# Laboratorio di Statistica e Analisi Dati: Lezione 5

Tommaso C. & Marco G.

23 - 25 Novembre 2016

# News

- Faq su come interagire con le slide .html
- Aggiunto link alla pagina principale IDE
- Corretto il calendario delle lezioni (rimossa la lezione del 7 - 9 dicembre)

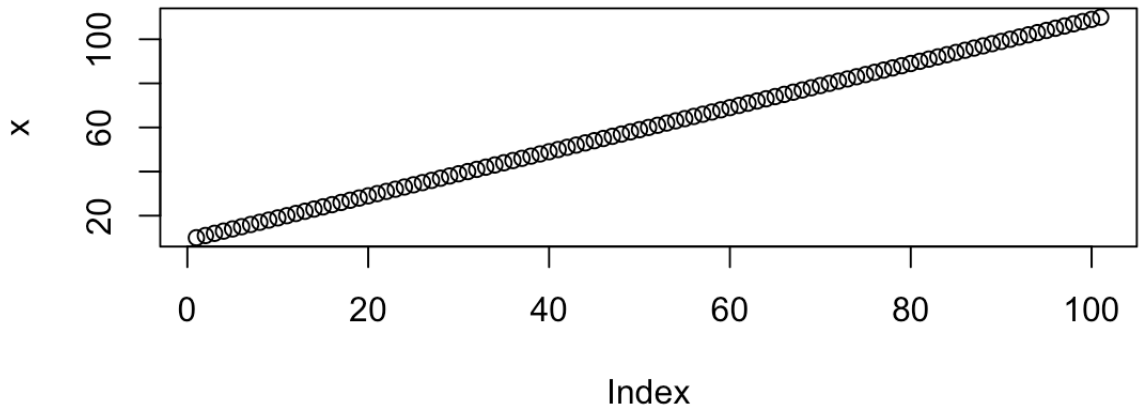
# Grafica in R

- Permette di generare facilmente grafici che:
  - *possono essere visualizzati in apposite finestre grafiche (nei sistemi dotati di interfaccia grafica)*
  - *possono essere salvati direttamente su file in diversi formati*
- Come avete visto durante le lezioni di teoria, la visualizzazione grafica permette di otterere subito una valutazione qualitativa dei vostri dati

# Comando plot: dati

Il comando che permette la generazione di un grafico è `plot`. Il tipo di grafico generato dipende dal tipo di argomento passato e dai parametri inseriti

```
x = 10:110
plot(x)
```

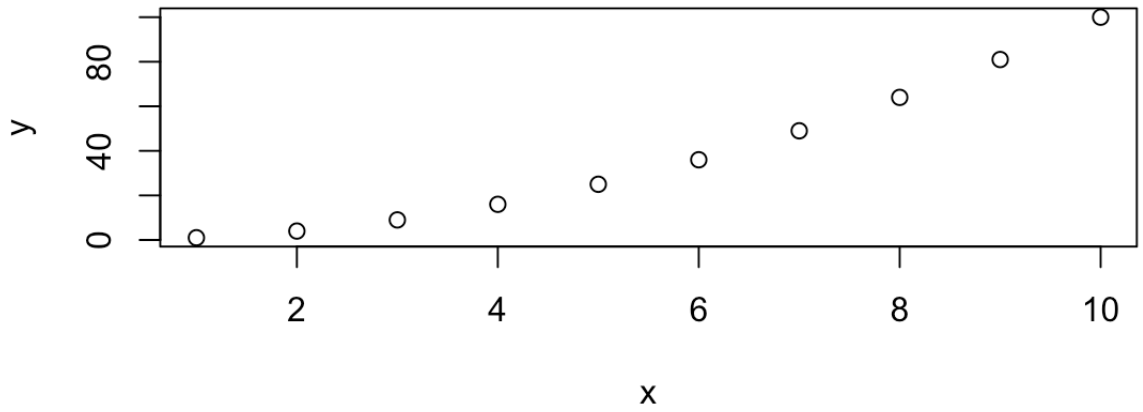



se x è un vettore produce un grafico dei valori del vettore rispetto ad un vettore di indici `y <- 1: length(x)`

# Comando plot: dati (cont.)

Il comando che permette la generazione di un grafico è `plot`. Il tipo di grafico generato dipende dal tipo di argomento passato e dai parametri inseriti

```
x = 1:10
y = x^2
plot(x,y)
```

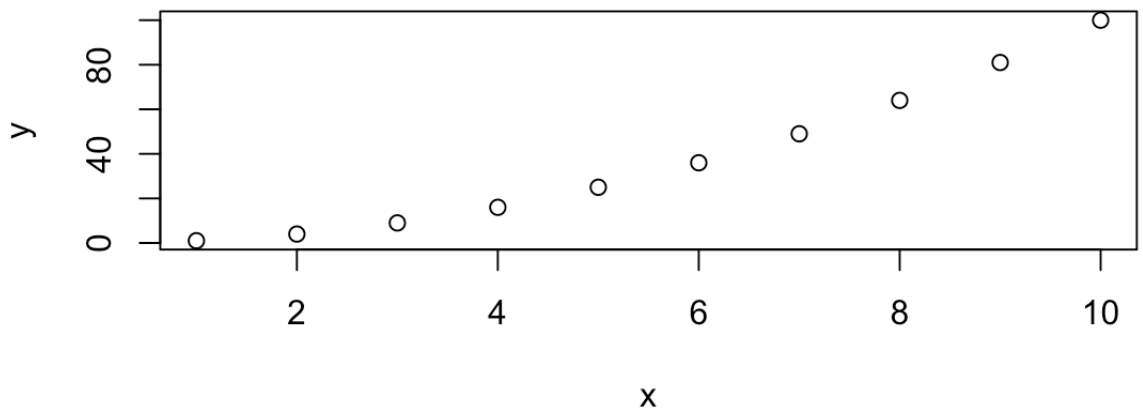



se x e y sono vettori produce un grafico delle coppie $(x_i, y_i)$

# Comando plot: dati (cont.)

Il comando che permette la generazione di un grafico è `plot`. Il tipo di grafico generato dipende dal tipo di argomento passato e dai parametri inseriti

```
plot(m)
```

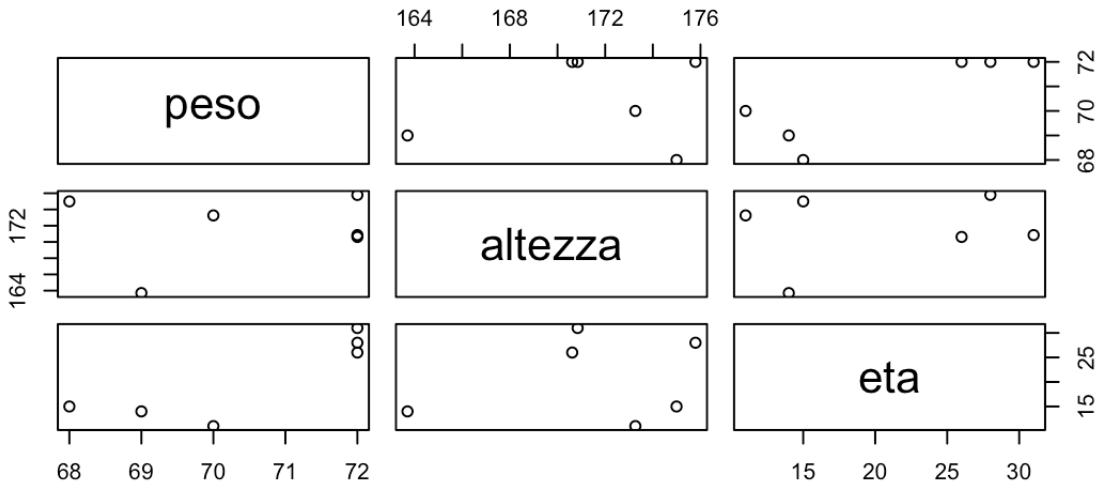



se m è una matrice produce un grafico delle coppie $(m_{i,1}, m_{i,2})$

# Comando plot: dati (cont.)

Il comando che permette la generazione di un grafico è `plot`. Il tipo di grafico generato dipende dal tipo di argomento passato e dai parametri inseriti

```
plot(df)
```



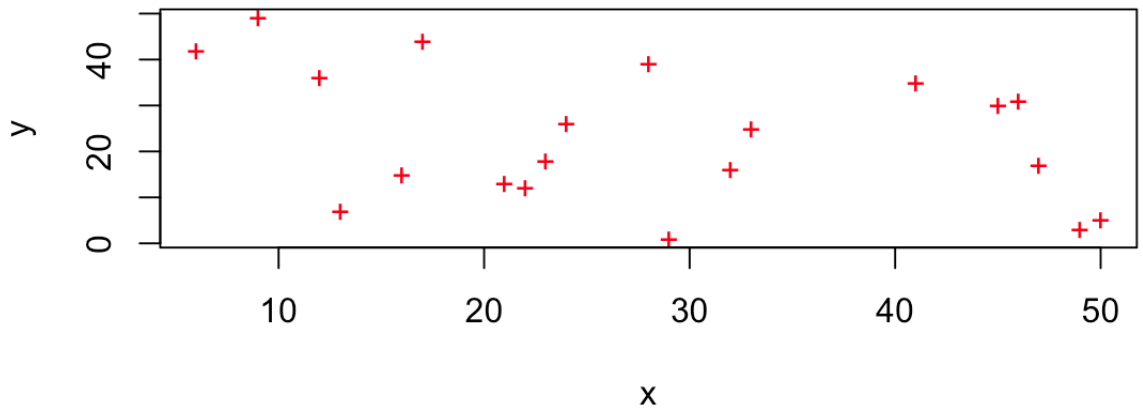

se df è un dataframe, produce i grafici delle distribuzioni delle variabili contenute nel data frame

N.B. Viene effettuato una prodotto cartesiano di tutte le colonne del dataframe
$\rightarrow$ che ad esempio con solo 6 colonne ottengo $6^2 - 6 = 30$ grafici

# Comando plot: altri argomenti (cont.)

Il grafico può essere personalizzato in più modi (per tutte le opzioni digitate ? plot.default).
I più comuni riguadano il tipo di indicatore dei valori `pch` e il loro colore `col`

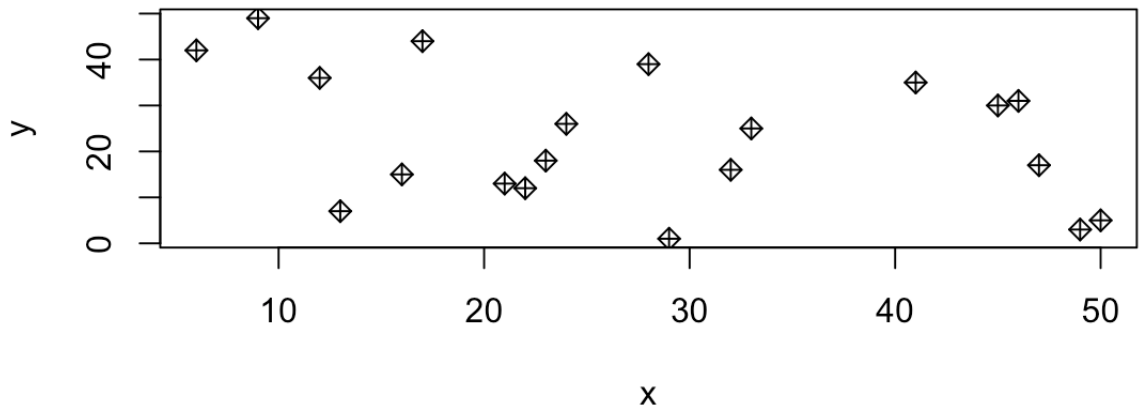
```
plot(x, y, col = 2, pch = "+")
```



# Comando plot: altri argomenti (cont.)

Il grafico può essere personalizzato in più modi (per tutte le opzioni digitate ? plot.default).
I più comuni riguadano il tipo di indicatore dei valori `pch` e il loro colore `col`

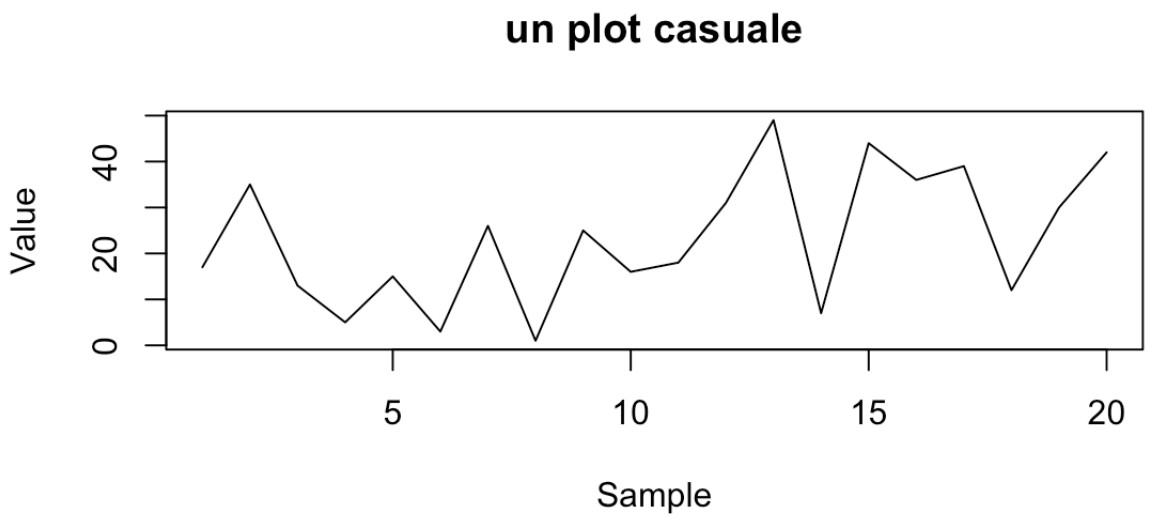
```
plot(x, y, col = "#000000", pch = 9) # con colori hex
```



# Comando plot: altri argomenti (cont.)

All'interno del grafico è possibile impostare le etichette per gli assi (`xlab, ylab`), il titolo (`main`) e il tipo (`type`, se linee o punti)

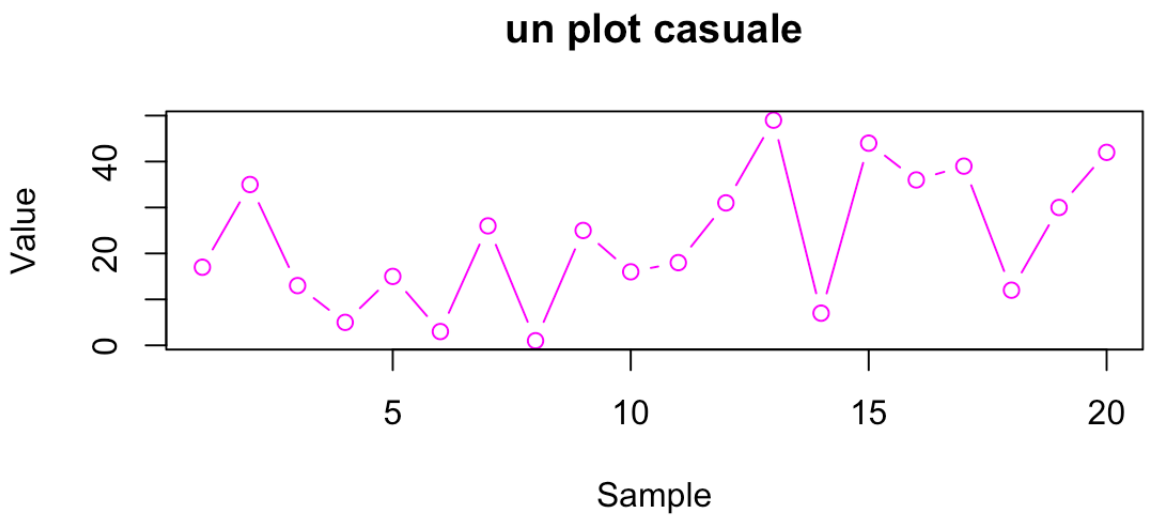
```
plot(x, y, xlab = "Sample", ylab = "Value", type = "l", main = "un plot casuale")
```



# Comando plot: altri argomenti (cont.)

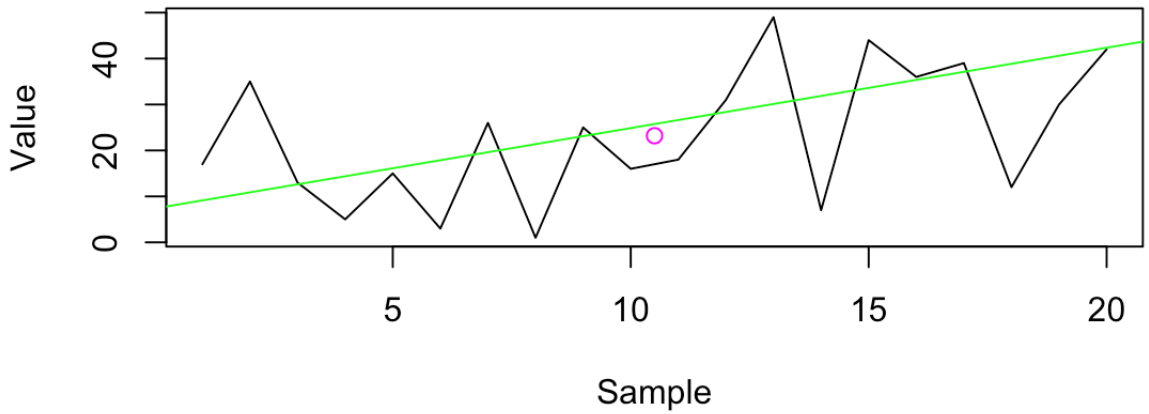
All'interno del grafico è possibile impostare le etichette per gli assi (`xlab, ylab`), il titolo (`main`) e il tipo (`type`)

```
plot(x, y, xlab = "Sample", ylab = "Value", type = "b", col = "#FF00FF", main = "un plot casuale")
```



# Aggiungere linee e punti

```
plot(sort(x),y, xlab = "Sample",ylab = "Value", type = "l")
abline(coef(line(x,y)), col = "#00FF00") # aggiunge una retta
points(mean(x), mean(y), col= "#FF00FF" ) # aggiunge un punto
```

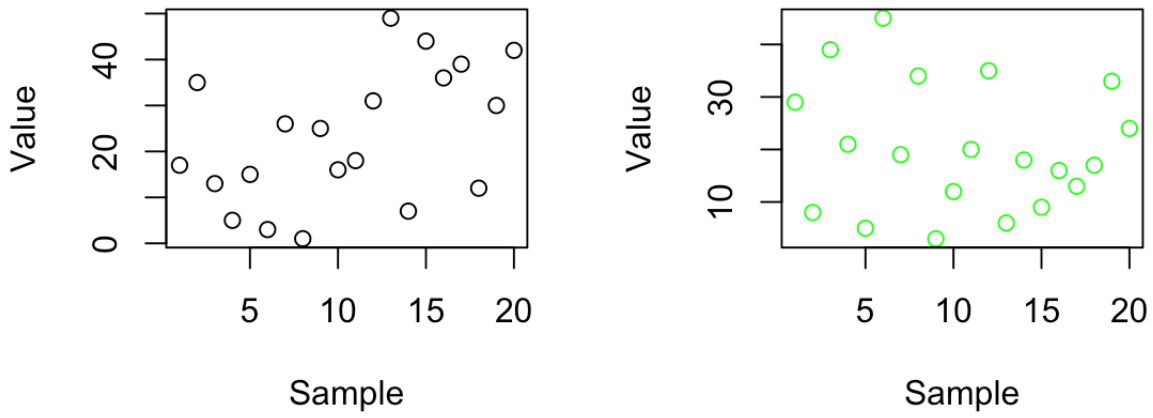



- line(x,y)????
- coeff()????
- abline()????

# Aggiungere linee e punti

R permette di aggiungere altri dati anche tramite il comando `par` (tutorial per gli interessati)

```
par(mfrow=c(1,2))
plot(x, y, xlab = "Sample", ylab = "Value")
plot(x, y1, xlab = "Sample", ylab = "Value", col = "#00FF00")
```




Se volete che il secondo plot venga inserito all'interno del primo inserite la riga

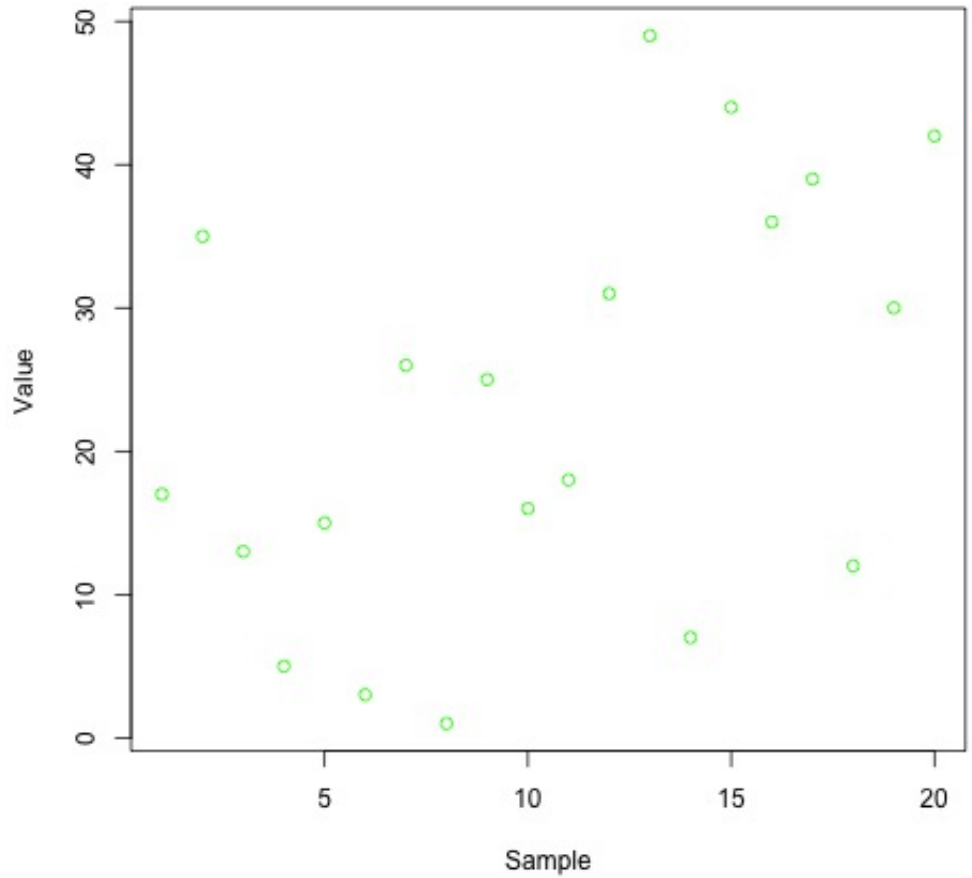
`par(new = TRUE)` tra le due righe contenenti il primo e il secondo plot

# Salvare una figura

- Aprire un “device” grafico specificando il path del file che vorrete salvare
- usare la funzione `plot()`
- chiudere il “device” grafico

```
jpeg("mioPlot.jpg")
plot(x, y, col = "#00FF00", xlab = "Sample", ylab = "Value")
dev.off()
## quartz_off_screen
##                 2
```

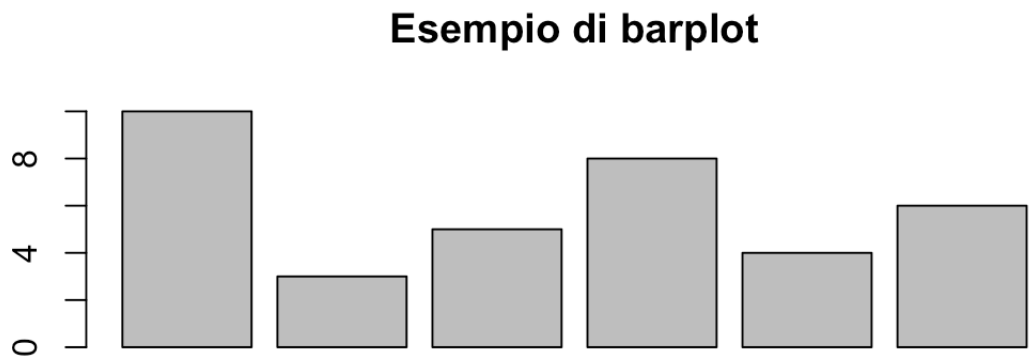
# Salvare una figura (cont.)



# Barplot

Genera un grafico a barre partendo dai valori del vettore `x`

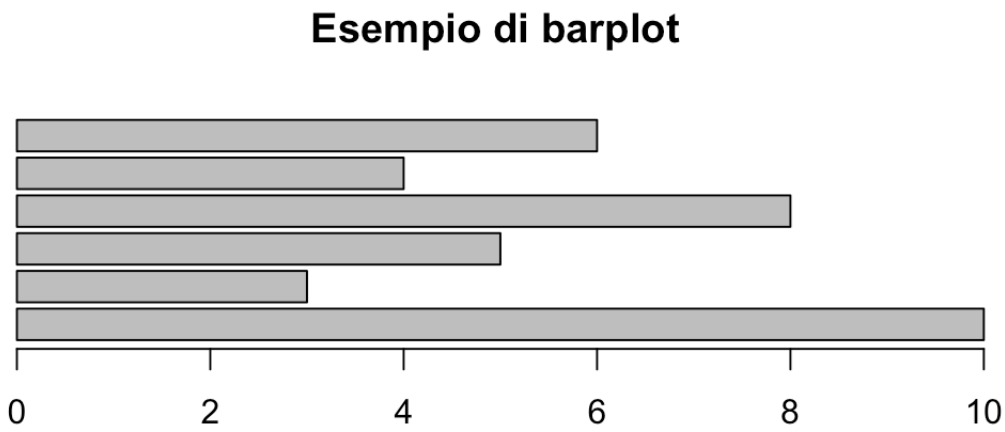
```r
dati <- c(10, 3, 5, 8, 4, 6)
barplot(dati, main="Esempio di barplot")
```



# Barplot (cont.)

Genera un grafico a barre partendo dai valori del vettore `x`

```
barplot(dati, main="Esempio di barplot", horiz = TRUE)
```

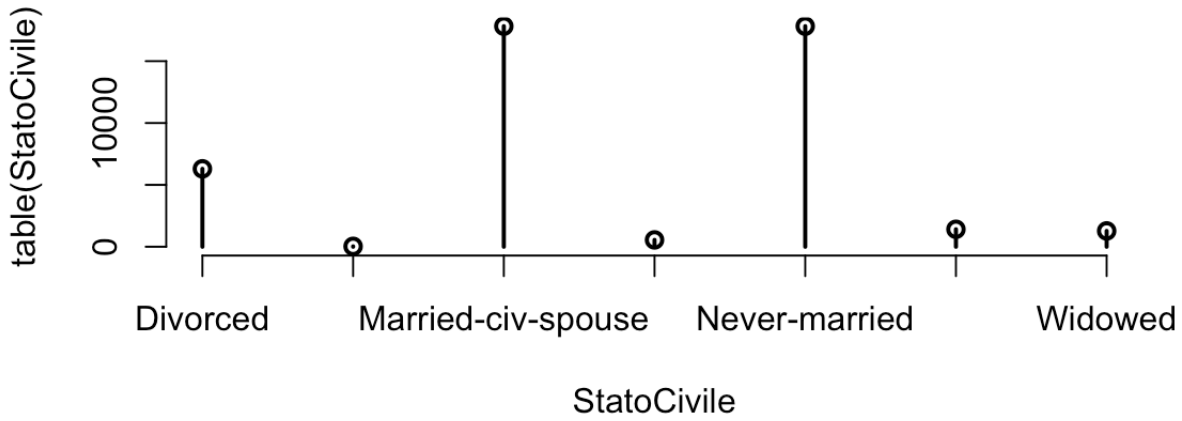



Ci sono parametri che permettono di modificare a piacere il grafico

# Diagrammi a bastoncini

Molto comodo per rappresentare le frequenze relative/assolute di un vettore. Si creano inserendo i punti nel grafico e successivamente aggiungendo le linee (`lines()`)

```
reddito <- read.csv("soluzioni/dataset/reddito.csv")
attach(reddito)
plot(table(StatoCivile), type = "p")
lines(table(StatoCivile))
```

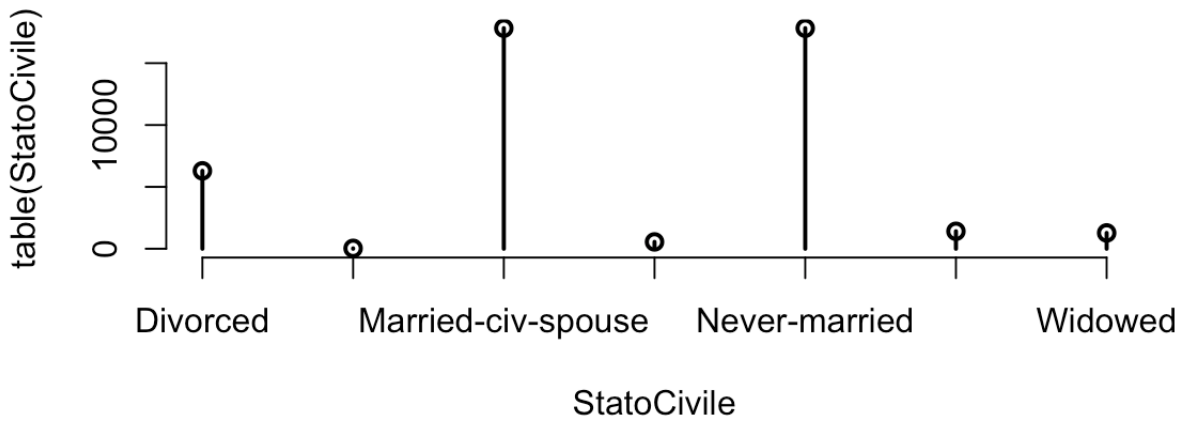


# Cosa fa il comando `attach()`?

# Diagrammi a bastoncini

Si creano inserendo i punti nel grafico e successivamente aggiungendo le linee (`lines()`)

```r
plot(table(StatoCivile), type = "p")
lines(table(StatoCivile))
```



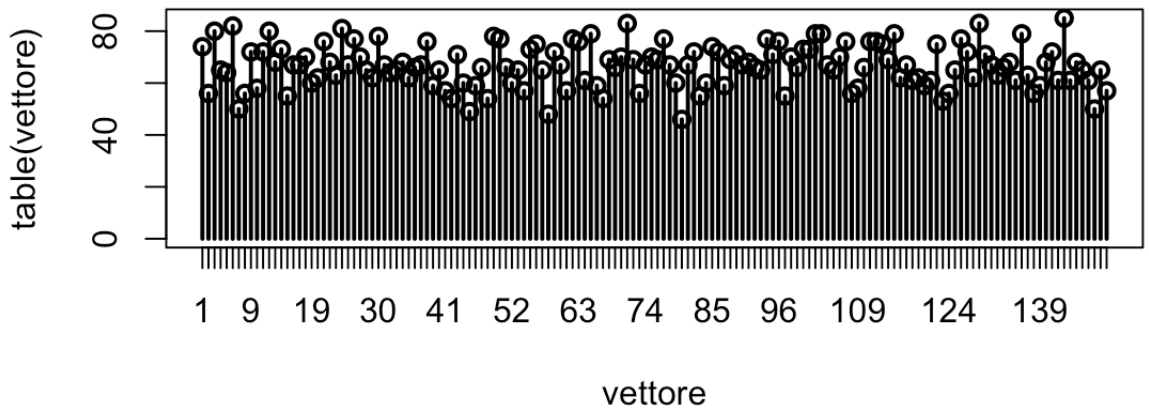

`attach()` serve per fissare il dataframe, in questo modo è possibile accedere alle componenti senza usare "`$`". Ricordatevi di usare `detach()` quando finite

di usare il dataframe

# Diagrammi a bastoncini (MAI E POI MAI)

Usare un diagramma a bastoncini per rappresentare le frequenze assolute/relative di valori reali (come l'età, il peso, l'altezza)

```
vettore <-sample (1:150, 10000,replace = TRUE)
plot(table(vettore), type = "p")
lines(table(vettore))
```



# Diagrammi ramo-foglia

- Sono utili quando si hanno pochi dati, perché permettono di riassumere rapidamente i dati mantenendo però tutte le informazioni di partenza
- Separano la parte intera di un numero da quella decimale
- Il numero di foglie su un ramo corrisponde alla frequenza assoluta dell'intervallo dal quel ramo al ramo successivo

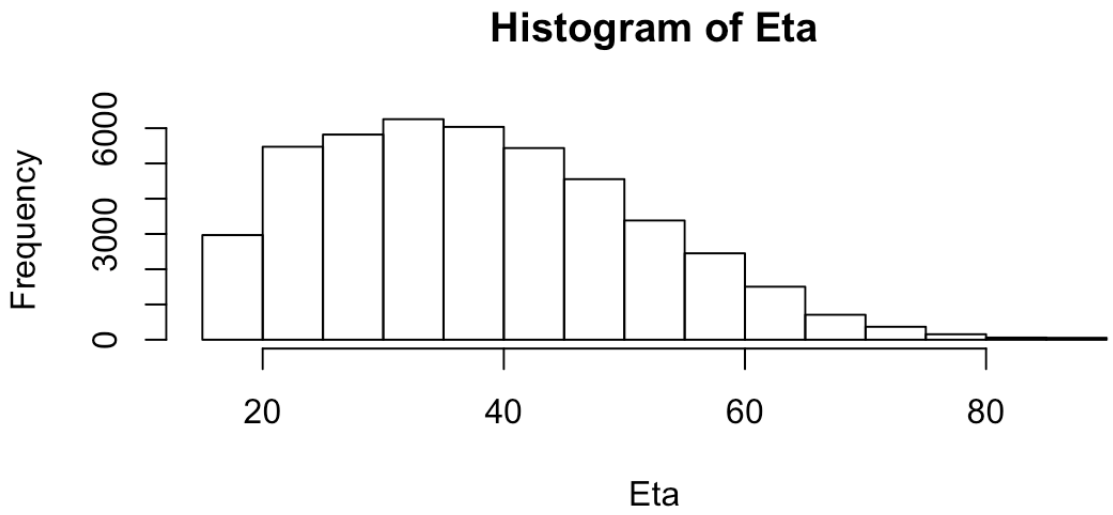
```
vettore <- c(0.1, 2.5, 3.7, 3.7, 0.7, 3.7, 1.5, 1.7, 0.4, 3.6, 0.2)
stem(vettore)
##
##   The decimal point is at the |
##
##   0 | 1247
##   1 | 57
##   2 | 5
##   3 | 6777
```

# Istogrammi

Il comando `hist()` genera istogrammi a partire da un vettore numerico

- `hist(x)` genera un'istogramma utilizzando il vettore numerico `x`

```
hist(Eta)
```

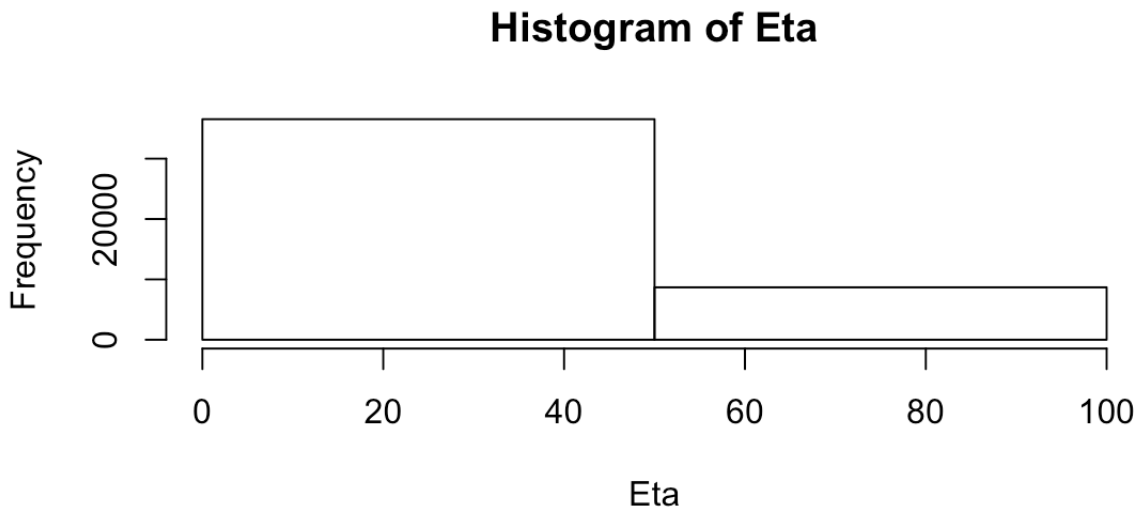


# Istogrammi

Il comando `hist()` genera istogrammi a partire da un vettore numerico

- `hist(x, nclass = n)` genera un'istogramma con un numero `n` di classi

```
hist(Eta, nclass = 2)
```

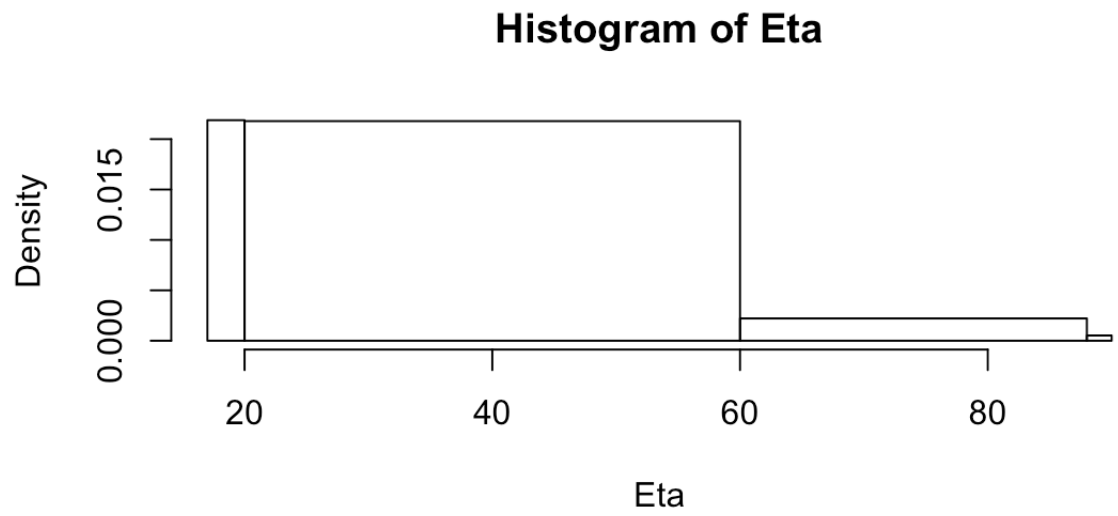


# Istogrammi

Il comando `hist()` genera istogrammi a partire da un vettore numerico

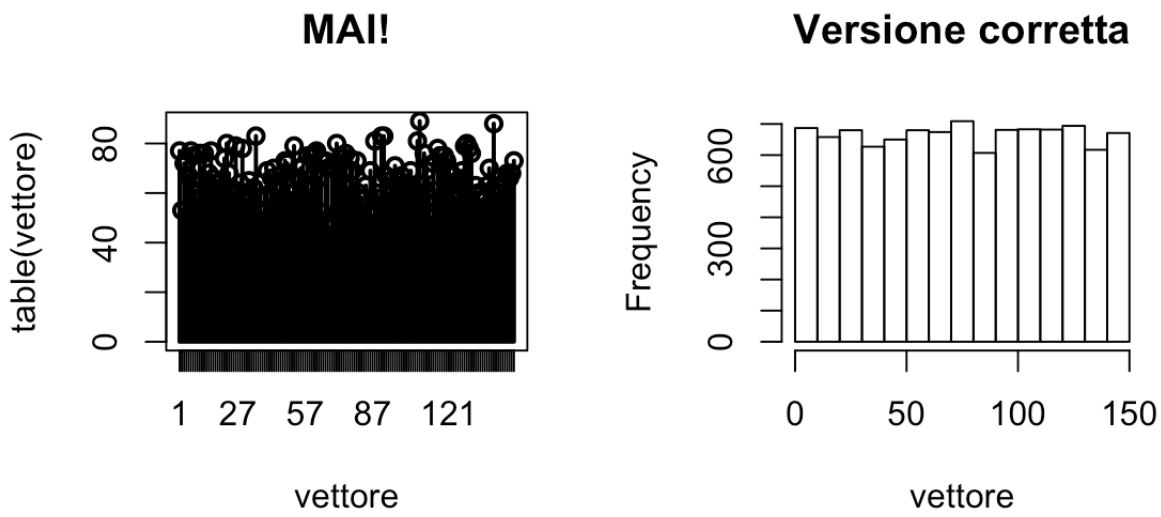
- `hist(x, breaks = v)` genera un istogramma usando i valori del vettore `v` come demilitatori

```r
hist(Eta, breaks = c(min(Eta),20,60,88, max(Eta)))
```



# Diagrammi a bastoncini vs. Istogrammi

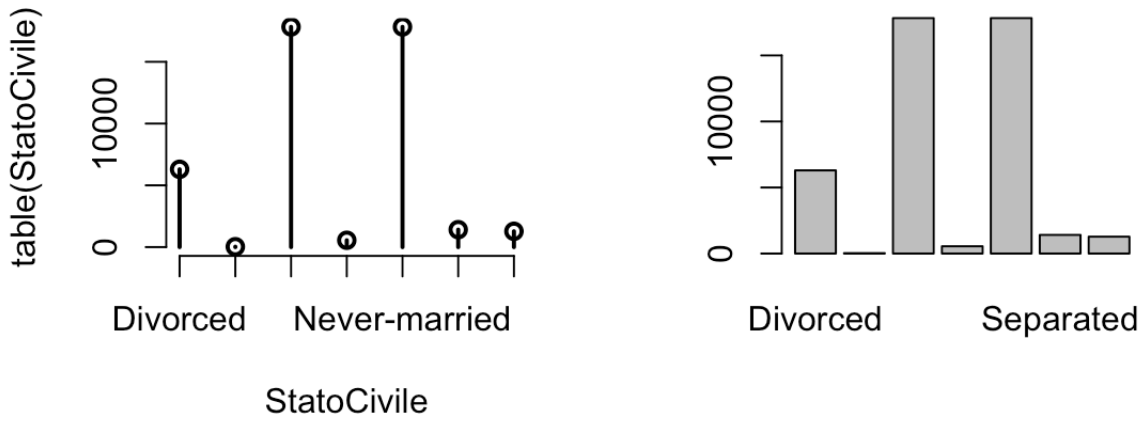
```
par(mfrow=c(1,2))
vettore <-sample (1:150, 10000, replace = TRUE)
plot(table(vettore), type = "p", main = "MAI!")
lines(table(vettore))
hist(vettore, main = "Versione corretta")
```



# Diagrammi a bastoncini vs. Istogrammi

Nel caso di fattori non è possibile usare il comando `hist()`, una soluzione è fare un barplot della tabella delle frequenze

```
par(mfrow=c(1,2))
plot(table(StatoCivile), type = "p")
lines(table(StatoCivile))
barplot(table(StatoCivile))
```



# Diagramma di Pareto (senza libreria qcc)

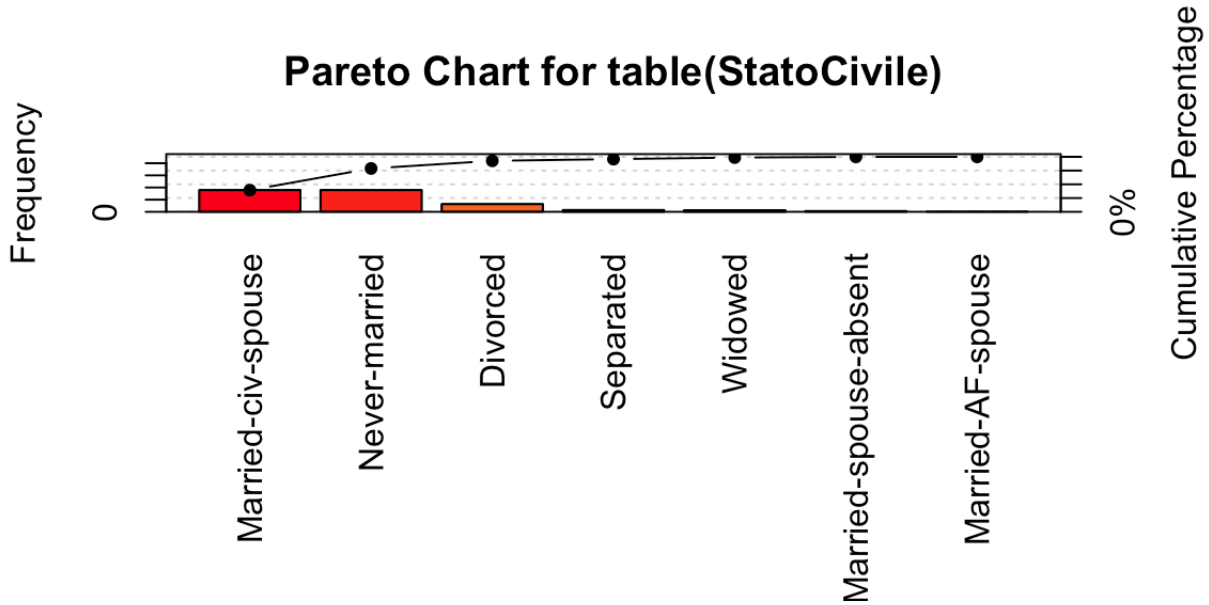
Mostra insieme frequenze relative e rispettivi valori cumulati.

```r
statoCivileRel <- sort(prop.table(table(StatoCivile)), decreasing = TRUE)
statoCivileCum <- cumsum(statoCivileRel)
# ylim mi crea un asse y da 0 a 1.3 (per avere più margine)
barplot(statoCivileRel, ylim = c(0,1.3), space = 0)
lines(statoCivileCum, type = "b")
```

# Diagramma di Pareto (con libreria qcc)

Installare la libreria tramite `install.packages("qcc")`, importare con `library(qcc)` e lanciare il comando `pareto.chart()`. Questo comando, oltre a visualizzare il grafico, scrive sulla consolle delle statistiche dei dati che state analizzando.

```
pareto.chart(table(StatoCivile))
```

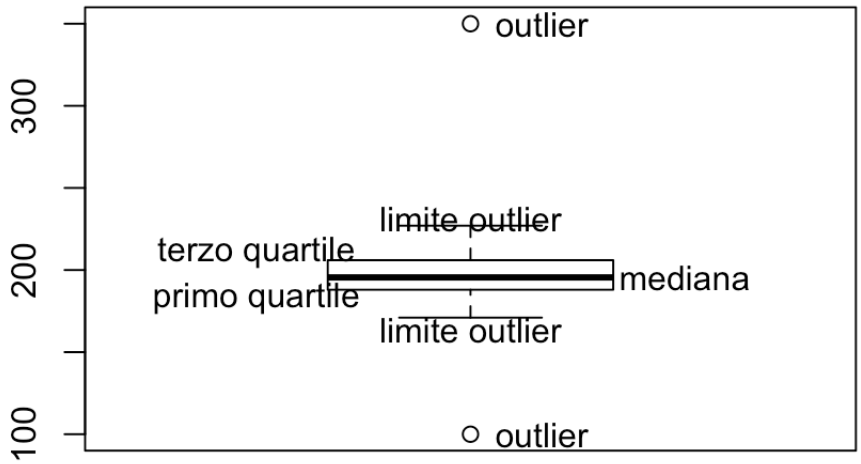



```
##
```

```
## Pareto chart analysis for table(StatoCivile)
##                          Frequency Cum.Freq.  Percentage Cum.Percent.
##   Married-civ-spouse         17826     17826 39.41886692     39.41887
##   Never-married              17826     35652 39.41886692     78.83773
##   Divorced                    6297     41949 13.92463845     92.76237
##   Separated                   1411     43360  3.12016275     95.88254
##   Widowed                     1278     44638  2.82605811     98.70859
##   Married-spouse-absent        552     45190  1.22064482     99.92924
##   Married-AF-spouse             32     45222  0.07076202    100.00000
```

# BoxPlot

Fornisce in maniera chiara una sintesi dei vostri dati **numerici**.



# Outlier e range interquartile

Gli **outlier** sono valori distanti dai valori osservati, nel caso di esperimenti reali possono ed esempio essere dovuti ad errori di misurazione.

Uno dei modi più comuni per calcore gli outlier è usare lo **scarto interquartile** (`IQR()`), ovvero la differenza tra il terzo $Q_3$ e il primo quartile $Q_1$

Un outlier $x$ è un valore tale per cui

$$x < Q_1 - 1.5 \cdot IQR \quad \vee \quad x > Q_3 + 1.5 \cdot IQR$$

```
iqr <- IQR(vettore)
iqr #scarto interquantile
## [1] 17
q1 <- quantile(vettore,0.25)
q1 - (iqr * 1.5) #i valori minori di questo sono outlier
## [1] 163.25
q3 <- quantile(vettore,0.75)
q3 + (iqr * 1.5) #i valori maggiori di questo sono outlier
## [1] 231.25
```

fonti

# Outlier e range interquartile

I **baffi** $(x_{min}, x_{max})$ del nostro boxplot sono i valori del vettore $v$ tali che:

- $x_{min} = min\{x|x > Q_1 - 1.5 \cdot IQR\}$
- $x_{max} = max\{x|x < Q_3 + 1.5 \cdot IQR\}$

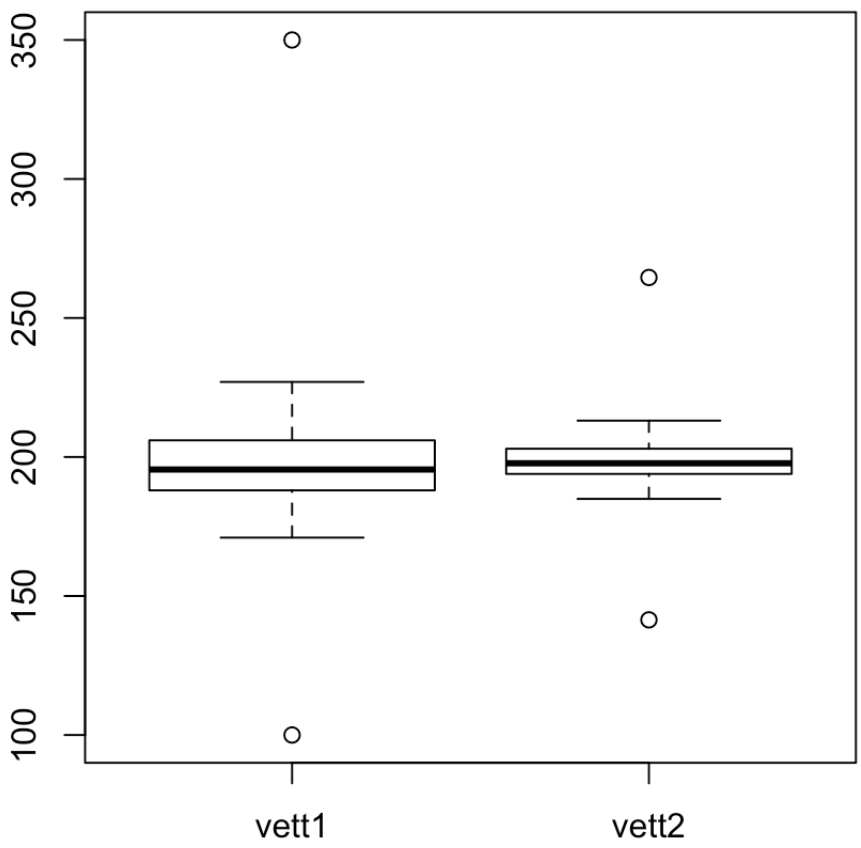
per calcolarli è necessario filtrare i valori e poi prendere il massimo e il minimo

```
vettoreNoOutlier <- vettore[vettore < q3 + (iqr * 1.5) & vettore > q1 - (iqr * 1.5)]
# valore limite superiore
max(vettoreNoOutlier)
## [1] 227
# valore limite inferiore
min(vettoreNoOutlier)
## [1] 171
```

# Boxplot

- Per disegnare un box plot è necessario chiamare `boxplot()`

```
boxplot(vettore, vettore2, main = "due vettori", names = c("vett1","vett2") )
```

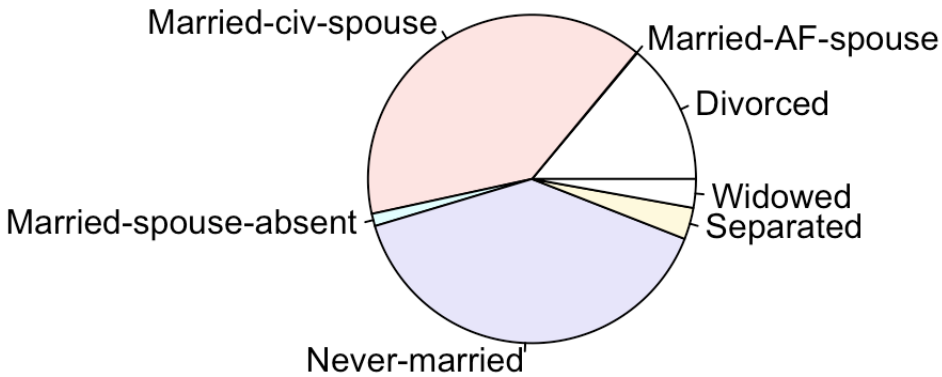

due vettori
350
300
250
200
150
100
vett1
vett2

# Grafici a torta

- Vengono generati con il comando `pie`
- Sono molto comuni e di facile interpretazione
- **Attenzione a non abusarne** (vedi esempi successivi)

```
# esempio usando le frequenze assolute calcolate da R
pie(table(StatoCivile), main = "Torta degli stati civili")
```

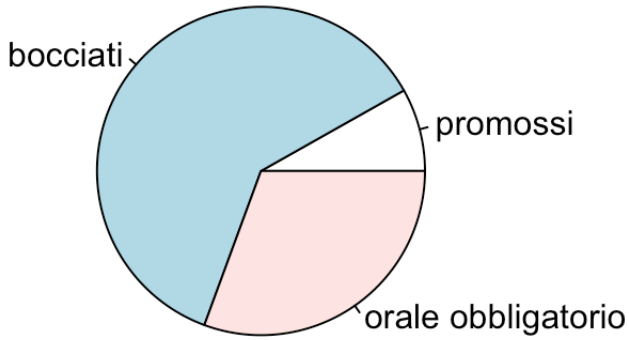
## Torta degli stati civili



# Grafici a torta (cont.)

- Vengono generati con il comando `pie`
- Sono molto comuni e di facile interpretazione
- **Attenzione a non abusarne** (vedi esempi successivi)

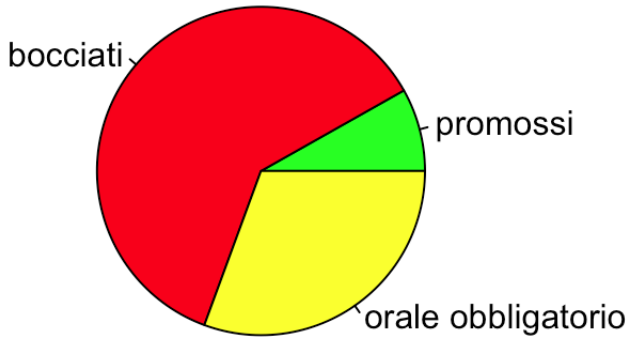
```
# esempio usando due vettori, uno di dati e uno di etichette
dati <- c (4,30,15)
etichette <- c("promossi", "bocciati", "orale obbligatorio")
pie(dati, etichette, main = "Appello gennaio")
```


Appello gennaio
bocciati
promossi
orale obbligatorio

# Grafici a torta (cont.)

- Possiamo specificare un vettore di colori corrispondenti ad ogni fetta
- Oppure una funziona automatica `col=rainbow(length(dati))`

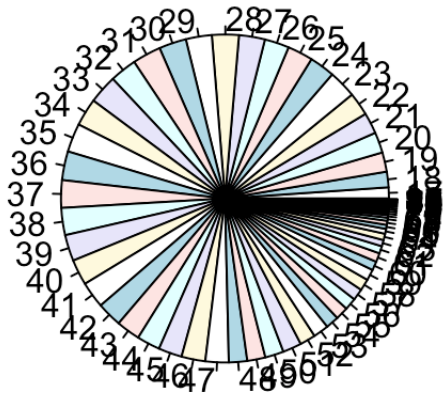
```r
# esempio usando due vettori, uno di dati e uno di etichette
dati <- c (4,30,15)
etichette <- c("promossi", "bocciati", "orale obbligatorio")
pie(dati, etichette, main = "Appello gennaio", col = c("green", "red", "yellow"))
```


Appello gennaio
bocciati
promossi
orale obbligatorio

# Grafici a torta (cont.)

Nel caso di variabili numeriche con molti valori (es. età, peso, altezza) non usate i grafici a torta

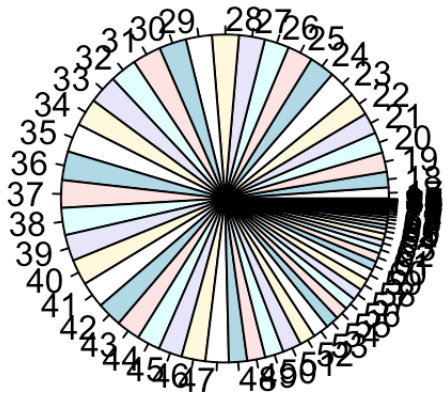
```
pie(table(Eta))
```



# Grafici a torta (cont.)

Nel caso di variabili numeriche con molti valori (es. età, peso, altezza) non usate i grafici a torta

```
pie(table(Eta))
```



*Questo è una porcata!*

# Grafici a torta (cont.)

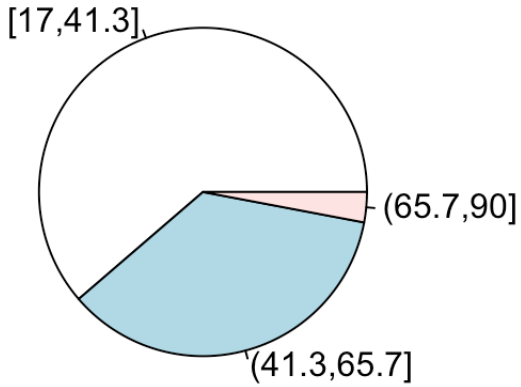
Soluzione:

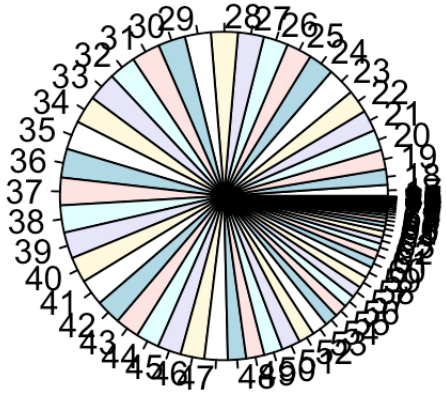
- creare degli intervalli di valori e usarli per dividere i dati col comando `cut`
- usare il comando table per calcolare le frequenze degli intervalli e plottarle

```
# creiamo un vettore che divida i dati in 3 blocchi -> lunghezza 4
vettoreBreaks <- seq (from = min(Eta), to = max(Eta), length = 4)
table(cut(Eta, breaks = vettoreBreaks, include.lowest = TRUE))
##
##   [17,41.3] (41.3,65.7]   (65.7,90]
##       27725       16153        1344
```

# Grafici a torta (cont.)

```r
frequenzeIntervalli <- table(cut(Eta, breaks = vettoreBreaks, include.lowest = TRUE))
pie(frequenzeIntervalli)
pie(table(Eta))
```



Impostate la lunghezza del vettore con i breaks in maniera ragionevole.

# Esercizio 1

1. Scaricare e importare il dataset capelli.csv
2. Usare le funzioni predefinite di R calcolate covarianza e correlazione tra la lunghezza e le altre misure.
3. Fare dei grafici a dispersione per confermare la vostra ipotesi
4. Fare un boxplot e un istogramma della colonna della spesaShampoMensile, cosa li accomuna?
5. Fare un grafico a torta della probabilità di fidanzamento dividendo in 3 blocchi i dati in questo modo:
    - *nell'intervallo* $[0, 0.3]$ *uncool*
    - *tra* $(0.3, 0.6]$ *normali*
    - *tra* $(0.6, 1]$ *cool*

# Esercizio 2 (really hard)

Per questo esercizio è necessario aver compreso molto bene la teoria.

1. Scaricare e importare il dataset daticlassificatore.csv
2. Visualizzare le prime 6 righe del dataframe
3. Calcore specificity e sensibility al variare della soglia.
4. Fare un plot della curva di ROC
5. Quale valore sembra ragionevole da usare come soglia per il nostro classificatore?

Aiuto

# Questionario