la suddivisione in cluster oltre che graficamente, è possibile utilizzare la funzione *cutree()*.

```
cutree(hls, k=2)
```

```
##    Piemonte Valle d'Aos     Liguria   Lombardia   T-A Adige      Veneto
##           1           1           1           1           1           1
##  F-V Giulia    Em-Romag     Toscana      Umbria      Marche       Lazio
##           1           1           1           1           1           1
##     Abruzzo      Molise    Campania      Puglia  Basilicata    Calabria
##           1           2           1           1           1           1
##     Sicilia    Sardegna
##           1           1
```

Calcoliamo adesso la misura di non omogeneità statistica totale

```
n <- nrow(tabella)
trHI <- (n-1) * sum(apply(tabella, 2, var))
trHI
```

```
## [1] 7067.801
```

La misura di non omogeneità statistica totale è pari a 7097.801. Procediamo adesso al calcolo delle misure di non omogeneità statistiche e lo facciamo per la suddivisione in 2,3,4 cluster.

```
d <- dist(tabella, method="euclidean", diag = TRUE, upper=TRUE)
d2<-d^2
tree <- hclust(d2, method = "single")
taglio <-cutree(tree, k=2, h=NULL)
num <- table(taglio)
tagliolist <- list(taglio)

agvar <- aggregate(tabella, tagliolist, var)[,-1]

trH1<-(num[[1]]-1) * sum(agvar[1,])
if(is.na(trH1))
  trH1 <- 0

trH2<-(num[[2]]-1) * sum(agvar[2,])
if(is.na(trH2))
  trH2 <- 0

trS <- trH1 + trH2

trB <- trHI-trH1-trH2
trB
```

```
## [1] 2681.522
```

```
trB/trHI
```

```
## [1] 0.3793997
```

Con la suddivisione in due cluster otteniamo una misura di non omogeneità tra cluster pari a 2681.522. Questa suddivisione ricordiamo, ci è stata suggerita anche dallo scatterplot. Il rapporto fra trB/trHI è pari a 0.3793997 un valore decisamente basso che ci porta all'aumentare il numero di cluster.

```
d <- dist(tabella, method="euclidean", diag = TRUE, upper=TRUE)
d2<-d^2
tree <- hclust(d2, method = "single")
taglio <-cutree(tree, k=3, h=NULL)
num <- table(taglio)
tagliolist <- list(taglio)
```

```
agvar <- aggregate(tabella, tagliolist, var)[,-1]

trH1<-(num[[1]]-1) * sum(agvar[1,])
if(is.na(trH1))
  trH1 <- 0

trH2<-(num[[2]]-1) * sum(agvar[2,])
if(is.na(trH2))
  trH2 <- 0

trH3<-(num[[3]]-1) * sum(agvar[3,])
if(is.na(trH3))
  trH3 <- 0

trS <- trH1 + trH2 + trH3

trB <- trHI-trH1-trH2-trH3
trB/trHI
```

```
## [1] 0.4232395
```

Con la suddivisione in tre cluster, suggeritaci durante l'analisi con i metodi non gerarchici, il rapporto fra trB/trHI è pari a 0.4232935 un valore che, seppur più alto di quello a due cluster risulta comunque essere basso.

```
d <- dist(tabella, method="euclidean", diag = TRUE, upper=TRUE)
d2<-d^2
tree <- hclust(d2, method = "single")
taglio <-cutree(tree, k=4, h=NULL)
num <- table(taglio)
tagliolist <- list(taglio)

agvar <- aggregate(tabella, tagliolist, var)[,-1]

trH1<-(num[[1]]-1) * sum(agvar[1,])
if(is.na(trH1))
  trH1 <- 0

trH2<-(num[[2]]-1) * sum(agvar[2,])
if(is.na(trH2))
  trH2 <- 0

trH3<-(num[[3]]-1) * sum(agvar[3,])
if(is.na(trH3))
```

```
  trH3 <- 0

trH4<-(num[[4]]-1) * sum(agvar[4,])
if(is.na(trH4))
  trH4 <- 0

trS <- trH1 + trH2 + trH3 + trH4

trB <- trHI-trH1-trH2-trH3-trH4
trB/trHI
```

```
## [1] 0.7699938
```

La suddivisione in quattro cluster ci ha portato al risultato migliore fra le tre suddivisioni con un rapporto fra trB/trHI pari a 0.7699938, che ricade nel range di valori accettabili.

Metodo del legame completo

In questo metodo la distanza tra i gruppi G_1 e G_2 è definita come la massima tra tutte le $n_1 n_2$ distanze che si possono calcolare tra ogni individuo di G_1 e ogni individuo di G_2. A livello 0, viene considerato inizialmente un insieme di n cluster, uno per ogni individuo. Nel passo successivo, si cerca il coefficiente di distanza minima all'interno della matrice delle distanze D e si raggruppano nello stesso cluster i due individui ad esso associati. A livello 1 si modifica la matrice delle distanze valutando le distanze dal cluster appena creato con gli altri mediante la seguente relazione:

$$d_{(ij),k} = max(d_{ik}, d_{jk})$$

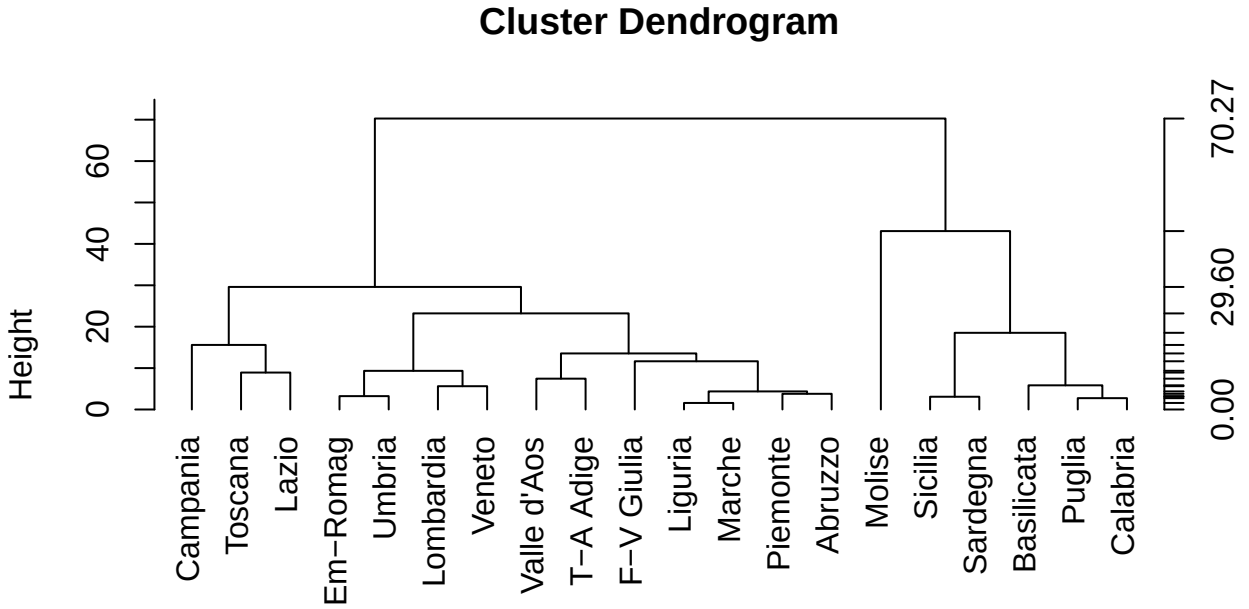
Ad ogni passo successivo vengono ripetute le stesse azioni, funo ad ottenere un unico cluster formato da tutti gli individui.

```
d <- dist(tabella, method = "euclidean",
          diag = TRUE, upper = FALSE)
hlc <- hclust(d, method = "complete")
str(hlc)
```

```
## List of 7
##  $ merge      : int [1:19, 1:2] -3 -16 -19 -8 -1 1 -4 -17 -2 -9 ...
##  $ height     : num [1:19] 1.58 2.74 3.08 3.23 3.79 ...
##  $ order      : int [1:20] 15 9 12 8 10 4 6 2 5 7 ...
##  $ labels     : chr [1:20] "Piemonte" "Valle d'Aos" "Liguria" "Lombardia" ...
##  $ method     : chr "complete"
##  $ call       : language hclust(d = d, method = "complete")
##  $ dist.method: chr "euclidean"
##  - attr(*, "class")= chr "hclust"
```

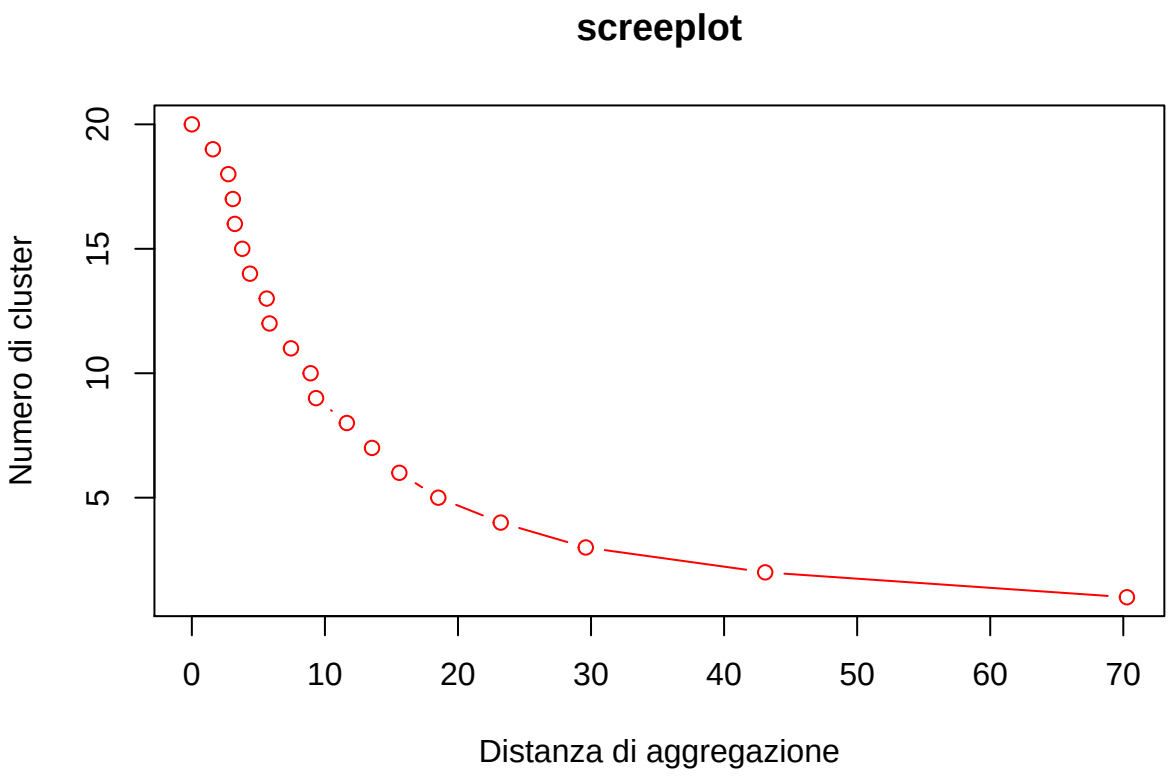
Procediamo adesso alla stampa del dendrogramma:

```
plot(hlc, hang = -1, xlab = "Metodo gerarchico agglomerativo",
     sub="del legame completo")
axis(side = 4, at= round(c(0, hlc$height), 2))
```



Come fatto in precedenza, procediamo alla stampa dello screeplot:

```
plot(c(0, hlc$height), seq(20, 1), type = "b",
     main = "screeplot", xlab = "Distanza di aggregazione",
     ylab = "Numero di cluster", col = "red")
```




```
hlc$height
```

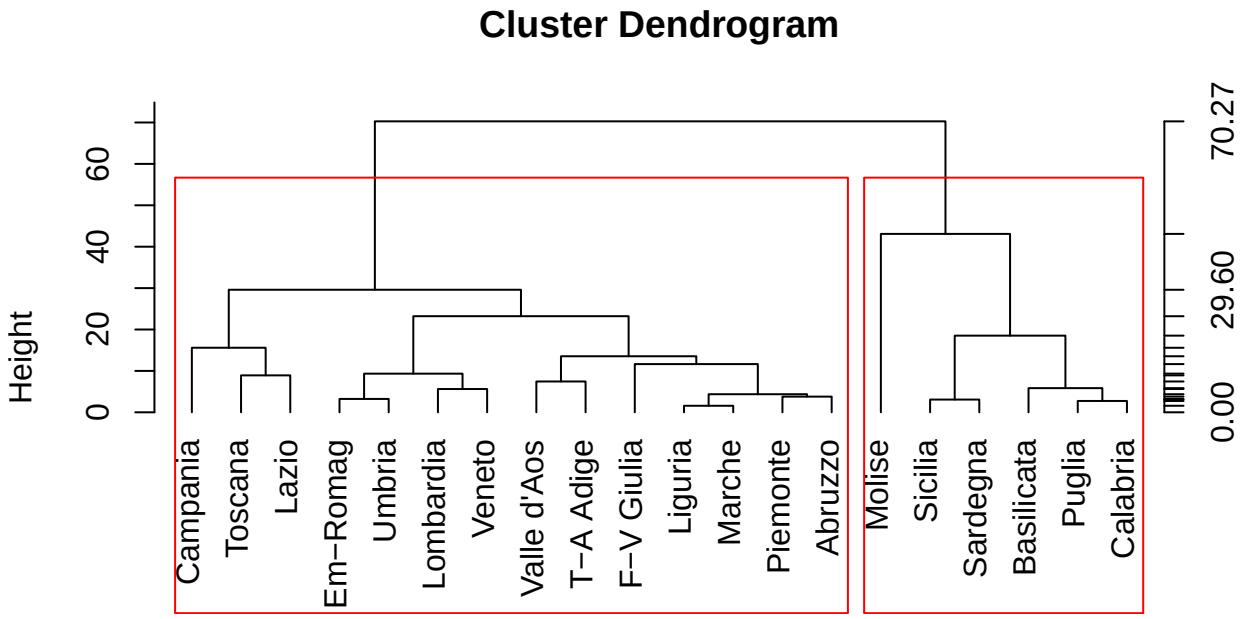
```
##  [1]  1.581139  2.740894  3.080844  3.229938  3.789670  4.366646  5.625913
##  [8]  5.838502  7.450503  8.924887  9.335331 11.648966 13.542688 15.594281
## [15] 18.522605 23.213791 29.604001 43.088240 70.274764
```

Anche in questo caso il max lo abbiamo fra:

$$\delta_2 = h_1 - h_2 = 70.274764 - 43.088240 = 27.186524$$

Quindi il valore di k per il quale δ_k è massimo si ha con $k = 2$. Applichiamo la suddivisione al dendrogramma utilizzando la funzione *rect.hclust()*:

```
plot(hlc, hang = -1, xlab = "Metodo gerarchico agglomerativo",
     sub="del legame completo")
axis(side = 4, at= round(c(0, hlc$height), 2))
rect.hclust(hlc, k=2)
```



Metodo gerarchico agglomerativo
del legame completo

Per ottenere la suddivisione in cluster oltre che graficamente, è possibile utilizzare la funzione *cutree()*.

```
cutree(hlc, k=2)
```

```
##    Piemonte Valle d'Aos     Liguria   Lombardia   T-A Adige      Veneto
##           1           1           1           1           1           1
##  F-V Giulia    Em-Romag     Toscana      Umbria      Marche       Lazio
##           1           1           1           1           1           1
##     Abruzzo      Molise    Campania      Puglia  Basilicata    Calabria
##           1           2           1           2           2           2
##     Sicilia    Sardegna
##           2           2
```

Procediamo adesso all'analisi utilizzando le misure di non omogeneità statistiche, in quanto, anche in precedenza sono risultate essere più precise rispetto allo screeplot. Faremo l'analisi per la suddivisione in 2,3 e 4 cluster.

Siccome la misura di non omogeneità statistica totale è uguale a quella calcolata in precedenza evitiamo di ricalcolarla.

Procediamo al calcolo delle misure di omogeneità statistiche per le varie suddivisioni:

```
d <- dist(tabella, method="euclidean",
          diag = TRUE, upper=TRUE)
d2<-d^2
tree <- hclust(d2, method = "complete")
taglio <-cutree(tree, k=2, h=NULL)
num <- table(taglio)
tagliolist <- list(taglio)

agvar <- aggregate(tabella, tagliolist, var)[,-1]

trH1<-(num[[1]]-1) * sum(agvar[1,])
if(is.na(trH1))
  trH1 <- 0

trH2<-(num[[2]]-1) * sum(agvar[2,])
if(is.na(trH2))
  trH2 <- 0

trS <- trH1 + trH2

trB <- trHI-trH1-trH2
trB/trHI
```

```
## [1] 0.6405491
```

Con la suddivisione in due cluster, suggeritaci dallo scatterplot, il rapporto tra $\frac{trB}{trT}$ non è risultato essere abbastanza alto.

```
d <- dist(tabella, method="euclidean",
          diag = TRUE, upper=TRUE)
d2<-d^2
tree <- hclust(d2, method = "complete")
taglio <-cutree(tree, k=3, h=NULL)
num <- table(taglio)
tagliolist <- list(taglio)

agvar <- aggregate(tabella, tagliolist, var)[,-1]

trH1<-(num[[1]]-1) * sum(agvar[1,])
if(is.na(trH1))
  trH1 <- 0

trH2<-(num[[2]]-1) * sum(agvar[2,])
if(is.na(trH2))
```

```
  trH2 <- 0

trH3<-(num[[3]]-1) * sum(agvar[3,])
if(is.na(trH3))
  trH3 <- 0

trS <- trH1 + trH2 + trH3

trB <- trHI-trH1-trH2-trH3
trB/trHI
```

```
## [1] 0.7718826
```

Con la suddivisione in tre cluster, suggeritaci durante l'analisi con i metodi non gerarchici, invece il rapporto tra $\frac{trB}{trT}$ ha raggiunto un risultato di sufficienza ed è possibile tenerlo in considerazione. Da notare che il valore è pressochè identico a quello trovato con il metodo k-means.

```
d <- dist(tabella, method="euclidean",
          diag = TRUE, upper=TRUE)
d2<-d^2
tree <- hclust(d2, method = "complete")
taglio <-cutree(tree, k=4, h=NULL)
num <- table(taglio)
tagliolist <- list(taglio)

agvar <- aggregate(tabella, tagliolist, var)[,-1]

trH1<-(num[[1]]-1) * sum(agvar[1,])
if(is.na(trH1))
  trH1 <- 0

trH2<-(num[[2]]-1) * sum(agvar[2,])
if(is.na(trH2))
  trH2 <- 0

trH3<-(num[[3]]-1) * sum(agvar[3,])
if(is.na(trH3))
  trH3 <- 0

trH4<-(num[[4]]-1) * sum(agvar[4,])
if(is.na(trH4))
  trH4 <- 0
```

```
trS <- trH1 + trH2 + trH3 + trH4

trB <- trHI-trH1-trH2-trH3-trH4
trB/trHI
```

```
## [1] 0.8529709
```

Con quest'ultima suddivisione, il rapporto fra $\frac{trB}{trT}$ raggiunge il risultato più alto fra tutti quelli visti con i metodi gerarchici e risulta essere un ottimo risultato dato il numero di cluster scelti.

La relativa suddivisione in cluster è la seguente:

```
cutree(hlc, k=4, h=NULL)
```

```
##    Piemonte Valle d'Aos     Liguria   Lombardia   T-A Adige      Veneto
##           1           1           1           1           1           1
##  F-V Giulia    Em-Romag     Toscana      Umbria      Marche       Lazio
##           1           1           2           1           1           2
##     Abruzzo      Molise    Campania      Puglia  Basilicata    Calabria
##           1           3           2           4           4           4
##     Sicilia    Sardegna
##           4           4
```

$C_1 = \{Campania,\ Lazio,\ Toscana\}$ $C_2 = \{Emilia\ Romagna,\ Umbria,\ Valle\ d'Aosta,\ Veneto,\ Lombardia,\ Trentino\text{-}alto\ Adige,\ Friuli\text{-}venezia\ Giulia,\ Liguria,\ Marche,\ Piemonte,\ Abruzzo\}$ $C_3 = \{Molise\}$ $C_4 = \{Sicilia,\ Sardegna,\ Basilicata,\ Puglia,\ Calabria\}$

Metodo del legame medio

In questo metodo la distanza tra i gruppi G_1 e G_2 è definita come la media aritmetica delle distanze fra tutte le coppie che compongono gli insiemi. All'inizio del metodo, a livello 0 abbiato tanti cluster quanti sono gli individui. Successivamente controlliamo la matrice delle distanze D e raggruppiamo nello stesso cluster gli individui associati alla distanza minima. A livello 1 di modifica la matrice delle distanze valutando le distanze del cluster appena creato e tutti gli altri rimanenti, mediante la seguente relazione:

$$d_{(i,j),k} = \frac{1}{2}(d_{i,k} + d_{j,k}) \qquad (k = 1, 2, ..., n; k \neq i, j)$$

Viene quindi costruita una nuova matrice delle distanze. Ad ogni passo successivo, dopo aver individuato i due cluster da raggruppare, la distanza fra il nuovo cluster G_{uv} e un altro cluster G_z è definita nella seguente maniera:

$$d_{(uv),z} = \frac{N_u}{N_u + N_v} d_{uz} + \frac{N_v}{N_u + N_v} d_{vz}$$

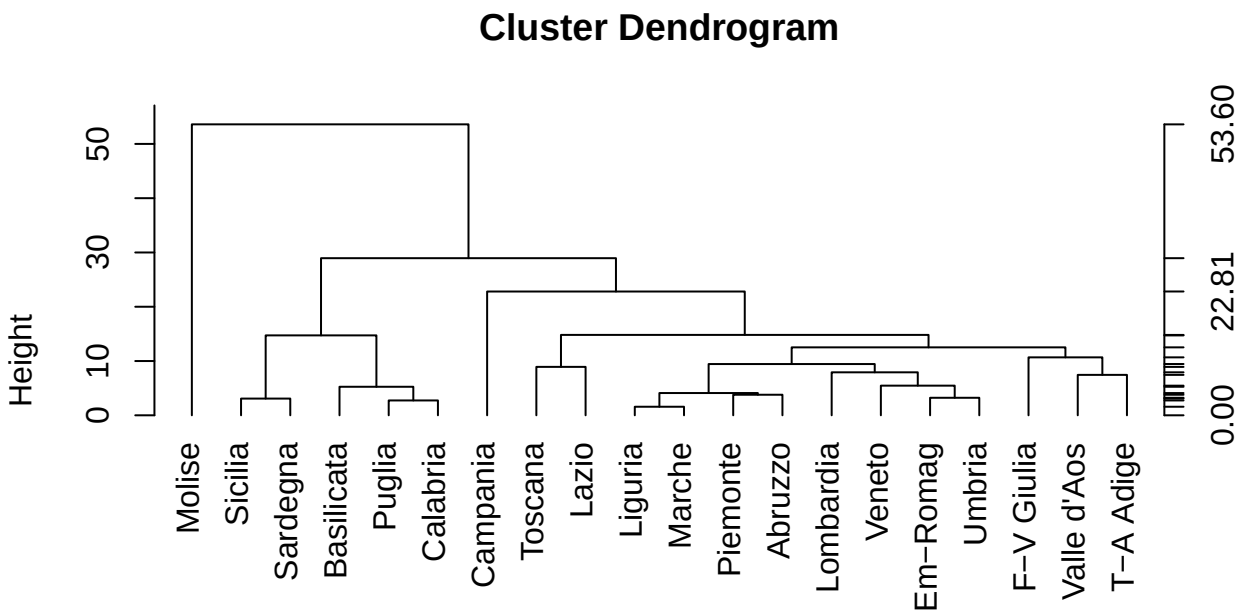
dove N_u, N_v e N_z sono rispettivamente il numero di individui del cluster G_u, G_v e G_z. La procedura si ripete fino a quando non si ottiene un unico cluster con tutti gli individui.

```
d <- dist(tabella, method = "euclidean",
          diag = TRUE, upper = FALSE)
hlm <- hclust(d, method = "average")
str(hlm)
```

```
## List of 7
##  $ merge      : int [1:19, 1:2] -3 -16 -19 -8 -1 1 -17 -6 -2 -4 ...
##  $ height     : num [1:19] 1.58 2.74 3.08 3.23 3.79 ...
##  $ order      : int [1:20] 14 19 20 17 16 18 15 9 12 3 ...
##  $ labels     : chr [1:20] "Piemonte" "Valle d'Aos" "Liguria" "Lombardia" ...
##  $ method     : chr "average"
##  $ call       : language hclust(d = d, method = "average")
##  $ dist.method: chr "euclidean"
##  - attr(*, "class")= chr "hclust"
```

Procediamo adesso alla stampa del dendrogramma:

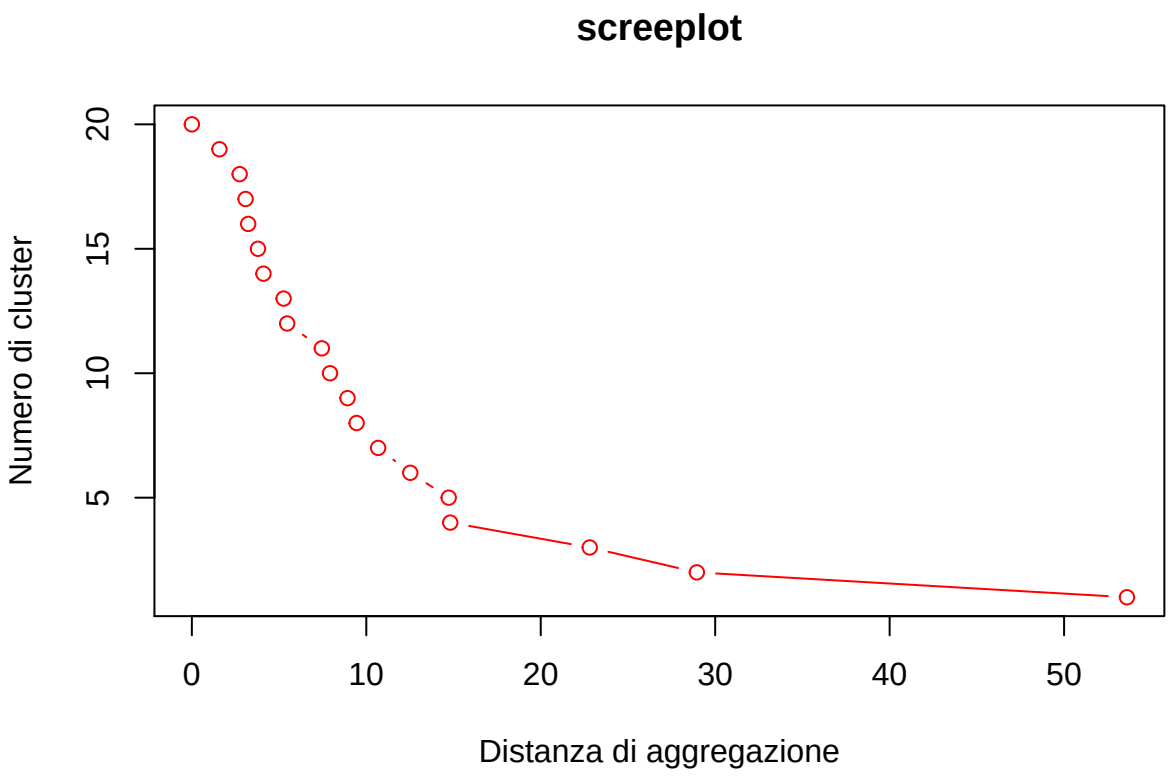
```
plot(hlm, hang = -1, xlab = "Metodo gerarchico agglomerativo",
     sub="del legame medio")
axis(side = 4, at= round(c(0, hlm$height), 2))
```



Metodo gerarchico agglomerativo
del legame medio

Come fatto in precedenza, procediamo alla stampa dello screeplot:

```
plot(c(0, hlm$height), seq(20, 1), type = "b",
     main = "screeplot", xlab = "Distanza di aggregazione",
     ylab = "Numero di cluster", col = "red")
```




```
hlm$height
```

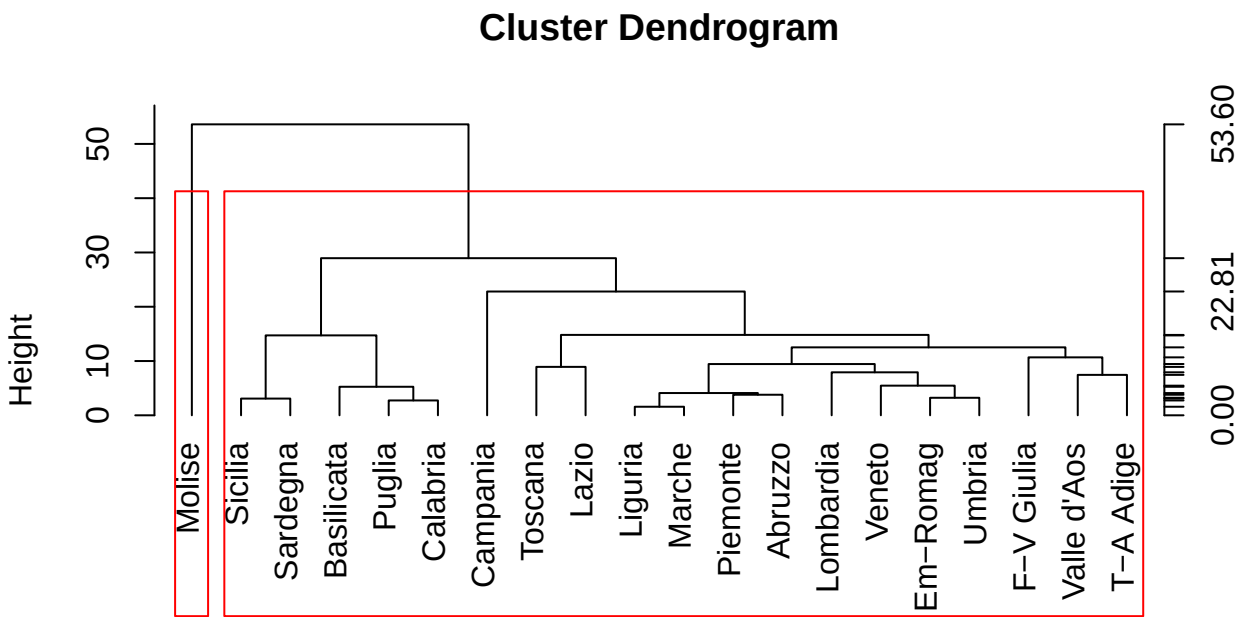
```
##  [1]  1.581139  2.740894  3.080844  3.229938  3.789670  4.104984  5.260917
##  [8]  5.464511  7.450503  7.920432  8.924887  9.445813 10.678466 12.520514
## [15] 14.724801 14.813469 22.808357 28.947554 53.603492
```

Anche in questo caso il max lo abbiamo fra:

$$\delta_2 = h_1 - h_2 = 53.603492 - 28.947554 = 24.65594$$

Quindi il valore di k per il quale δ_k è massimo si ha con $k = 2$. Applichiamo la suddivisione al dendrogramma utilizzando la funzione *rect.hclust()*:

```
plot(hlm, hang = -1, xlab = "Metodo gerarchico agglomerativo",
     sub="del legame medio")
axis(side = 4, at= round(c(0, hlm$height), 2))
rect.hclust(hlm, k=2)
```



Metodo gerarchico agglomerativo
del legame medio

Per ottenere la suddivisione in cluster oltre che graficamente, è possibile utilizzare la funzione *cutree()*.

```
cutree(hlm, k=2)
```

```
##    Piemonte Valle d'Aos     Liguria   Lombardia   T-A Adige      Veneto
##           1           1           1           1           1           1
##  F-V Giulia    Em-Romag     Toscana      Umbria      Marche       Lazio
##           1           1           1           1           1           1
##     Abruzzo      Molise    Campania      Puglia  Basilicata    Calabria
##           1           2           1           1           1           1
##     Sicilia    Sardegna
##           1           1
```

Procediamo adesso all'analisi utilizzando le misure di non omogeneità statistiche, in quanto, anche in precedenza sono risultate essere più precise rispetto allo screeplot. Faremo l'analisi per la suddivisione in 2,3 e 4 cluster.

Siccome la misura di non omogeneità statistica totale è uguale a quella calcolata in precedenza evitiamo di ricalcolarla.

Procediamo al calcolo delle misure di omogeneità statistiche per le varie suddivisioni:

```
d <- dist(tabella, method="euclidean",
          diag = TRUE, upper=TRUE)
d2<-d^2
tree <- hclust(d2, method = "average")
taglio <-cutree(tree, k=2, h=NULL)
num <- table(taglio)
tagliolist <- list(taglio)

agvar <- aggregate(tabella, tagliolist, var)[,-1]

trH1<-(num[[1]]-1) * sum(agvar[1,])
if(is.na(trH1))
  trH1 <- 0

trH2<-(num[[2]]-1) * sum(agvar[2,])
if(is.na(trH2))
  trH2 <- 0

trS <- trH1 + trH2

trB <- trHI-trH1-trH2
trB/trHI
```

```
## [1] 0.3793997
```

Con la suddivisione in due cluster, suggeritaci dallo scatterplot, il rapporto tra $\frac{trB}{trT}$ è risultato essere estremamente basso.

```
d <- dist(tabella, method="euclidean",
          diag = TRUE, upper=TRUE)
d2<-d^2
tree <- hclust(d2, method = "average")
taglio <-cutree(tree, k=3, h=NULL)
num <- table(taglio)
tagliolist <- list(taglio)

agvar <- aggregate(tabella, tagliolist, var)[,-1]

trH1<-(num[[1]]-1) * sum(agvar[1,])
if(is.na(trH1))
  trH1 <- 0

trH2<-(num[[2]]-1) * sum(agvar[2,])
if(is.na(trH2))
```

```
  trH2 <- 0

trH3<-(num[[3]]-1) * sum(agvar[3,])
if(is.na(trH3))
  trH3 <- 0

trS <- trH1 + trH2 + trH3

trB <- trHI-trH1-trH2-trH3
trB/trHI
```

```
## [1] 0.7718826
```

Con la suddivisione in tre cluster, suggeritaci durante l'analisi con i metodi non gerarchici, invece il rapporto tra $\frac{trB}{trT}$ ha raggiunto un risultato di sufficienza ed è possibile tenerlo in considerazione. Da notare che il valore è pressochè identico a quello trovato con il metodo k-means.

```
d <- dist(tabella, method="euclidean",
          diag = TRUE, upper=TRUE)
d2<-d^2
tree <- hclust(d2, method = "average")
taglio <-cutree(tree, k=4, h=NULL)
num <- table(taglio)
tagliolist <- list(taglio)

agvar <- aggregate(tabella, tagliolist, var)[,-1]

trH1<-(num[[1]]-1) * sum(agvar[1,])
if(is.na(trH1))
  trH1 <- 0

trH2<-(num[[2]]-1) * sum(agvar[2,])
if(is.na(trH2))
  trH2 <- 0

trH3<-(num[[3]]-1) * sum(agvar[3,])
if(is.na(trH3))
  trH3 <- 0

trH4<-(num[[4]]-1) * sum(agvar[4,])
if(is.na(trH4))
  trH4 <- 0
```

```r
trS <- trH1 + trH2 + trH3 + trH4

trB <- trHI-trH1-trH2-trH3-trH4
trB/trHI
```

```
## [1] 0.8529709
```

Con quest'ultima suddivisione, suggeritaci dal metodo k-means fatto in precedenza, il rapporto fra $\frac{trB}{trT}$ raggiunge il risultato più alto fra i precedenti ed è in linea con quello ottenuto con il metodo del legame completo e risulta essere un ottimo risultato dato il numero di cluster scelti.

La relativa suddivisione in cluster è la seguente:

```r
cutree(hlm, k=4, h=NULL)
```

```
##    Piemonte Valle d'Aos     Liguria   Lombardia   T-A Adige      Veneto
##           1           1           1           1           1           1
##  F-V Giulia    Em-Romag     Toscana      Umbria      Marche       Lazio
##           1           1           1           1           1           1
##     Abruzzo      Molise    Campania      Puglia  Basilicata    Calabria
##           1           2           3           4           4           4
##     Sicilia    Sardegna
##           4           4
```

C_1 = {Campania} C_2 = {Emilia Romagna, Umbria, Valle d'Aosta, Veneto, Lombardia, Trentino-alto Adige, Friuli-venezia Giulia, Liguria, Marche, Piemonte, Abruzzo, Lazio, Toscana} C_3 = {Molise} C_4 = {Sicilia, Sardegna, Basilicata, Puglia, Calabria}

Metodo del centroide

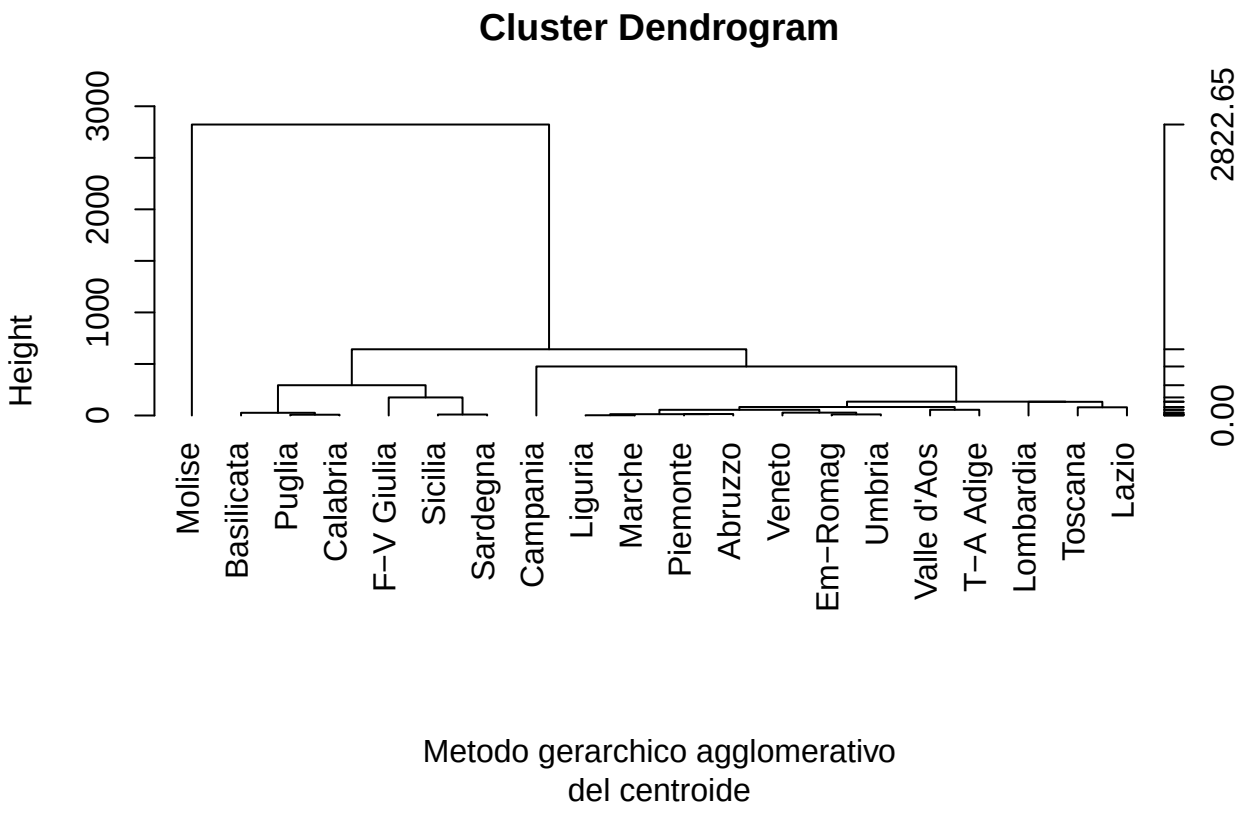
In questo metodo la distanza tra il gruppo G_1 e il gruppo G_2 è definita come la distanza tra i centroidi, ovvero le medie campionarie calcolate sugli individui dei gruppi. A differenza dei metodi precedenti qui viene considerata la matrice contenente i quadrati delle singole distanze euclidee.

```r
d <- dist(tabella, method = "euclidean",
          diag = TRUE, upper = FALSE)
d2 <- d^2
hlcentroid <- hclust(d2, method = "centroid")
str(hlcentroid)
```

```
## List of 7
##  $ merge      : int [1:19, 1:2] -3 -16 -19 -8 -1 1 -17 -6 6 -2 ...
##  $ height     : num [1:19] 2.5 7.51 9.49 10.43 14.36 ...
##  $ order      : int [1:20] 14 17 16 18 7 19 20 15 3 11 ...
##  $ labels     : chr [1:20] "Piemonte" "Valle d'Aos" "Liguria" "Lombardia" ...
##  $ method     : chr "centroid"
##  $ call       : language hclust(d = d2, method = "centroid")
##  $ dist.method: chr "euclidean"
##  - attr(*, "class")= chr "hclust"
```

Procediamo adesso alla stampa del dendrogramma:

```
plot(hlcentroid, hang = -1, xlab = "Metodo gerarchico agglomerativo",
     sub="del centroide")
axis(side = 4, at= round(c(0, hlcentroid$height), 2))
```



Procediamo adesso all'analisi utilizzando le misure di non omogeneità statistiche. Faremo l'analisi per la suddivisione in 2,3 e 4 cluster.

Siccome la misura di non omogeneità statistica totale è uguale a quella calcolata in precedenza evitiamo di ricalcolarla.

Procediamo al calcolo delle misure di omogeneità statistiche per le varie suddivisioni:

```
d <- dist(tabella, method="euclidean",
          diag = TRUE, upper=TRUE)
d2<-d^2
tree <- hclust(d2, method = "centroid")
taglio <-cutree(tree, k=2, h=NULL)
num <- table(taglio)
tagliolist <- list(taglio)

agvar <- aggregate(tabella, tagliolist, var)[,-1]

trH1<-(num[[1]]-1) * sum(agvar[1,])
if(is.na(trH1))
  trH1 <- 0

trH2<-(num[[2]]-1) * sum(agvar[2,])
if(is.na(trH2))
  trH2 <- 0

trS <- trH1 + trH2

trB <- trHI-trH1-trH2
trB/trHI
```

```
## [1] 0.3793997
```

Con la suddivisione in due cluster, suggeritaci dallo scatterplot, il rapporto tra $\frac{trB}{trT}$ è risultato essere estremamente basso.

```
d <- dist(tabella, method="euclidean",
          diag = TRUE, upper=TRUE)
d2<-d^2
tree <- hclust(d2, method = "centroid")
taglio <-cutree(tree, k=3, h=NULL)
num <- table(taglio)
tagliolist <- list(taglio)

agvar <- aggregate(tabella, tagliolist, var)[,-1]

trH1<-(num[[1]]-1) * sum(agvar[1,])
if(is.na(trH1))
  trH1 <- 0

trH2<-(num[[2]]-1) * sum(agvar[2,])
if(is.na(trH2))
```

```
  trH2 <- 0

trH3<-(num[[3]]-1) * sum(agvar[3,])
if(is.na(trH3))
  trH3 <- 0

trS <- trH1 + trH2 + trH3

trB <- trHI-trH1-trH2-trH3
trB/trHI
```

```
## [1] 0.7525784
```

Con la suddivisione in tre cluster, suggeritaci durante l'analisi con i metodi non gerarchici, invece il rapporto tra $\frac{trB}{trT}$ ha raggiunto un risultato di sufficienza ed è possibile tenerlo in considerazione.

```
d <- dist(tabella, method="euclidean",
          diag = TRUE, upper=TRUE)
d2<-d^2
tree <- hclust(d2, method = "centroid")
taglio <-cutree(tree, k=4, h=NULL)
num <- table(taglio)
tagliolist <- list(taglio)

agvar <- aggregate(tabella, tagliolist, var)[,-1]

trH1<-(num[[1]]-1) * sum(agvar[1,])
if(is.na(trH1))
  trH1 <- 0

trH2<-(num[[2]]-1) * sum(agvar[2,])
if(is.na(trH2))
  trH2 <- 0

trH3<-(num[[3]]-1) * sum(agvar[3,])
if(is.na(trH3))
  trH3 <- 0

trH4<-(num[[4]]-1) * sum(agvar[4,])
if(is.na(trH4))
  trH4 <- 0

trS <- trH1 + trH2 + trH3 + trH4
```

```
trB <- trHI-trH1-trH2-trH3-trH4
trB/trHI
```

```
## [1] 0.8146854
```

Con quest'ultima suddivisione, il rapporto fra $\frac{trB}{trT}$ raggiunge il risultato più alto fra i precedenti ed è poco inferiore a quello ottenuto con i due metodi precedenti e risulta essere un ottimo risultato dato il numero di cluster scelti.

La relativa suddivisione in cluster è la seguente:

```
cutree(hlm, k=4, h=NULL)
```

```
##    Piemonte Valle d'Aos     Liguria   Lombardia   T-A Adige      Veneto
##           1           1           1           1           1           1
##  F-V Giulia    Em-Romag     Toscana      Umbria      Marche       Lazio
##           1           1           1           1           1           1
##     Abruzzo      Molise    Campania      Puglia  Basilicata    Calabria
##           1           2           3           4           4           4
##     Sicilia    Sardegna
##           4           4
```

C_1 = {Emilia Romagna, Umbria, Valle d'Aosta, Veneto, Lombardia, Trentino-alto Adige, Friuli-venezia Giulia, Liguria, Marche, Piemonte, Abruzzo, Lazio, Toscana} C_2 = {Molise} C_3 = {Campania} C_4 = {Sicilia, Sardegna, Basilicata, Puglia, Calabria}

Metodo della mediana

Questo metodo è simile a quello del centroide e differisce per il fatto che la procedura è indipendente dalla numerosità dei cluster. Infatti, quando due gruppi si aggregano il nuovo centroide è calcolato come la semisomma dei due centroidi precedenti. Anche il metodo della mediana, come quello del legame singolo può dare vita alla formazione di una catena fra gli individui.

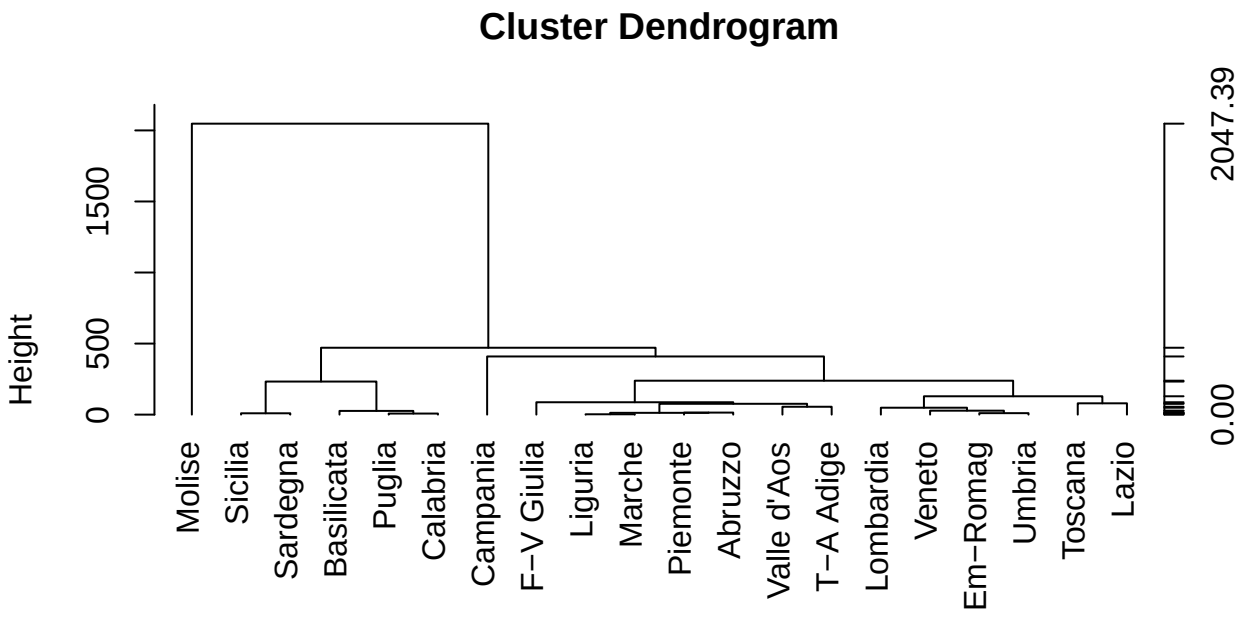
```
d <- dist(tabella, method = "euclidean",
          diag = TRUE, upper = FALSE)
d2 <- d^2
hlmedian <- hclust(d2, method = "median")
str(hlmedian)
```

```
## List of 7
##  $ merge      : int [1:19, 1:2] -3 -16 -19 -8 -1 1 -17 -6 -4 -2 ...
```

```
##  $ height     : num [1:19] 2.5 7.51 9.49 10.43 14.36 ...
##  $ order      : int [1:20] 14 19 20 17 16 18 15 7 3 11 ...
##  $ labels     : chr [1:20] "Piemonte" "Valle d'Aos" "Liguria" "Lombardia" ...
##  $ method     : chr "median"
##  $ call       : language hclust(d = d2, method = "median")
##  $ dist.method: chr "euclidean"
##  - attr(*, "class")= chr "hclust"
```

Procediamo adesso alla stampa del dendrogramma:

```
plot(hlmedian, hang = -1, xlab = "Metodo gerarchico agglomerativo",
     sub="della mediana")
axis(side = 4, at= round(c(0, hlmedian$height), 2))
```



Procediamo adesso all'analisi utilizzando le misure di non omogeneità statistiche. Faremo l'analisi per la suddivisione in 2,3 e 4 cluster.

Siccome la misura di non omogeneità statistica totale è uguale a quella calcolata in precedenza evitiamo di ricalcolarla.

Procediamo al calcolo delle misure di omogeneità statistiche per le varie suddivisioni:

```
d <- dist(tabella, method="euclidean",
          diag = TRUE, upper=TRUE)
d2<-d^2
tree <- hclust(d2, method = "median")
taglio <-cutree(tree, k=2, h=NULL)
num <- table(taglio)
tagliolist <- list(taglio)

agvar <- aggregate(tabella, tagliolist, var)[,-1]

trH1<-(num[[1]]-1) * sum(agvar[1,])
if(is.na(trH1))
  trH1 <- 0

trH2<-(num[[2]]-1) * sum(agvar[2,])
if(is.na(trH2))
  trH2 <- 0

trS <- trH1 + trH2

trB <- trHI-trH1-trH2
trB/trHI
```

```
## [1] 0.3793997
```

Con la suddivisione in due cluster, suggeritaci dallo scatterplot, il rapporto tra $\frac{trB}{trT}$ è risultato essere estremamente basso.

```
d <- dist(tabella, method="euclidean",
          diag = TRUE, upper=TRUE)
d2<-d^2
tree <- hclust(d2, method = "median")
taglio <-cutree(tree, k=3, h=NULL)
num <- table(taglio)
tagliolist <- list(taglio)

agvar <- aggregate(tabella, tagliolist, var)[,-1]

trH1<-(num[[1]]-1) * sum(agvar[1,])
if(is.na(trH1))
  trH1 <- 0

trH2<-(num[[2]]-1) * sum(agvar[2,])
if(is.na(trH2))
```

```
  trH2 <- 0

trH3<-(num[[3]]-1) * sum(agvar[3,])
if(is.na(trH3))
  trH3 <- 0

trS <- trH1 + trH2 + trH3

trB <- trHI-trH1-trH2-trH3
trB/trHI
```

```
## [1] 0.7718826
```

Con la suddivisione in tre cluster, suggeritaci durante l'analisi con i metodi non gerarchici, invece il rapporto tra $\frac{trB}{trT}$ ha raggiunto un risultato di sufficienza ed è possibile tenerlo in considerazione.

```
d <- dist(tabella, method="euclidean",
          diag = TRUE, upper=TRUE)
d2<-d^2
tree <- hclust(d2, method = "median")
taglio <-cutree(tree, k=4, h=NULL)
num <- table(taglio)
tagliolist <- list(taglio)

agvar <- aggregate(tabella, tagliolist, var)[,-1]

trH1<-(num[[1]]-1) * sum(agvar[1,])
if(is.na(trH1))
  trH1 <- 0

trH2<-(num[[2]]-1) * sum(agvar[2,])
if(is.na(trH2))
  trH2 <- 0

trH3<-(num[[3]]-1) * sum(agvar[3,])
if(is.na(trH3))
  trH3 <- 0

trH4<-(num[[4]]-1) * sum(agvar[4,])
if(is.na(trH4))
  trH4 <- 0

trS <- trH1 + trH2 + trH3 + trH4
```

```
trB <- trHI-trH1-trH2-trH3-trH4
trB/trHI
```

```
## [1] 0.833741
```

Con quest'ultima suddivisione, il rapporto fra $\frac{trB}{trT}$ raggiunge il risultato più alto fra i precedenti ed è poco inferiore a quello ottenuto con i due metodi precedenti e risulta essere un ottimo risultato dato il numero di cluster scelti.

La relativa suddivisione in cluster è la seguente:

```
cutree(hlm, k=4, h=NULL)
```

```
##    Piemonte Valle d'Aos     Liguria   Lombardia   T-A Adige      Veneto
##           1           1           1           1           1           1
##  F-V Giulia    Em-Romag     Toscana      Umbria      Marche       Lazio
##           1           1           1           1           1           1
##     Abruzzo      Molise    Campania      Puglia  Basilicata    Calabria
##           1           2           3           4           4           4
##     Sicilia    Sardegna
##           4           4
```

C_1 = *{Emilia Romagna, Umbria, Valle d'Aosta, Veneto, Lombardia, Trentino-alto Adige, Friuli-venezia Giulia, Liguria, Marche, Piemonte, Abruzzo, Lazio, Toscana}* C_2 = *{Molise}* C_3 = *{Campania}* C_4 = *{Sicilia, Sardegna, Basilicata, Puglia, Calabria}*