STATISTICA E ANALISI DEI DATI

Capitolo 3 – Diagramma di Pareto

Dott. Stefano Cirillo
Dott. Luigi Di Biasi

a.a. 2025-2026

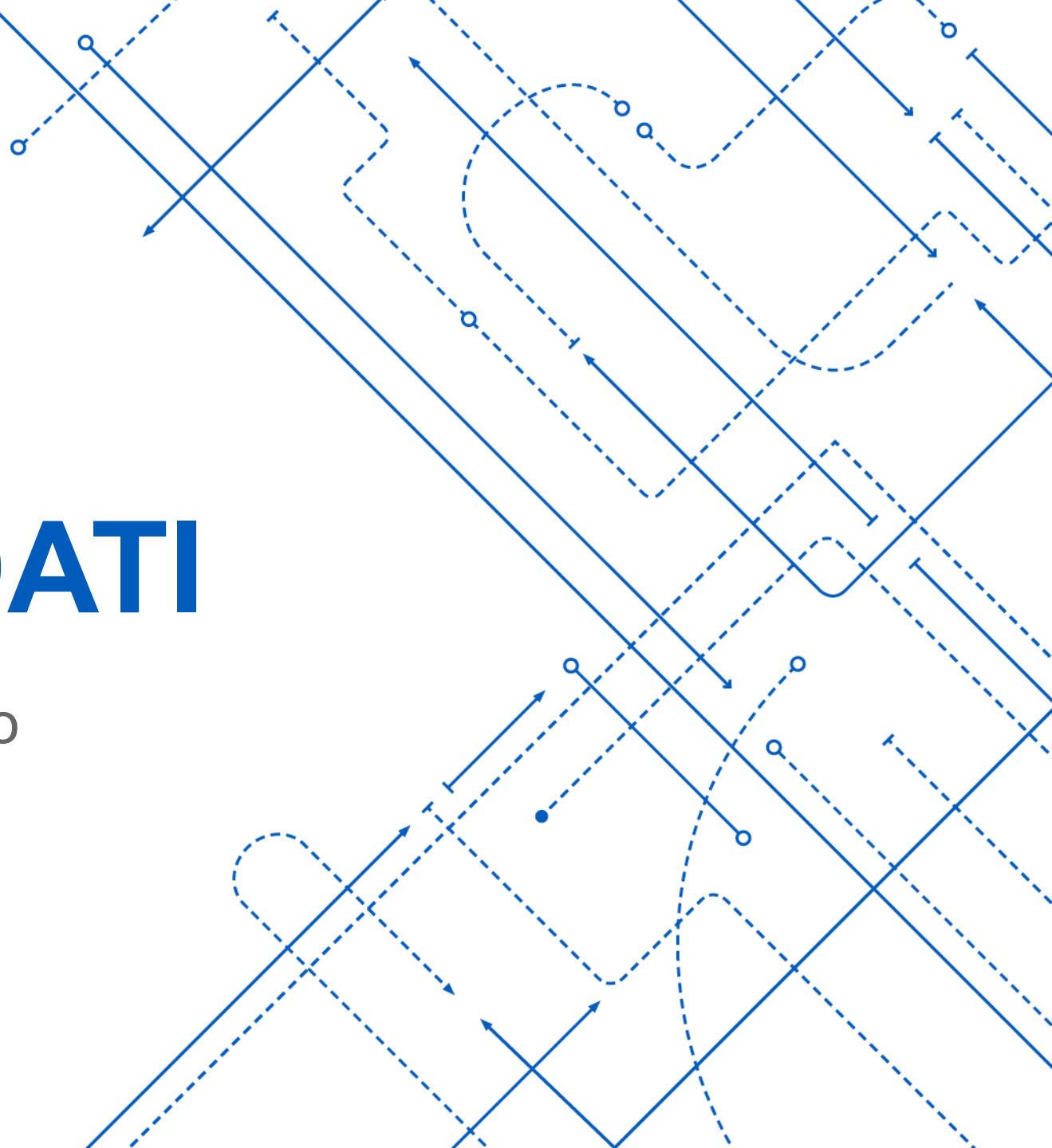


DIAGRAMMA DI PARETO

- Nel 1897 l'economista italiano **Vilfredo Pareto** (1848–1923), studiando la distribuzione dei redditi, mostrò che in una data regione solo pochi individui possedevano la maggior parte della ricchezza
- Questa osservazione ispirò la cosiddetta "**legge 80/20**", una legge empirica che è anche nota con il nome di principio di Pareto:

  "La maggior parte degli effetti è dovuta ad un numero ristretto di cause (considerando grandi numeri)"

- Secondo la legge 80/20 l'80% dei risultati dipende dal 20% delle cause
- Applicazioni Reali:
  - Economia: l'80% delle ricchezze è in mano al 20% della popolazione;
  - Qualità: il 20% dei tipi di guasti possibili in un processo produttivo genera l'80% dei problemi totali;
  - Informatica: l'80% del tempo di esecuzione è impiegato soltanto dal 20% delle istruzioni di un programma;
  - Telefonate: l'80% delle chiamate viene effettuato da e verso il 20% dei contatti in memoria;
  - ….

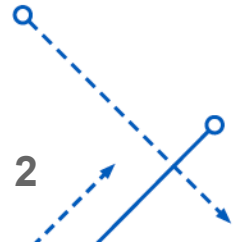


DIAGRAMMA DI PARETO

- Attraverso questo principio possiamo affermare che è possibile analizzare un insieme di dati in modo da determinare le poche variabili (fra le tante prese in esame) che influenzano in modo significativo i risultati finali di un determinato fenomeno (analisi di Pareto)
- Il diagramma di Pareto permette di analizzare fenomeni qualitativi
  - È un diagramma a **barre verticali** con le modalità **ordinate in ordine decrescente** rispetto alla loro frequenza relativa
  - Le frequenze relative sono visualizzate anche nella loro forma cumulata, mediante una sequenza di **segmenti crescenti**

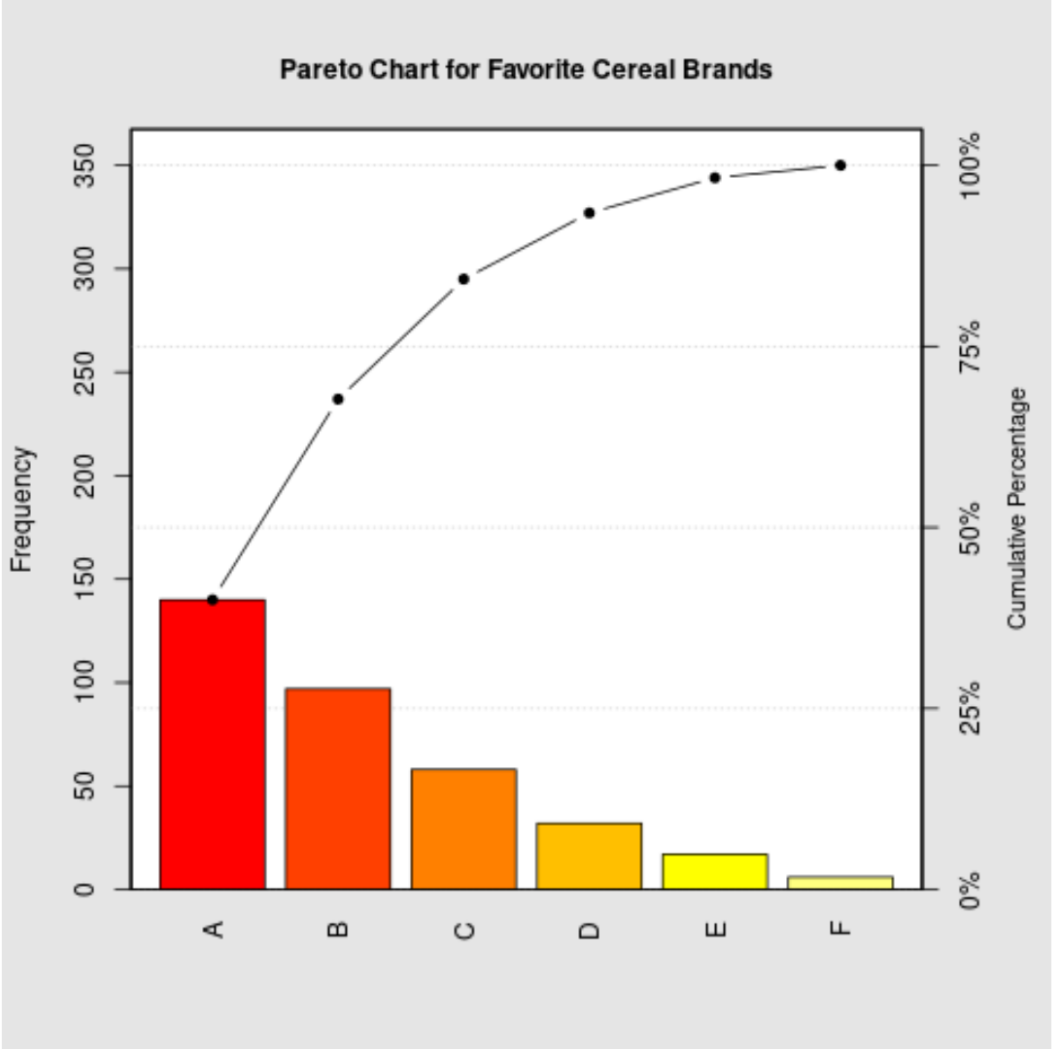
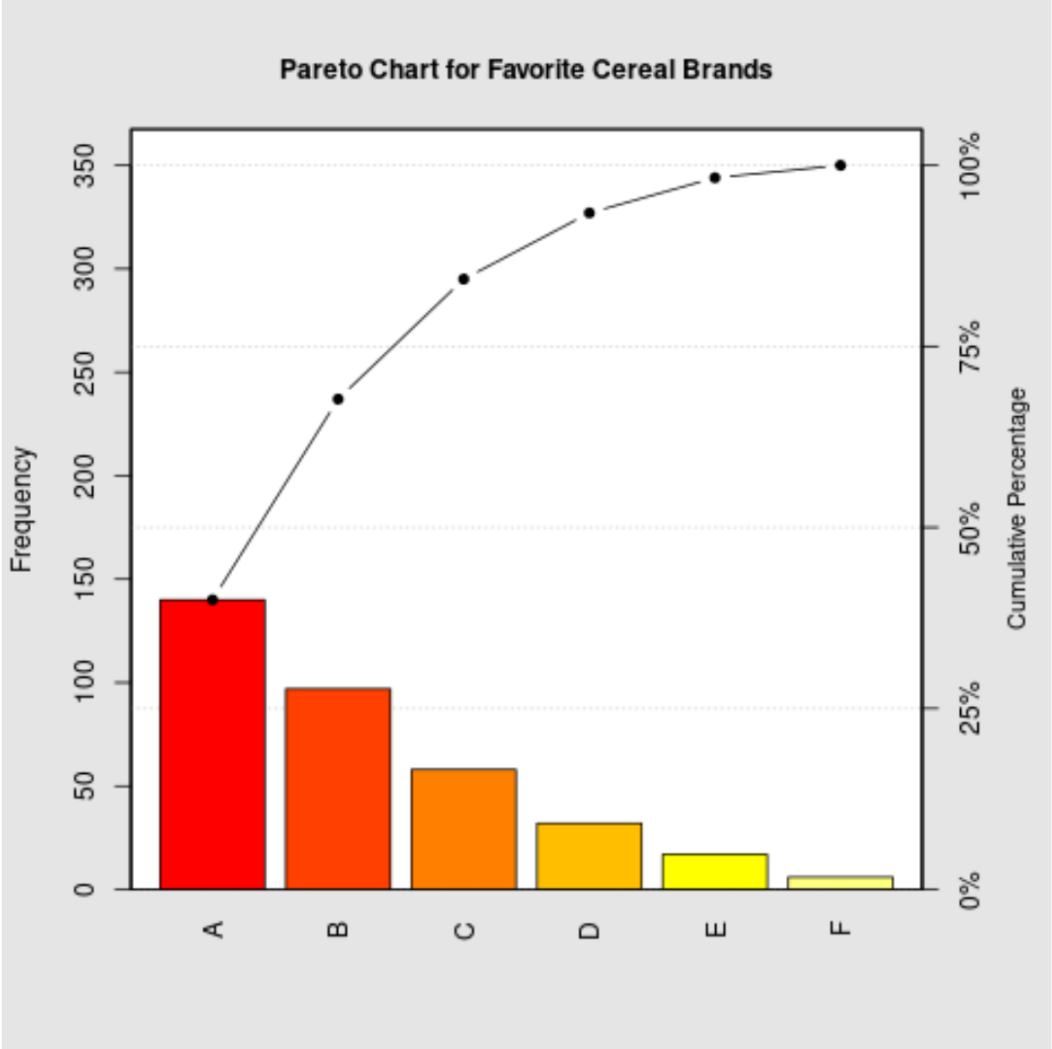


DIAGRAMMA DI PARETO

- L'analisi del diagramma si articola in 5 fasi:
  1. Decidere come classificare i dati;
  2. Rilevare i dati ed ordinarli;
  3. Disegnare il diagramma;
  4. Costruire la linea cumulativa;
  5. Aggiungere le informazioni di base.



ESEMPIO

- Supponiamo di dover valutare la qualità di un prodotto, come ad esempio un notebook
- Supponiamo di aver osservato per un determinato intervallo temporale, la numerosità dei difetti dei componenti
- **Fase 1**: Scelta del metodo con cui classificare i dati da raccogliere
  - Supponiamo di classificare i prodotti difettosi per tipo di componente difettoso
- **Fase 2**: Analisi dei dati raccolti per un determinato arco temporale

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

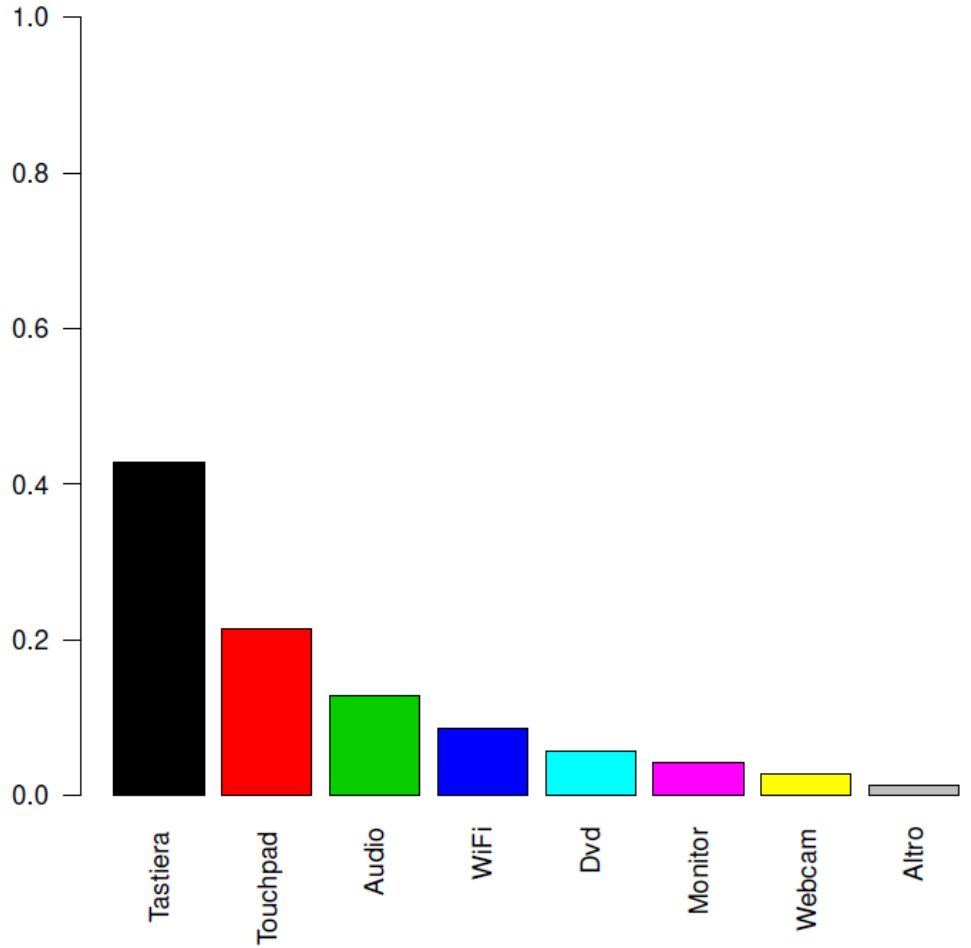
ESEMPIO

- Prima di costruire il diagramma di Pareto risulta utile riordinare le occorrenze nella tabella in base alla **rilevanza del parametro in esame**
  - Parametro: quantità di pezzi difettosi
  - Ordinamento: elenca prima il difetto più numeroso (Crescente)
- **Nota**:
  - Ai fini della rappresentazione nel diagramma di Pareto è utile raggruppare i difetti minori sotto la voce altro o altre cause
  - Tale voce dovrebbe risultare sempre all'ultimo posto in quanto deve essere numericamente non rilevante
  - Altrimenti nella nuova categoria sono confluite tipologie di difetti numerosi che invece meritano una valutazione specifica

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

Esempio

- **Fase 3**: rappresentazione grafica dei dati rilevati costruendo un grafico a barre
  - Asse verticale: mostra la frequenza relativa dei difetti rilevati
  - Asse orizzontale: i tipi di difetti
    - Il difetto più frequente è stato rilevato 30 volte per la tastiera;
      - ➢ quindi si traccerà una colonna alta fino al livello $\frac{30}{70} = 0.42857143$
    - Il secondo difetto più frequente è stato rilevato 15 volte:
      - ➢ a lato della colonna precedentemente disegnata ne tracceremo un'altra che arrivi al livello $\frac{15}{70} = 0.21428571$
    - Si continua per ogni difetto analizzato



ESEMPIO

- **Fase 4**: Tracciare la cosiddetta **linea dei valori cumulati** o linea cumulativa
  - Questa linea risulta utile quando si vogliono individuare le percentuali cumulate di più colonne
  - Segmento 1: congiunge il punto che indica la percentuale dei difetti del primo tipo con quello ad altezza pari alla somma delle percentuali dei difetti del primo e secondo tipo
  - Altri segmenti: si ripete il procedimento per i segmenti successivi;
  - Ultimo segmento: terminerà nel punto più alto della scala percentuale, corrispondente al 100% dei difetti.

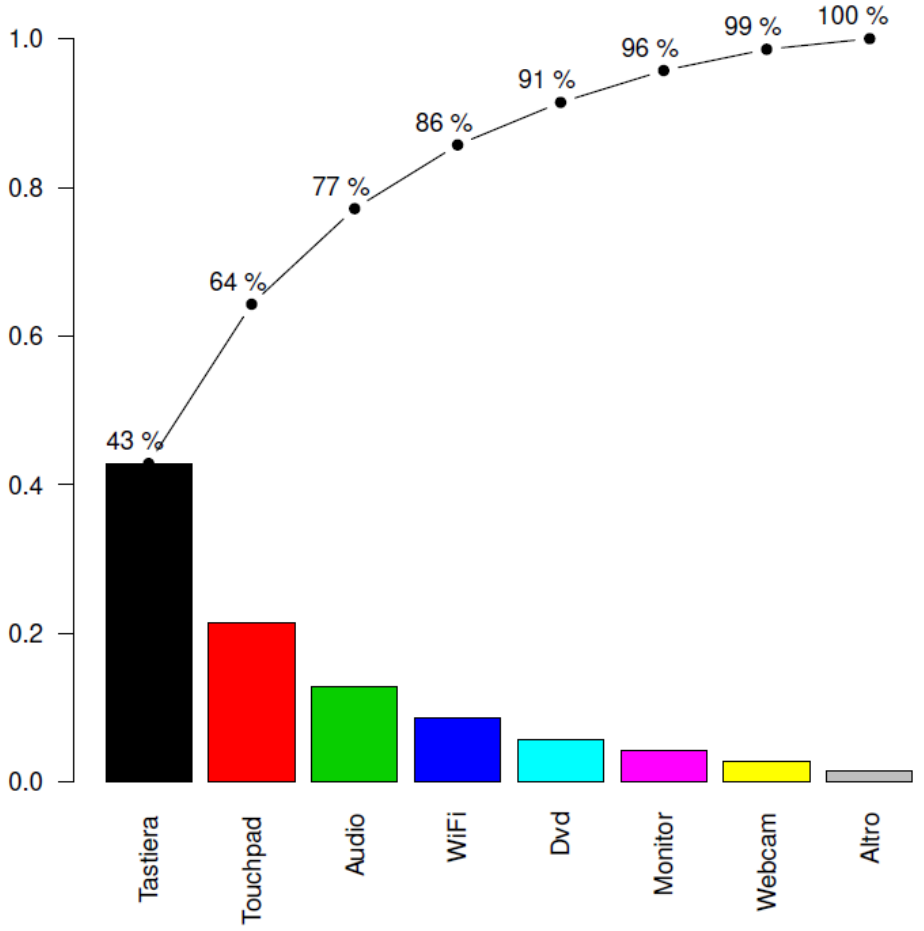


Diagramma di Pareto (R)

- **Fase 1**:

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

'Audio' · 'Audio' · 'Audio' · 'Audio' · 'Audio' · 'Audio' · 'Audio' · 'Audio' · 'Audio' · 'Dvd' · 'Dvd' · 'Dvd' · 'Dvd' · 'Monitor' · 'Monitor' · 'Monitor' · 'Tastiera' · 'Touchpad' · 'Touchpad' · 'Touchpad' · 'Touchpad' · 'Touchpad' · 'Touchpad' · 'Touchpad' · 'Touchpad' · 'Touchpad' · 'Touchpad' · 'Touchpad' · 'Touchpad' · 'Touchpad' · 'Touchpad' · 'Touchpad' · 'Webcam' · 'Webcam' · 'WiFi' · 'WiFi' · 'WiFi' · 'WiFi' · 'WiFi' · 'WiFi' · 'Altro'

- **Fase 2**:

```
tab <- table(computer)
ord <- sort(tab,decreasing=TRUE) #Ordino i dati
propOrd <- prop.table(ord)
propOrd
```

```
computer
  Tastiera   Touchpad      Audio       WiFi        Dvd    Monitor     Webcam
0.42857143 0.21428571 0.12857143 0.08571429 0.05714286 0.04285714 0.02857143
     Altro
0.01428571
```

DIAGRAMMA DI PARETO (R)

- **Fase 3**:

```
x <- barplot(propOrd,ylim=c(0,1.05), main="Diagramma di Pareto", col=1:8,las=2)
```

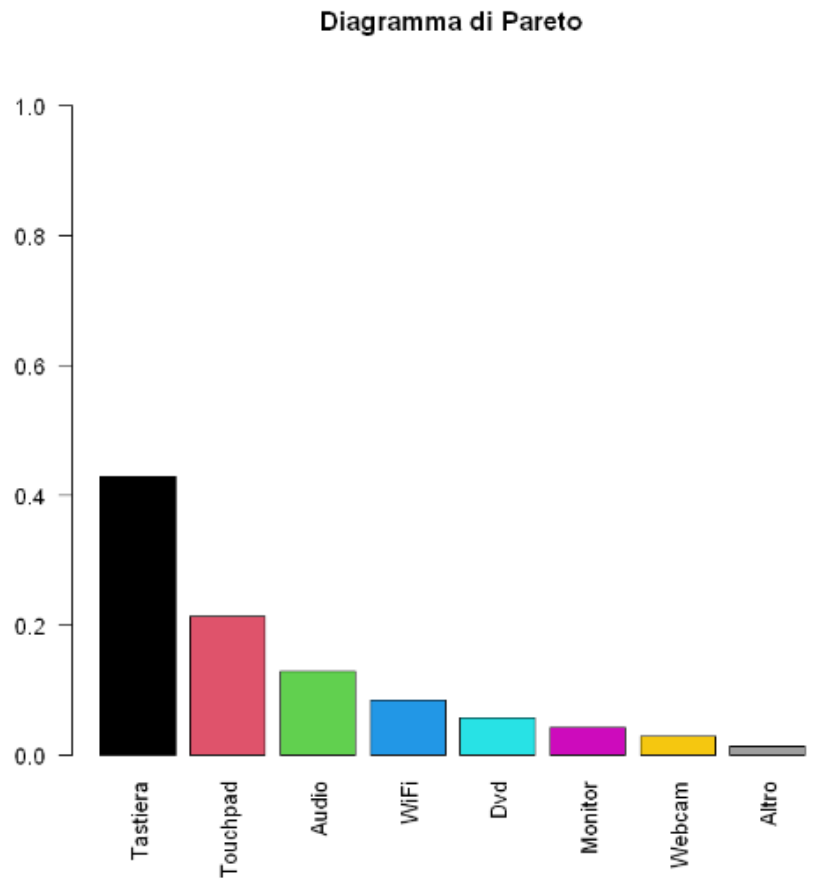


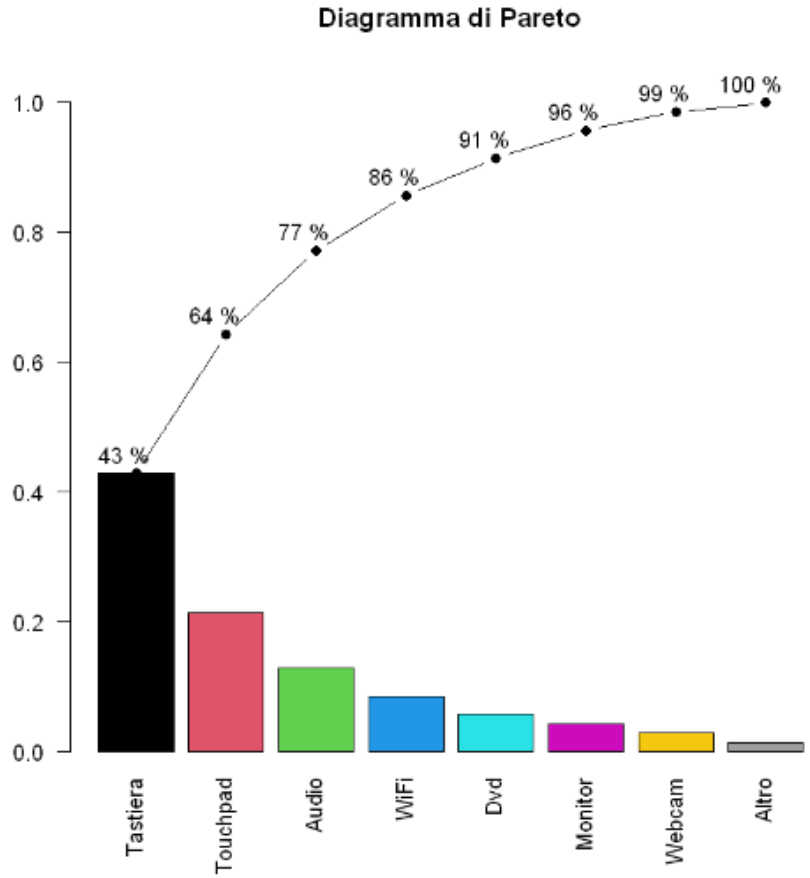
DIAGRAMMA DI PARETO (R)

- **Fase 4**:

```
lines(x,cumsum(propOrd),type="b",pch=16)
text(x - 0.2, cumsum(propOrd) + 0.03, paste(format(cumsum(propOrd)*100, digits=2),"%"))
```

- Dovendo quindi stabilire una strategia di ottimizzazione conviene migliorare la tastiera, il touchpad e il sistema audio che appresentano l'80% della difettosità totale.

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

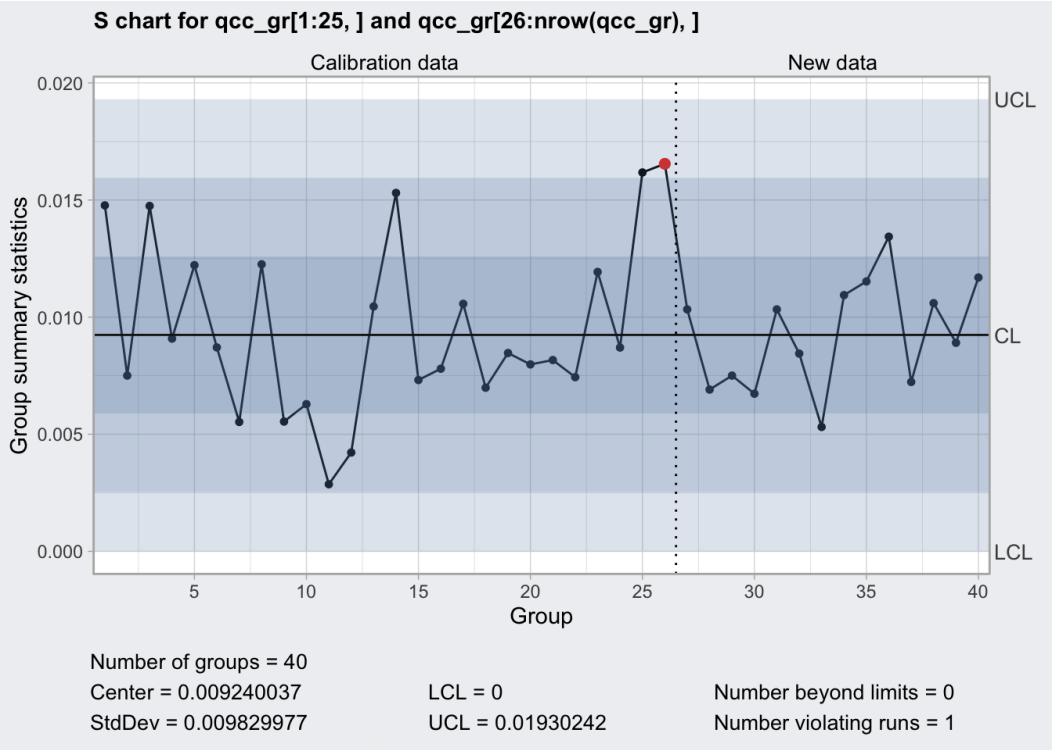


LIBRERIA QCC

- qcc = *Quality Control Charts*, una libreria R dedicata al **controllo statistico della qualità**
- Permette di creare:
  - Diagrammi di **controllo di processo** (Pareto, Shewhart, c-chart, ecc.)
  - **Diagrammi di Pareto**
  - Analisi di **capacità di processo**
- Installazione:

```
if(!require(qcc)) install.packages("qcc")
library(qcc)
```

  - qcc() → genera una carta di controllo per dati di processo
  - pareto.chart() → crea un diagramma di Pareto
  - process.capability() → analizza la capacità di processo
  - qcc.groups() → gestisce dati in sottogruppi per carte di controllo



ESEMPIO (1)

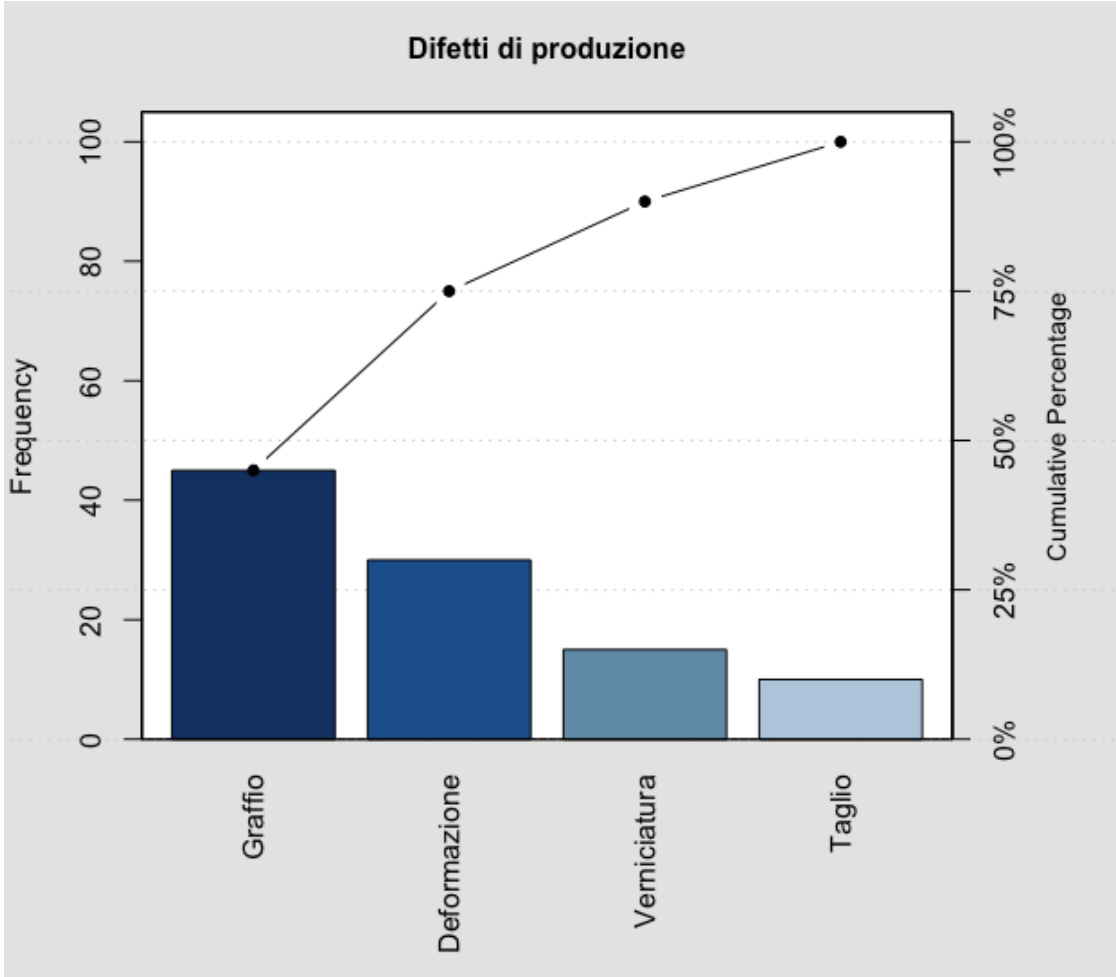
- **Contesto industriale o produttivo**
  - In un'azienda manifatturiera, su 1000 pezzi difettosi, si scopre che:
  - il 70% dei difetti deriva da 3 tipi di errori di lavorazione su 10 totali
    - Il diagramma di Pareto aiuta a **concentrarsi su quei 3 tipi di difetti** principali per migliorare la qualità

```
if(!require(qcc)) install.packages("qcc")
library(qcc)

par(mfrow = c(1, 1))

difetti <- c(Graffio = 45, Deformazione = 30, Verniciatura = 15, Taglio = 10)
pareto.chart(difetti, main = "Difetti di produzione")
```

```
  Pareto chart analysis for difetti
               Frequency Cum.Freq. Percentage Cum.Percent.
    Graffio           45        45         45           45
    Deformazione      30        75         30           75
    Verniciatura      15        90         15           90
    Taglio            10       100         10          100
```



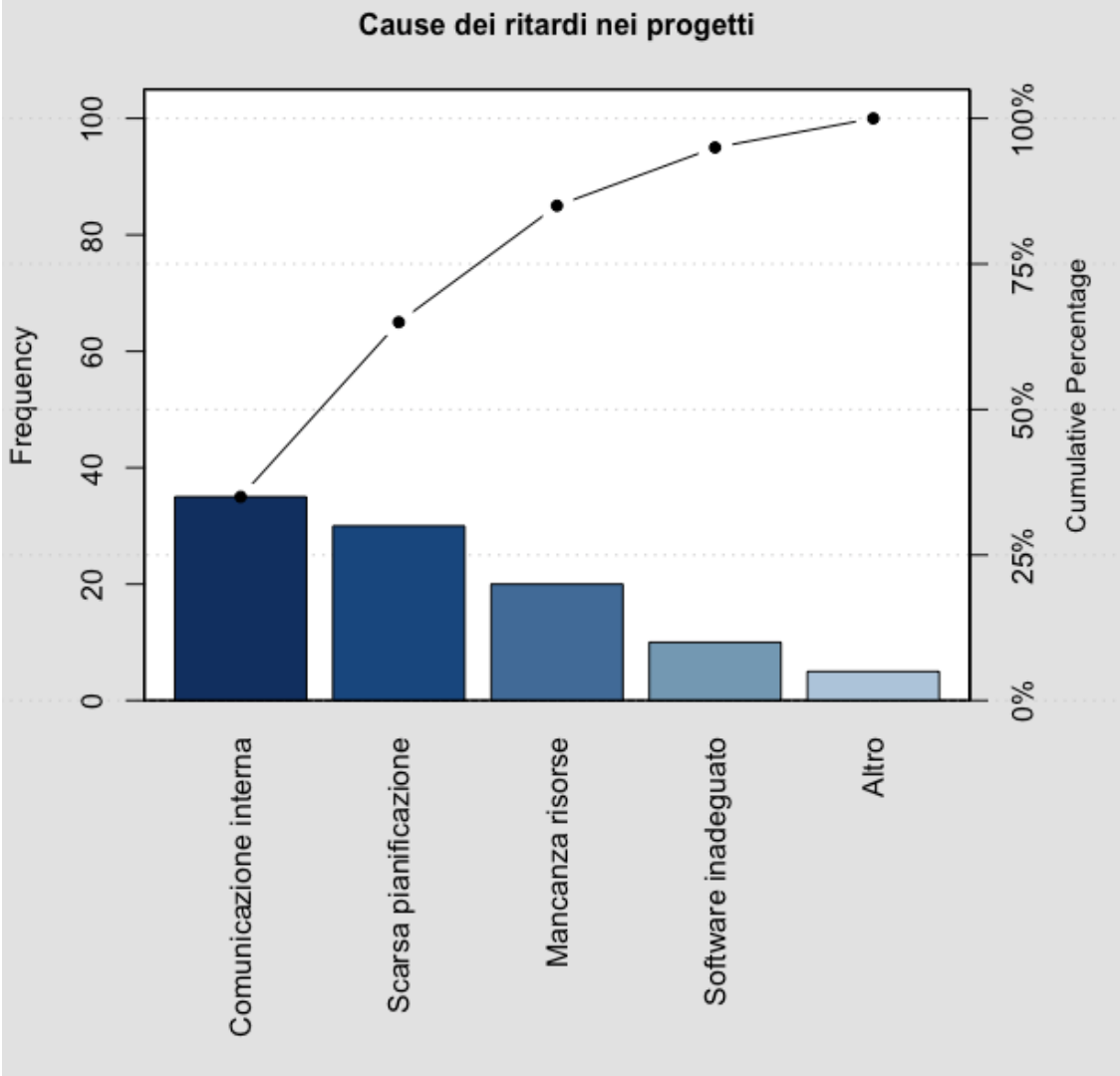
ESEMPIO (2)

- **Gestione aziendale**
  - In un'azienda di servizi, si vuole capire perché i progetti subiscono ritardi:
  - Comunicazione interna, scarsa pianificazione, mancanza di risorse, software inadeguato, ecc.
    - Il Pareto aiuta a capire **quali fattori pesano di più** sui ritardi complessivi.

```
if(!require(qcc)) install.packages("qcc")
library(qcc)

ritardi <- c("Comunicazione interna" = 35,
             "Scarsa pianificazione" = 30,
             "Mancanza risorse" = 20,
             "Software inadeguato" = 10,
             "Altro" = 5)
pareto.chart(ritardi, main = "Cause dei ritardi nei progetti")

Pareto chart analysis for ritardi
                      Frequency Cum.Freq. Percentage Cum.Percent.
  Comunicazione interna        35        35         35           35
  Scarsa pianificazione        30        65         30           65
  Mancanza risorse             20        85         20           85
  Software inadeguato          10        95         10           95
  Altro                         5       100          5          100
```

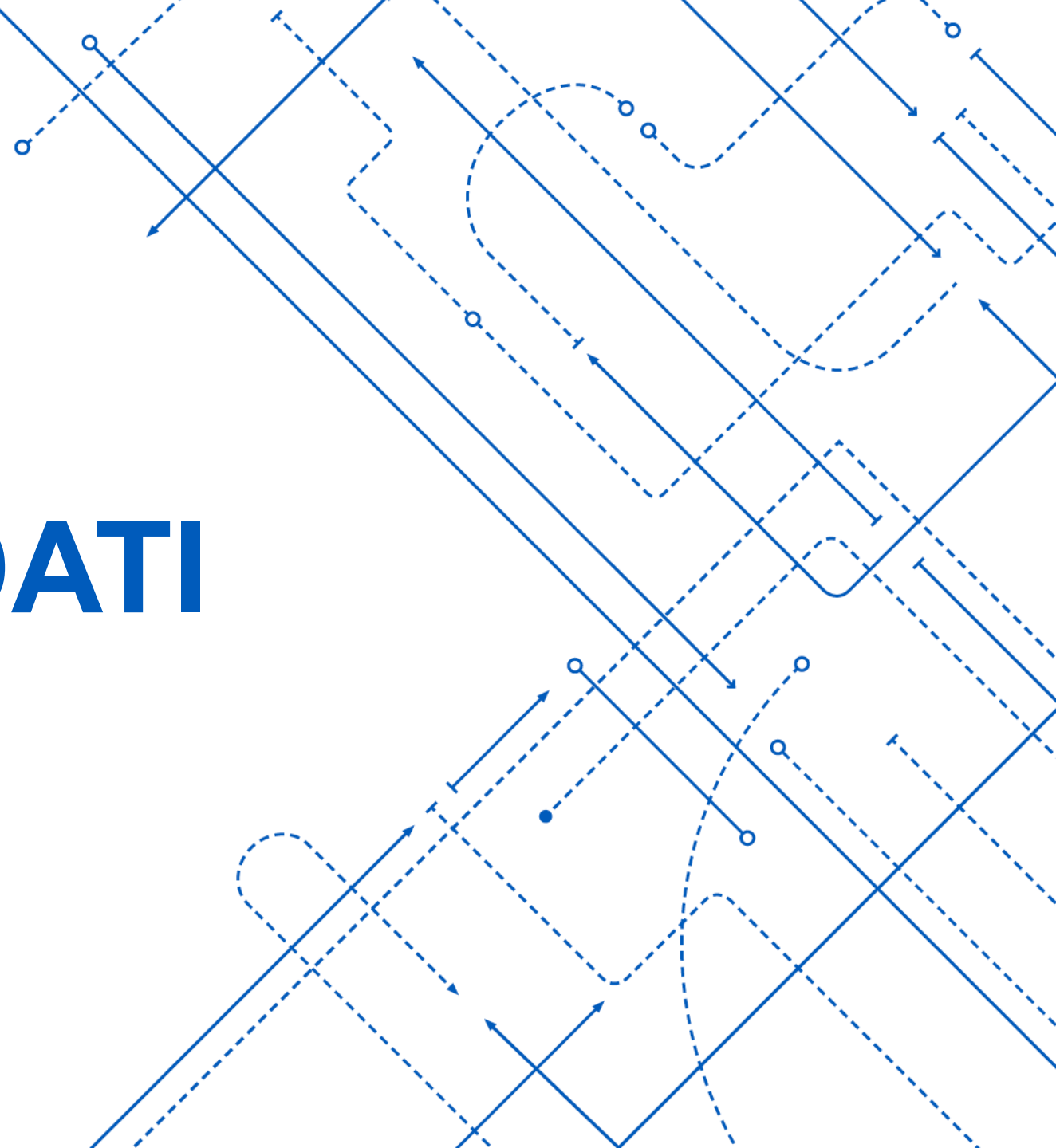


STATISTICA E ANALISI DEI DATI

Capitolo 3 – Grafici di Funzione

Dott. Stefano Cirillo
Dott. Luigi Di Biasi

a.a. 2025-2026



GRAFICI DI FUNZIONE

- Con il linguaggio R si possono rappresentare graficamente molte funzioni matematiche
- La funzione:

curve(*expr, from, to*)

- disegna una curva sulla base dell'espressione indicata in expr, nell'intervallo [*from*, *to*]

GRAFICI DI FUNZIONE

- Con il linguaggio R si possono rappresentare graficamente molte funzioni matematiche
- La funzione:

curve(*expr, from, to*)

- disegna una curva sulla base dell'espressione indicata in expr, nell'intervallo [*from*, *to*]
- Esempio:
  - Vogliamo tracciare il grafico della funzione:

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

  - Ha un asintoto orizzontale in y = 3
  - È decrescente nell'intervallo (−∞,−2)
  - È crescente in (−2, 0)
  - È decrescente in (0,+∞)

GRAFICI DI FUNZIONE

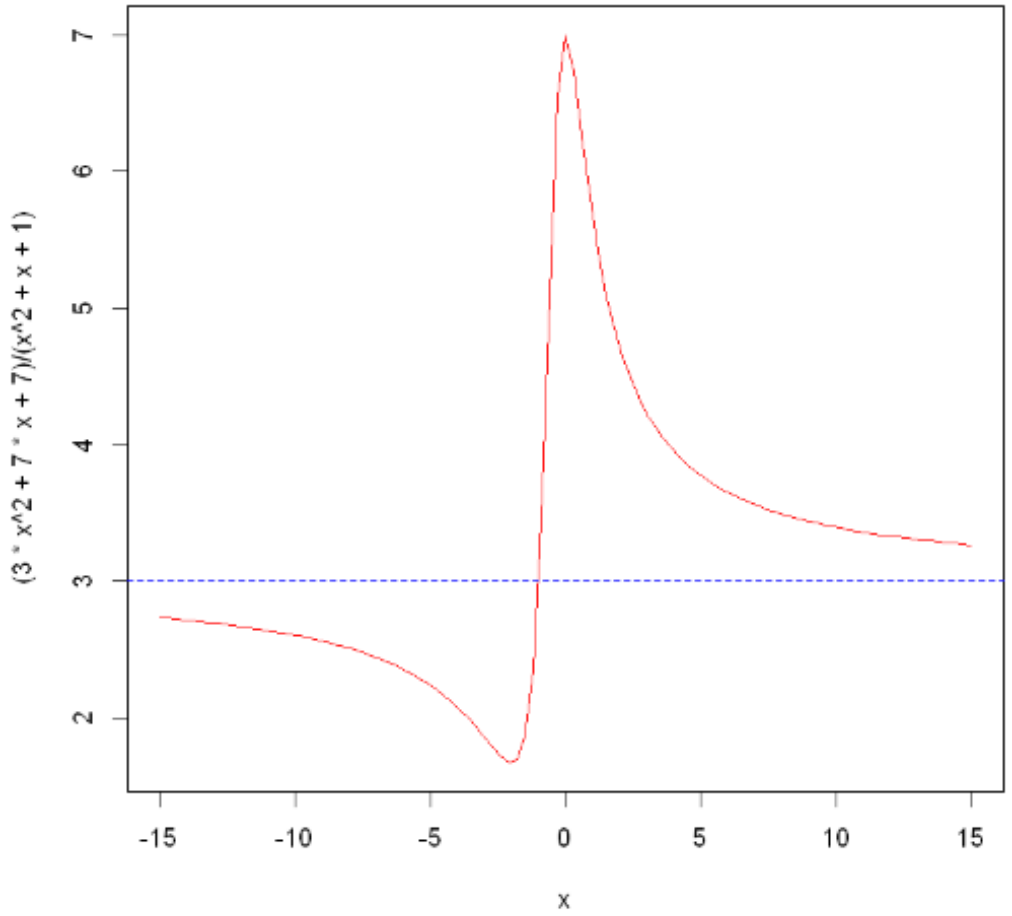
```
curve((3*x^2+7*x+7)/(x^2+x+1), from=-15, to=15, col="red")
abline(h=3,lty=2,col="blue")
```

- La funzione **abline**() permette di aggiungere al grafico della funzione una linea orizzontale
- Nel grafo della funzione $f(x)$ abbiamo aggiunto

y = 3

rappresentante l'asintoto orizzontale

  - L'argomento **lty** = 2 specifica che tale linea deve essere tratteggiata



GRAFICI MULTIPLI

- È possibile disegnare più grafici nella stessa finestra grafica utilizzando le funzioni:

par(*mfrow* = c(nr, nc))

par(*mfcol* = c(nr, nc))

- Parametro **nr:** numero di grafici da mettere in riga
- Parametro **nc:** numero di grafici da mettere in colonna
- Cioè definisco una **griglia di rappresentazione** in cui poter inserire **nr * nc** grafici

GRAFICI MULTIPLI

- È possibile disegnare più grafici nella stessa finestra grafica utilizzando le funzioni:

par(*mfrow* = c(nr, nc))

par(*mfcol* = c(nr, nc))

- Parametro **nr:** numero di grafici da mettere in riga
- Parametro **nc:** numero di grafici da mettere in colonna
- Cioè definisco una **griglia di rappresentazione** in cui poter inserire **nr * nc** grafici
- Esempio:
  - Griglia 1 riga e 2 colonne: Totale grafici 2

```
#Definisco la griglia di rappresentazione
par(mfrow=c(1,2))
```

  - Griglia 2 riga e 2 colonne: Totale grafici 4

```
#Definisco la griglia di rappresentazione
par(mfrow=c(2,2))
```

GRAFICI MULTIPLI

- Esempio: Rappresentiamo le seguenti quattro funzioni in una sola figura:

$$\cos(x)\,, \sin(x), x * \cos(x) - \sin(x)\,, \frac{\sin(x)}{x}$$

```r
#Definisco la griglia di rappresentazione
par(mfrow=c(2,2))

#Grafico 1
curve(cos(x),from=0,to=6*pi)
abline(h=0,col="red")
title("Funzione coseno")
```

GRAFICI MULTIPLI

- Esempio: Rappresentiamo le seguenti quattro funzioni in una sola figura:

$$\cos(x)\,, \sin(x), x * \cos(x) - \sin(x)\,, \frac{\sin(x)}{x}$$

```
#Definisco la griglia di rappresentazione
par(mfrow=c(2,2))

#Grafico 1
curve(cos(x),from=0,to=6*pi)
abline(h=0,col="red")
title("Funzione coseno")

#Grafico 2
curve(sin(x),from=0,to=6*pi)
abline(h=0,col="red")
title("Funzione seno")
```

GRAFICI MULTIPLI

- Esempio: Rappresentiamo le seguenti quattro funzioni in una sola figura:

$$\cos(x)\,, \sin(x), x * \cos(x) - \sin(x)\,, \frac{\sin(x)}{x}$$

```r
#Definisco la griglia di rappresentazione
par(mfrow=c(2,2))

#Grafico 1
curve(cos(x),from=0,to=6*pi)
abline(h=0,col="red")
title("Funzione coseno")

#Grafico 2
curve(sin(x),from=0,to=6*pi)
abline(h=0,col="red")
title("Funzione seno")

#Grafico 3
curve(x*cos(x)-sin(x),from=0,to=6*pi)
abline(h=0,col="red")
curve(+x,add=TRUE,lty=2,col="blue")
curve(-x,add=TRUE,lty=2,col="blue")
title("Funzione con\nOscillazioni")
```

GRAFICI MULTIPLI

- Esempio: Rappresentiamo le seguenti quattro funzioni in una sola figura (Griglia di rappresentazione 2x2):

$$\cos(x)\,, \sin(x), x * \cos(x) - \sin(x)\,, \frac{\sin(x)}{x}$$

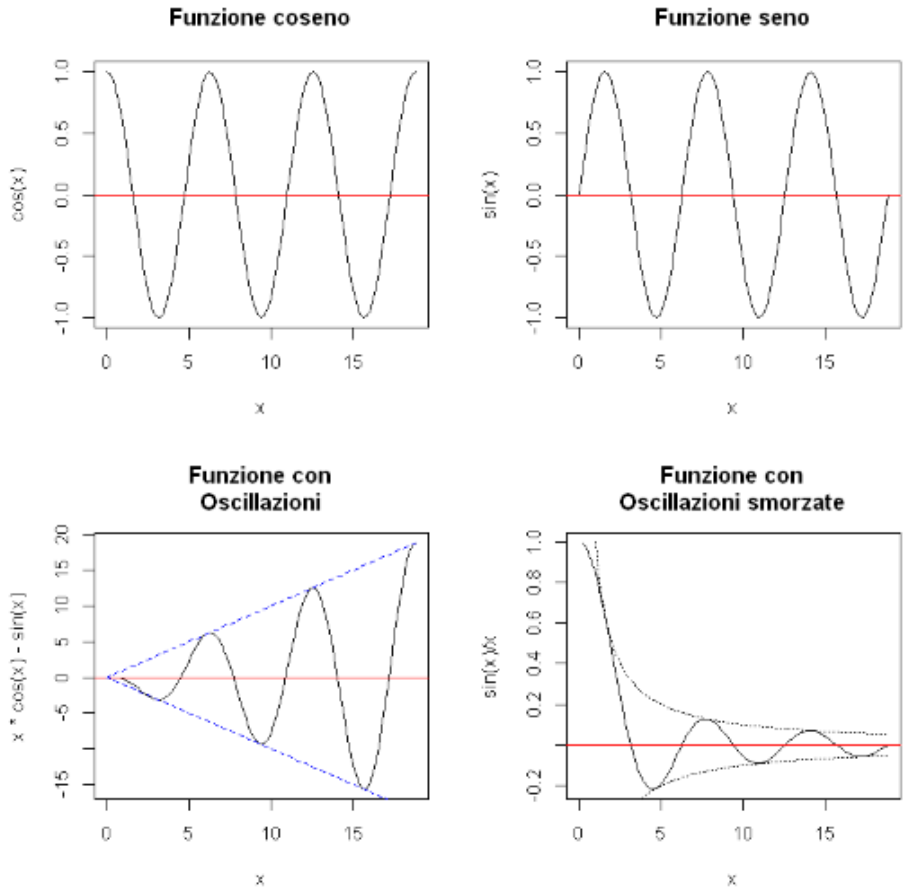
```r
#Definisco la griglia di rappresentazione
par(mfrow=c(2,2))

#Grafico 1
curve(cos(x),from=0,to=6*pi)
abline(h=0,col="red")
title("Funzione coseno")

#Grafico 2
curve(sin(x),from=0,to=6*pi)
abline(h=0,col="red")
title("Funzione seno")

#Grafico 3
curve(x*cos(x)-sin(x),from=0,to=6*pi)
abline(h=0,col="red")
curve(+x,add=TRUE,lty=2,col="blue")
curve(-x,add=TRUE,lty=2,col="blue")
title("Funzione con\nOscillazioni")

#Grafico 4
curve(sin(x)/x,from=0,to=6*pi)
abline(h=0,col="red")
curve(1/x,add=TRUE,lty=3,from=1,to=6*pi,col="black")
curve(-1/x,add=TRUE,lty=3,from=1,to=6*pi,col="black")
title("Funzione con\nOscillazioni smorzate")
```



STATISTICA E ANALISI DEI DATI

Capitolo 3 – Plot per le Correlazioni

Dott. Stefano Cirillo
Dott. Luigi Di Biasi

a.a. 2025-2026

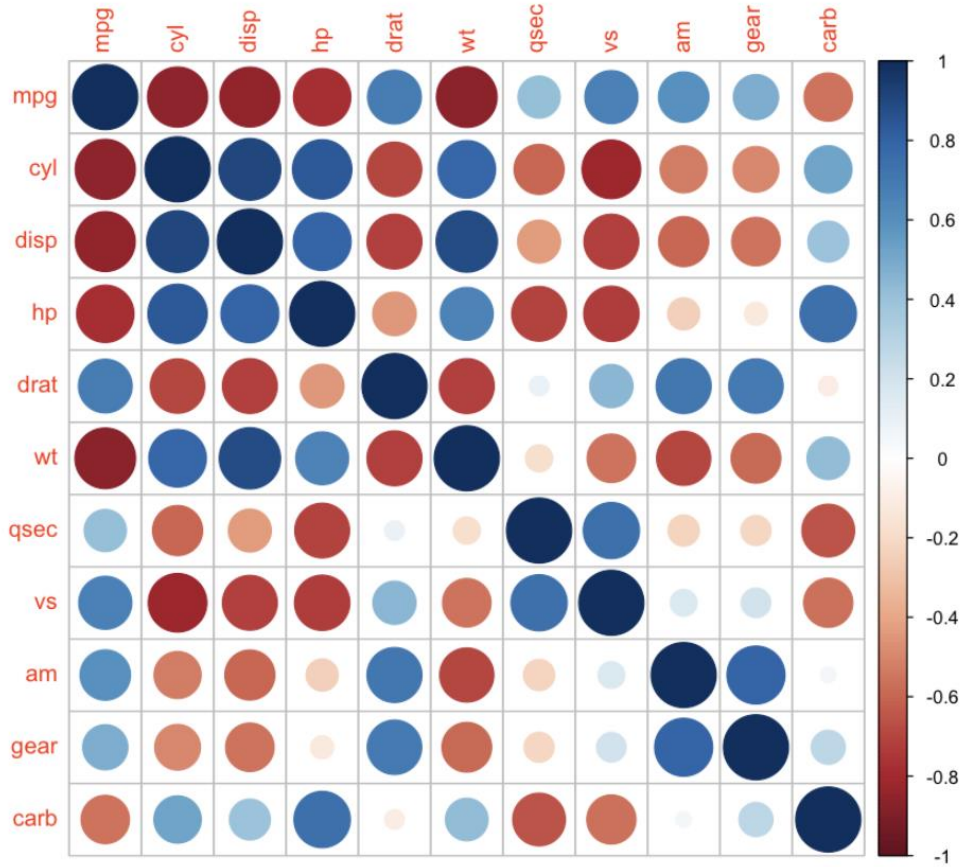
CORRELAZIONI

- La **correlazione** è una misura statistica che descrive la relazione tra due variabili.
  - Essa indica quanto e in che modo due variabili tendono a variare insieme
  - La correlazione può essere utilizzata per identificare se esiste una relazione tra le variabili e la direzione di tale relazione
- Correlazione **Positiva**
  - Si verifica quando un aumento in una variabile corrisponde a un aumento nell'altra variabile
    - Esempio: Altezza e peso. In generale, le persone più alte tendono ad avere un peso maggiore
- Correlazione **Negativa**
  - Si verifica quando un aumento in una variabile corrisponde a una diminuzione nell'altra variabile
    - Esempio: Il consumo di carburante e la distanza percorsa. Aumentando la distanza, si tende a utilizzare meno carburante per unità di distanza (in condizioni ideali)
- **Assenza** di Correlazione
  - Non c'è una relazione chiara tra le due variabili; i cambiamenti in una variabile non influenzano l'altra
    - Esempio: La dimensione delle scarpe e l'altezza in un campione eterogeneo di persone

PLOT PER LE CORRELAZIONI

- Il pacchetto corrplot è utilizzato per visualizzare matrici di correlazione
- Fornisce una rappresentazione grafica intuitiva delle correlazioni, facilitando l'interpretazione dei dati
- Installazione e caricamento

```
install.packages("corrplot")
library(corrplot)
```



PLOT PER LE CORRELAZIONI

- Il pacchetto corrplot è utilizzato per visualizzare matrici di correlazione
- Fornisce una rappresentazione grafica intuitiva delle correlazioni, facilitando l'interpretazione dei dati
- Installazione e caricamento

```
install.packages("corrplot")
library(corrplot)
```

- Caricamento dei dati:

```
#loading the dataset
data("iris")

# Select numeric features
numeric_iris <- iris[, sapply(iris, is.numeric)]
```

PLOT PER LE CORRELAZIONI

- Calcolo le correlazioni mediante il metodo cor():

```
correlations <- cor(numeric_iris)
correlations
```

A matrix: 4 × 4 of type dbl

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
Sepal.Length	1.0000000	-0.1175698	0.8717538	0.8179411
Sepal.Width	-0.1175698	1.0000000	-0.4284401	-0.3661259
Petal.Length	0.8717538	-0.4284401	1.0000000	0.9628654
Petal.Width	0.8179411	-0.3661259	0.9628654	1.0000000

PLOT PER LE CORRELAZIONI

- Calcolo le correlazioni mediante il metodo cor():

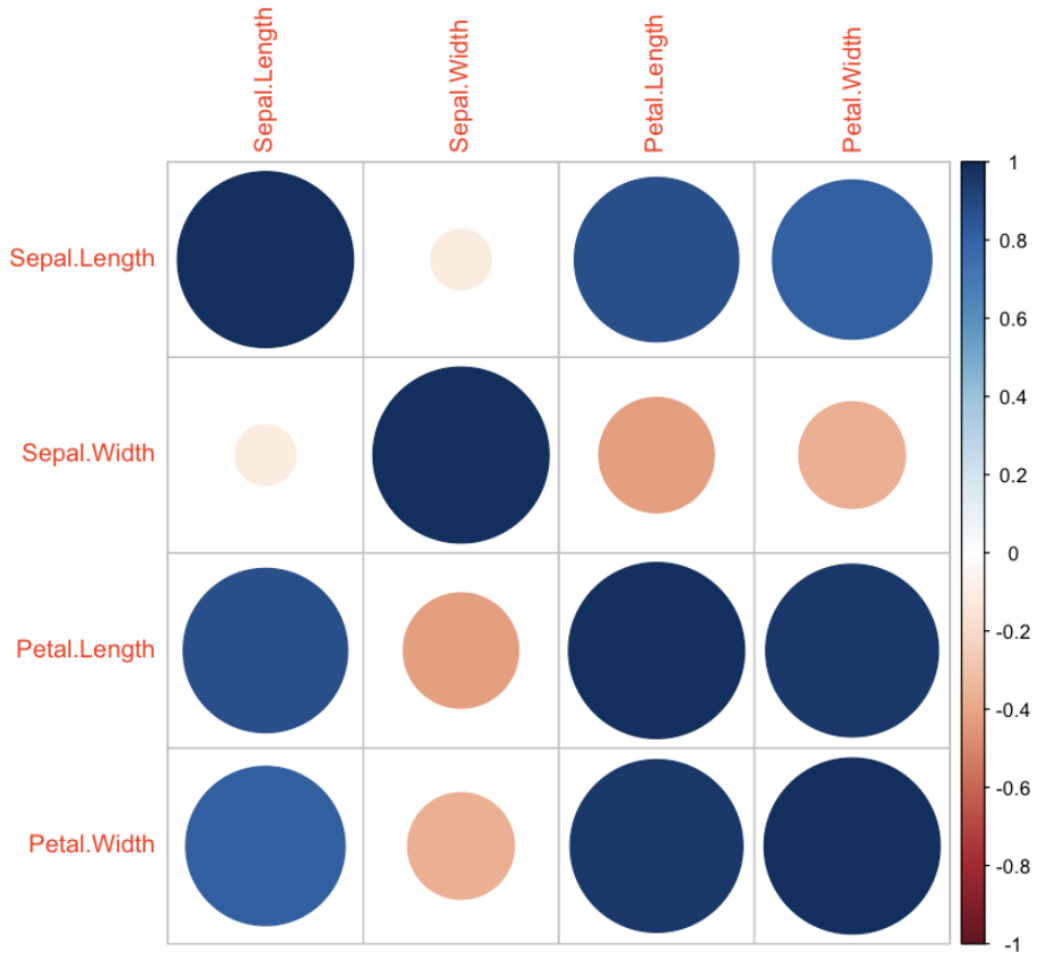
```
correlations <- cor(numeric_iris)
correlations
```

A matrix: 4 × 4 of type dbl

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
Sepal.Length	1.0000000	-0.1175698	0.8717538	0.8179411
Sepal.Width	-0.1175698	1.0000000	-0.4284401	-0.3661259
Petal.Length	0.8717538	-0.4284401	1.0000000	0.9628654
Petal.Width	0.8179411	-0.3661259	0.9628654	1.0000000

- Creo il plot per le correlazioni delle feature di *Iris*:

```
corrplot(cor(numeric_iris)) #it creates the correlation matrix
```

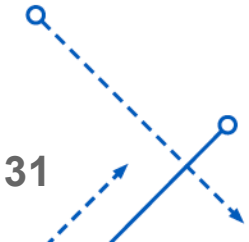


PLOT PER LE CORRELAZIONI

- Il metodo corrplot() del pacchetto corrplot è utilizzato per visualizzare le matrici di correlazione
  - La firma del metodo e i principali parametri che puoi utilizzare:

corrplot(m, method = "circle", type = "full", addCoef.col = NULL, tl.col = "black", tl.srt = 45, diag = TRUE, main = NULL)

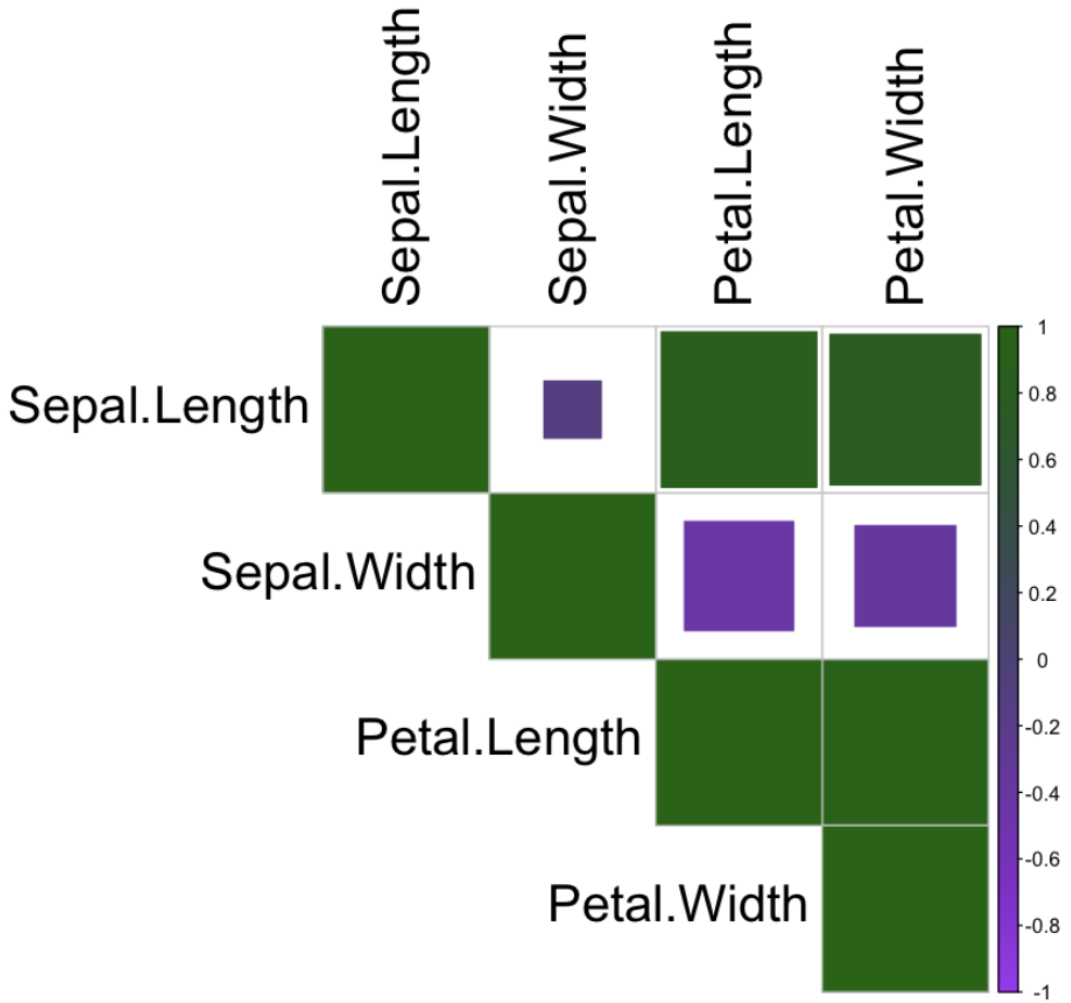
  - Dove:
    - *m* = Matrice di correlazione da visualizzare
    - *Method* = specifica il metodo di visualizzazione: "circle": cerchi."square": quadrati."number": numeri di correlazione."color": mappa di colori. Etc.
    - *Type* = Specifica quale parte della matrice di correlazione visualizzare. "full": visualizza l'intera matrice. "upper": visualizza solo la parte superiore. "lower": visualizza solo la parte inferiore.
    - *addCoef.col* = Colore dei coefficienti di correlazione da aggiungere al grafico
    - *tl.col* = Colore delle etichette
    - *tl.srt* = Angolo di rotazione delle etichette (in gradi)
    - *diag* = Specifica se visualizzare o meno la diagonale della matrice
    - *main* = Titolo principale del grafico



PLOT PER LE CORRELAZIONI

- Creo il plot per le correlazioni delle feature di *Iris*:

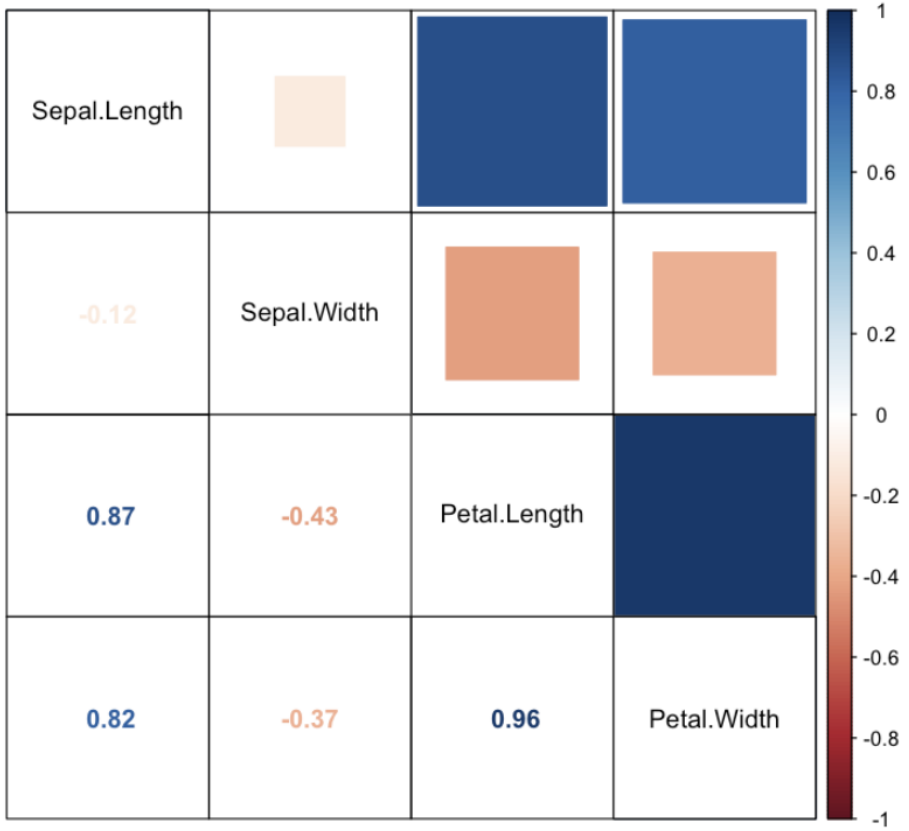
```
corrplot(
 cor(numeric_iris),
 method = "square",
 type = "upper",
 tl.col = "black",
 tl.cex = 2,
 col = colorRampPalette(c("purple", "dark green"))(200))
```



PLOT PER LE CORRELAZIONI

- Creo il plot per le correlazioni delle feature di *Iris*:

```
corrplot.mixed(cor(numeric_iris),
 upper = "square",
 lower = "number",
 addgrid.col = "black",
 tl.col = "black")
```

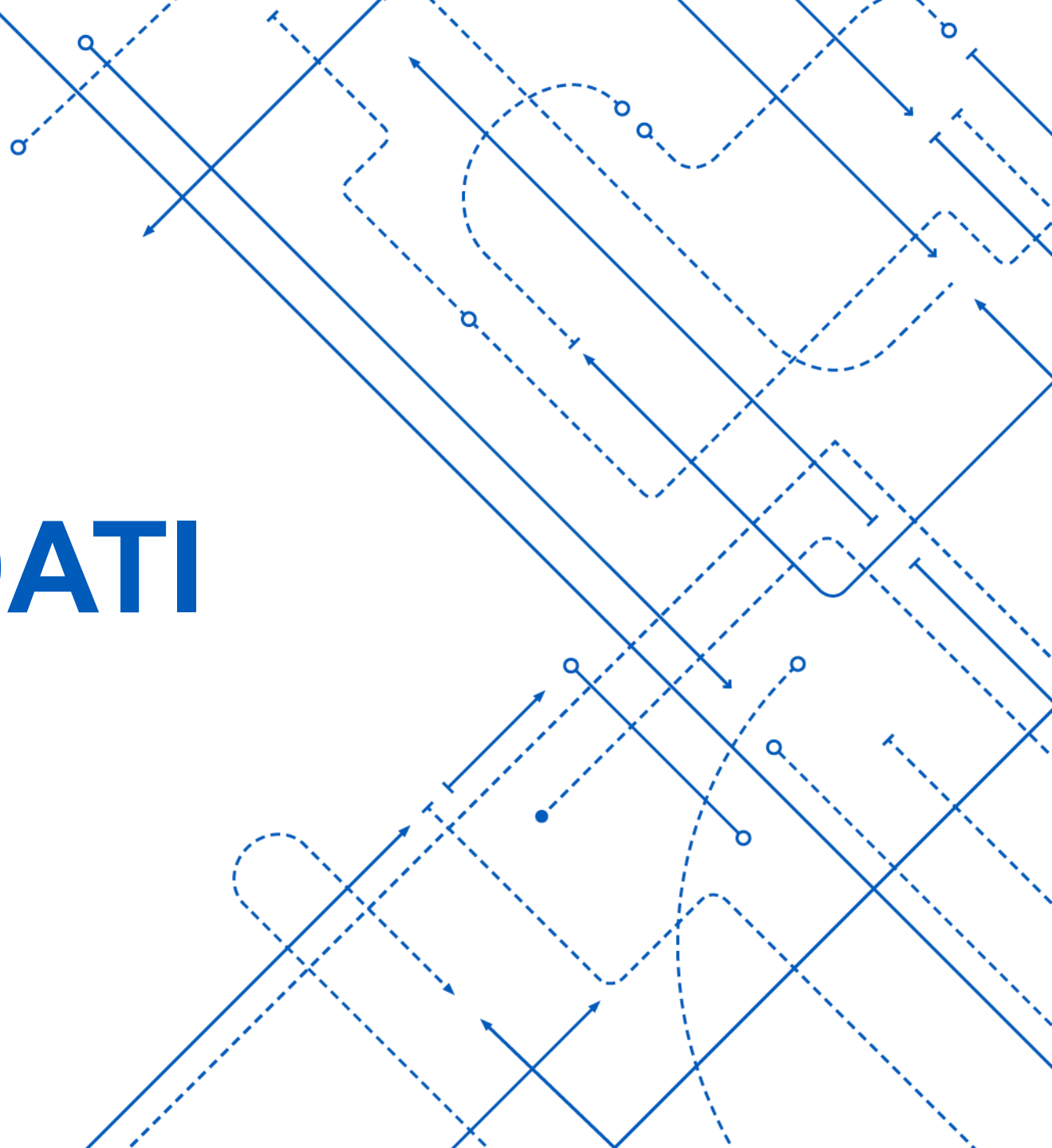


STATISTICA E ANALISI DEI DATI

Capitolo 3 – Mappe Interattive

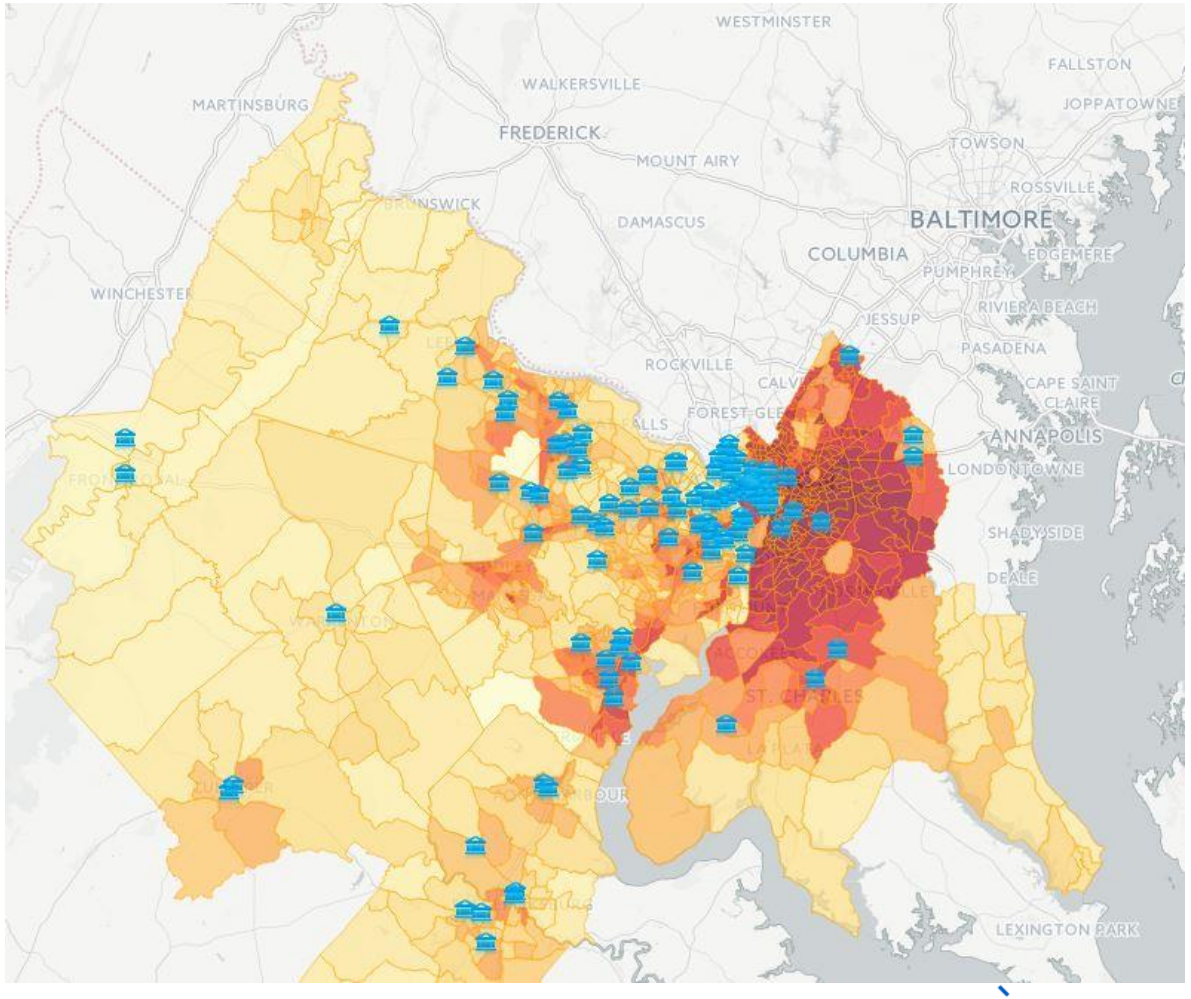
Dott. Stefano Cirillo
Dott. Luigi Di Biasi

a.a. 2025-2026



MAPPE INTERATTIVE

- leaflet è un pacchetto R che consente di creare mappe interattive, facilmente personalizzabili, basato sulla libreria JavaScript Leaflet
    - Ottimo per la visualizzazione di dati geospaziali e la creazione di mappe dinamiche con funzionalità di zoom, etichette, marker e livelli.
- Caratteristiche principali
    - **Facile da usare**: Crea mappe interattive con poche righe di codice.
    - **Altamente personalizzabile**: Aggiungi marker, pop-up, colori personalizzati, livelli sovrapposti, ecc.
    - **Supporto per dati geospaziali**: Compatibile con vari formati di dati GIS
    - **Funzionalità interattive**: Zoom, spostamento, pop-up e tooltip dinamici.



MAPPE INTERATTIVE

- Installazione (Consigliato R Studio):

```
> install.packages('leaflet')
also installing the dependencies ‘colorspace’, ‘lazye
val’, ‘terra’, ‘farver’, ‘labeling’, ‘munsell’, ‘cros
stalk’, ‘leaflet.providers’, ‘png’, ‘raster’, ‘RColor
Brewer’, ‘scales’, ‘sp’, ‘viridisLite’
```

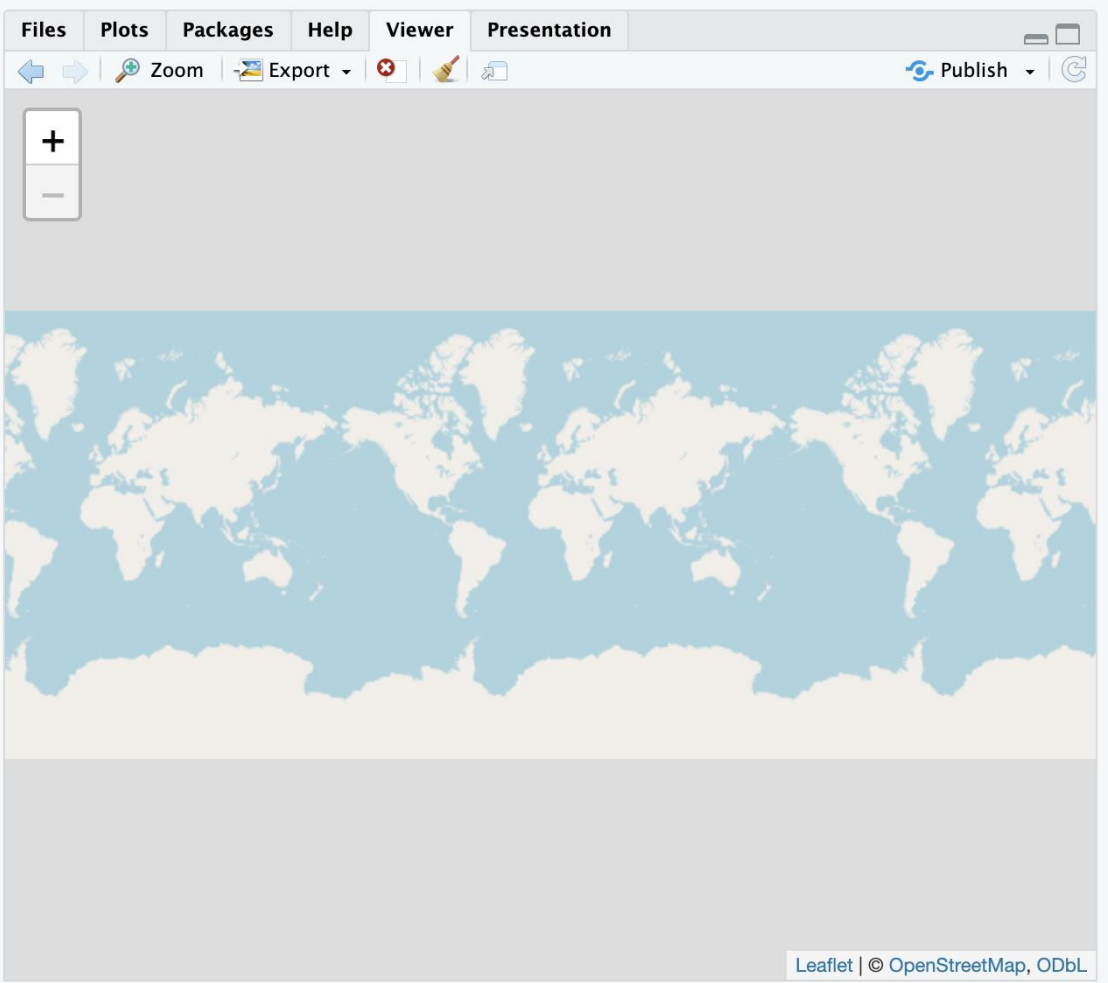
- Caricamento della libreria:

```
> library("leaflet")
```

- Esecuzione della prima mappa:

```
> leaflet() %>% addTiles()
```

  - leaflet(): Crea una nuova mappa
  - addTiles(): Aggiunge una mappa di sfondo

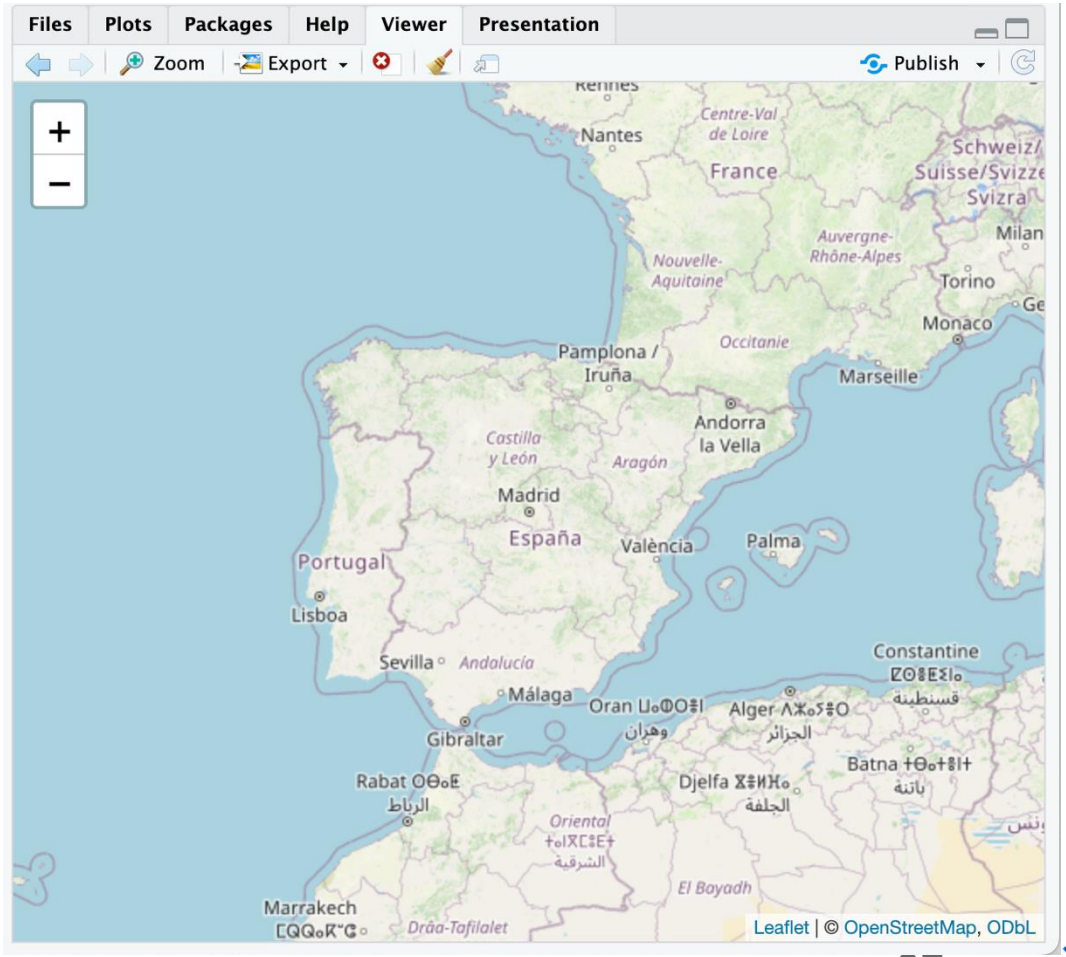


MAPPE INTERATTIVE

- Per posizionare la mappa su uno specifico punto utilizziamo il metodo setView():

```
> leaflet() %>% addTiles() %>% setView(lng = -3.7, lat = 40.4, zoom = 5)
```

  - leaflet(): Crea una nuova mappa
  - addTiles(): Aggiunge una mappa di sfondo
  - setView(): legge le coordinate della località dove visualizzare la mappa



MAPPE INTERATTIVE

- Per posizionare la mappa su uno specifico punto utilizziamo il metodo setView():

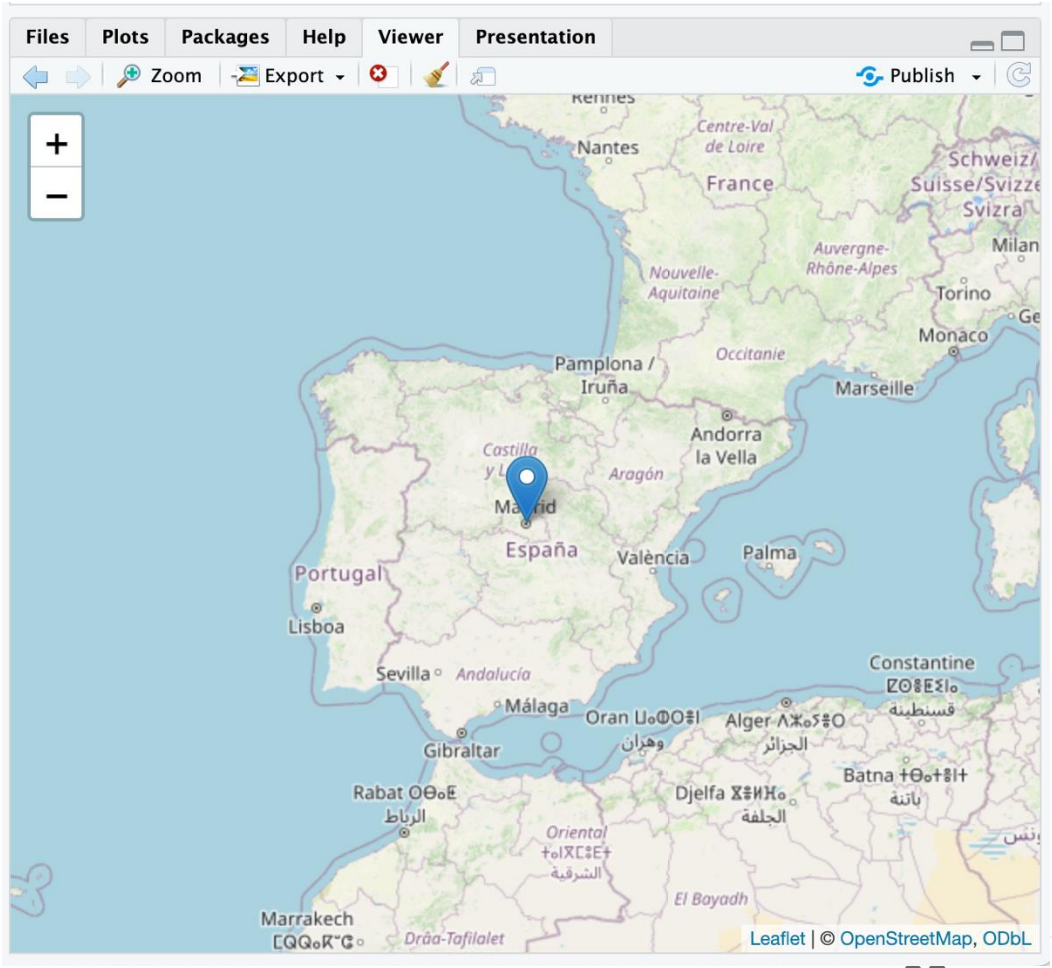
```
> leaflet() %>% addTiles() %>% setView(lng = -3.7, lat = 40.4, zoom = 5)
```

  - leaflet(): Crea una nuova mappa
  - addTiles(): Aggiunge una mappa di sfondo
  - setView(): legge le coordinate della località dove visualizzare la mappa

- È possibile aggiungere un singolo marcatore al grafico passando le coordinate agli argomenti lng(longitudine) e lat(latitudine) della funzione addMarkers

  - La funzione consente un singolo valore o un vettore di valori come input per ogni argomento.

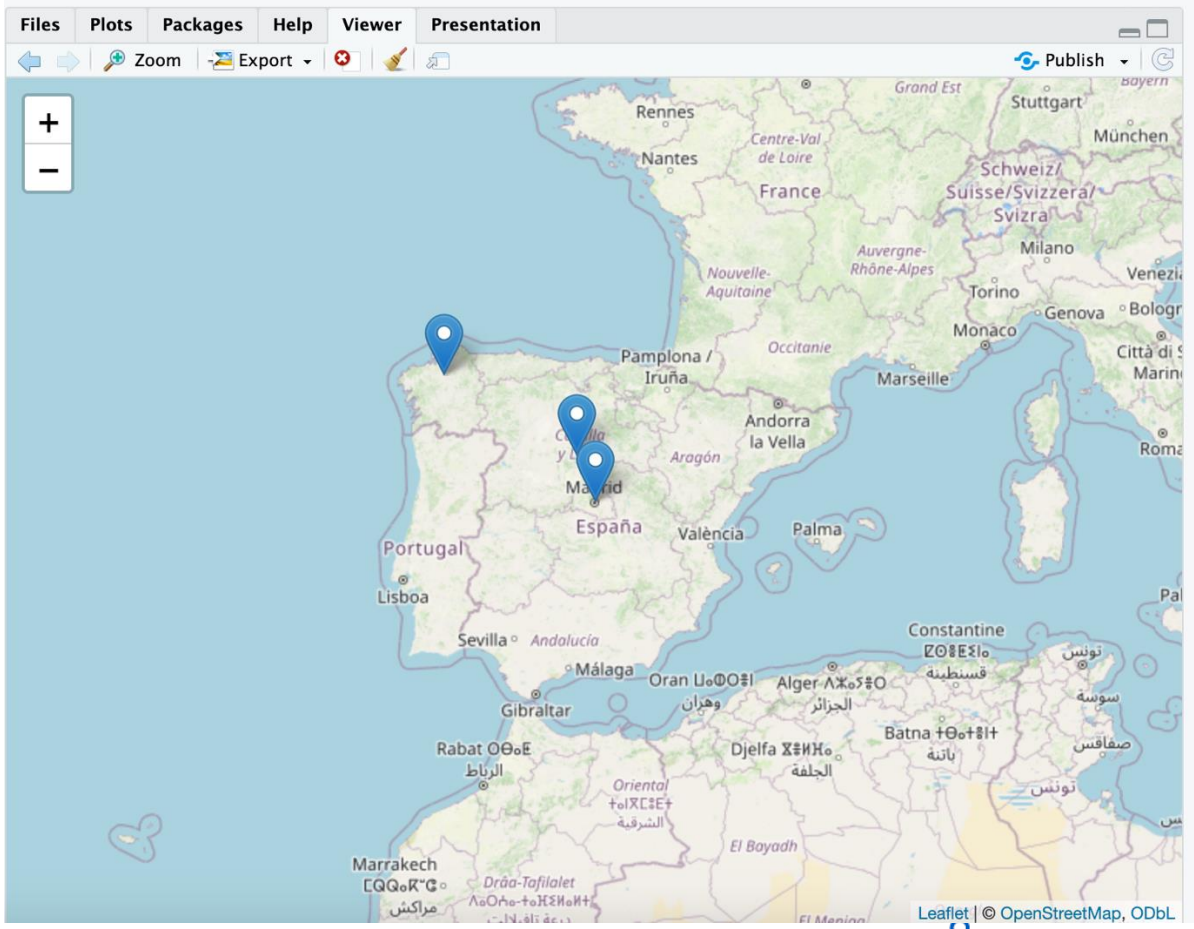
```
> leaflet() %>%
+     addTiles() %>%
+     setView(lng = -3.7, lat = 40.4, zoom = 5) %>%
+     addMarkers(lng = -3.7, lat = 40.4)
```



MARKER

- Aggiungere più marker da un dataframe
- Se i dati che stiamo analizzando provengono da un dataframe puoi passare il dataframe al metodo addMarkers()

```
> df <- data.frame(lng = c(-3.7, -8, -4.2), lat = c(40.4, 43.1, 41.4))
> df
   lng  lat
1 -3.7 40.4
2 -8.0 43.1
3 -4.2 41.4
> leaflet() %>%
+     addTiles() %>%
+     setView(lng = -3.7, lat = 40.4, zoom = 5) %>%
+     addMarkers(data = df)
```



- **Nota:** i nomi devono essere lng e lat

POLYGON E POLYLINE

- **Polygon**:
  - Un **poligono** è una forma geometrica chiusa formata da una sequenza di vertici connessi da segmenti di linea
  - Viene usato per rappresentare aree su una mappa, come **confini geografici**, **regioni** o **aree di interesse**
  - Ogni poligono può avere stili personalizzati come colori di bordo e riempimento
- **Polyline**:
  - Una **polyline** è una serie di segmenti di linea collegati ma non chiusi, spesso utilizzata per rappresentare percorsi, strade, o percorsi di viaggio su una mappa
  - Le polylines possono avere stili per il colore e la larghezza della linea

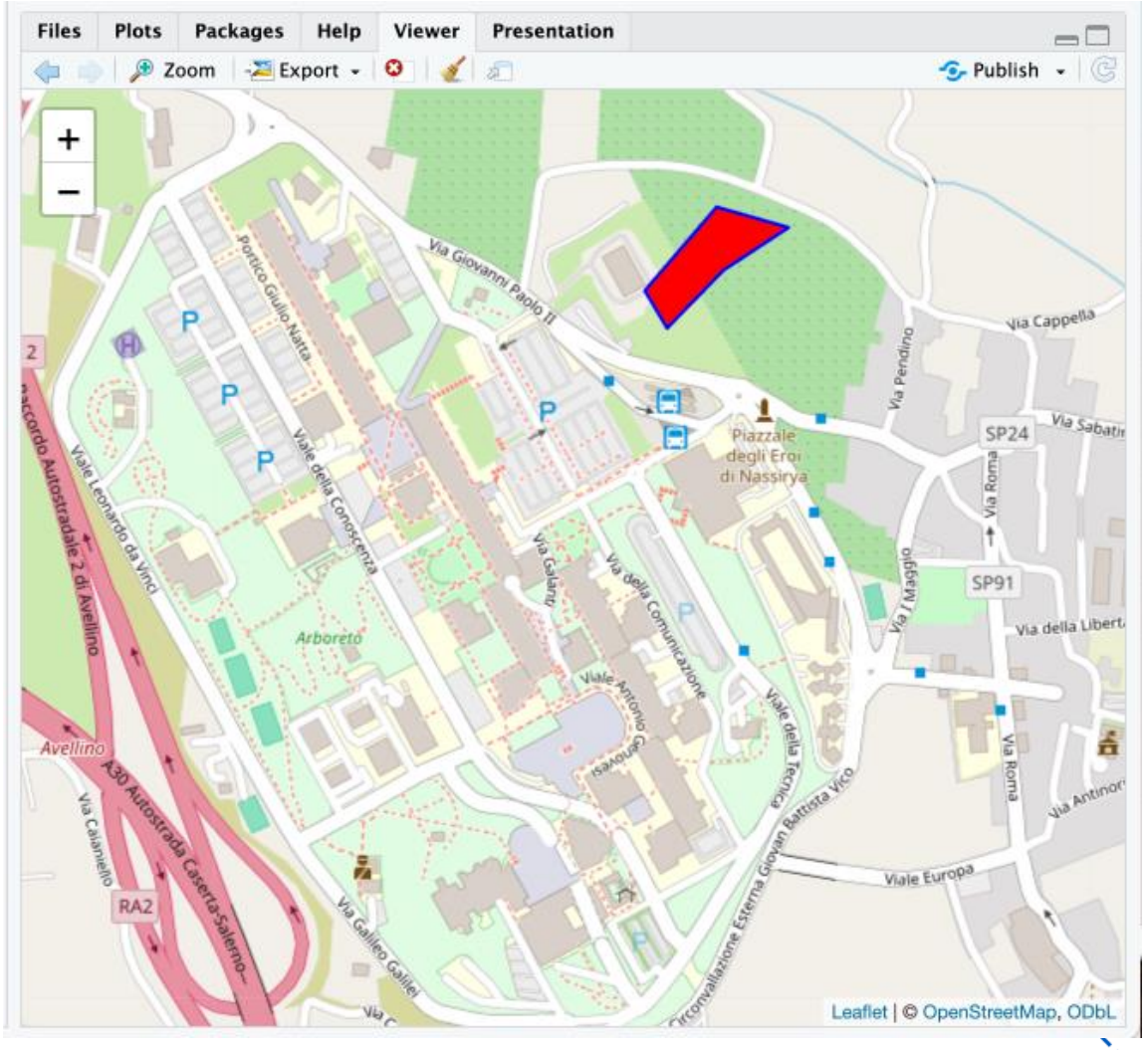
POLYGON E POLYLINE

- Il pacchetto leaflet in R permette di creare mappe interattive con supporto per poligoni e polylines
  - **addPolygons()**: Aggiunge un **poligono** a una mappa Leaflet
  - **addPolylines()**: Aggiunge una **polyline** a una mappa Leaflet
- È possibile personalizzare l'aspetto di poligoni e polylines utilizzando diversi parametri:
  - lat, lng: Coordinate dei vertici del poligono (vettori numerici).
  - color: Colore del bordo del poligono (es: "blue").
  - weight: Spessore del bordo del poligono.
  - opacity: Trasparenza del bordo del poligono (0 - 1).
  - fillColor: Colore di riempimento del poligono.
  - fillOpacity: Trasparenza del riempimento (0 - 1).
  - popup: Testo che appare quando si clicca sul poligono.
  - label: Etichetta che appare al passaggio del mouse sul poligono

POLYGON E POLYLINE

- Esempio di Poligono : Università di Salerno

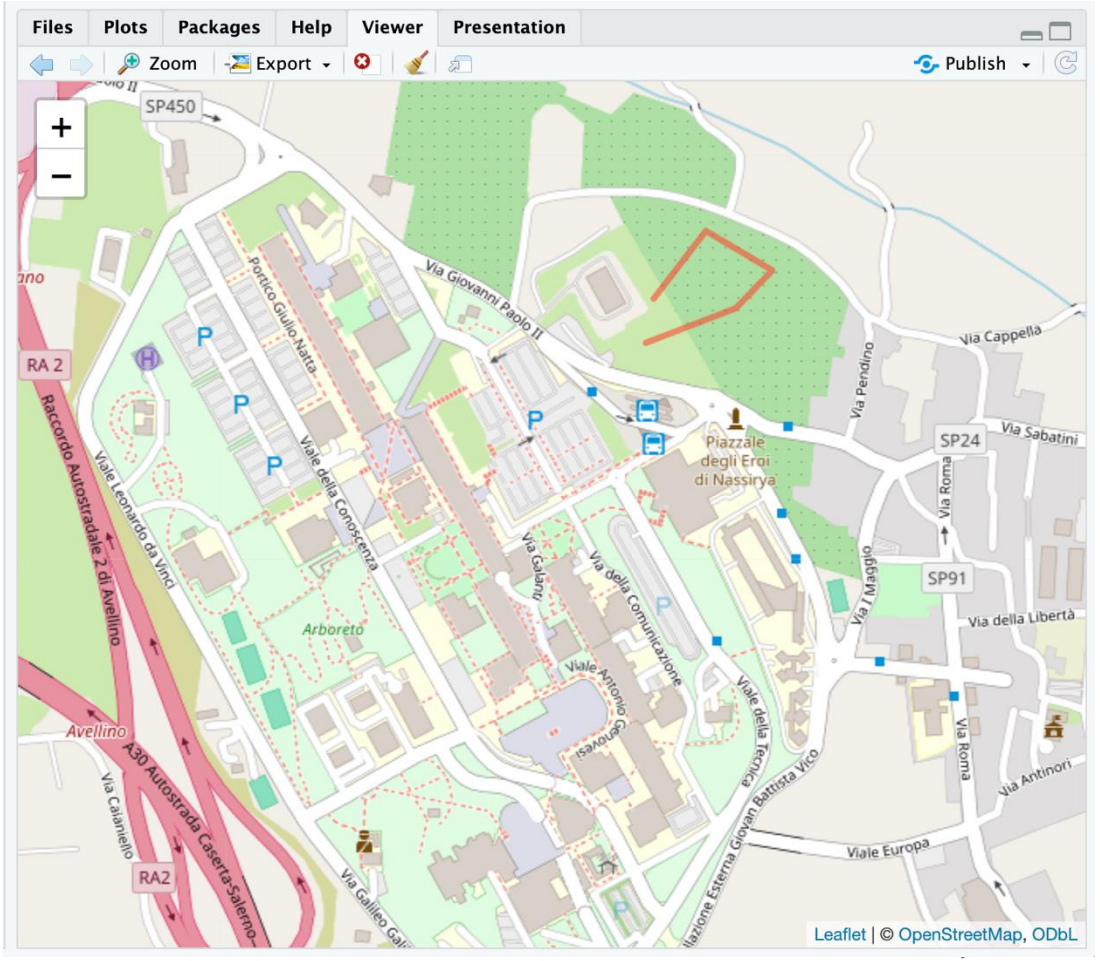
```
> library(leaflet)
>
> # Definizione dei vertici del poligono
> coords <- list(
+   c(40.77445, 14.79328),  # Vertice 1
+   c(40.77500, 14.79400),  # Vertice 2
+   c(40.77540, 14.79480),  # Vertice 3
+   c(40.77560, 14.79390),  # Vertice 4
+   c(40.77480, 14.79300)   # Vertice 5
+ )
> leaflet() %>%
+     addTiles() %>%
+     addPolygons(lng = sapply(coords, "[", 2),
+                 lat = sapply(coords, "[", 1),
+                 color = "blue", fillColor = "red",
+                 weight = 2, opacity=1,
+                 fillOpacity = 1)
```



POLYGON E POLYLINE

- Esempio di Polylines: Università di Salerno

```
> # Definizione dei punti della polyline
> line_coords <- list(
+     c(40.77445, 14.79328),  # Punto 1: Ingresso Università di Salerno
+     c(40.77480, 14.79450),  # Punto 2: Punto intermedio
+     c(40.77520, 14.79500),  # Punto 3: Punto intermedio
+     c(40.77560, 14.79410),  # Punto 4: Punto intermedio
+     c(40.77490, 14.79340)   # Punto 5: Punto finale
+ )
>
> # Creazione della mappa con polyline
> leaflet() %>%
+     addTiles() %>%
+     addPolylines(lng = sapply(line_coords, "[", 2),
+                  lat = sapply(line_coords, "[", 1),
+                  color = "red", weight = 4)
```



MANIPOLARE DATI GEOSPAZIALI

- **sf** (Simple Features) è una libreria R che permette di gestire e analizzare dati geospaziali in modo efficiente
- Implementa lo standard Simple Features per rappresentare oggetti geospaziali, come:
  - Punti (Points)
  - Linee (Lines)
  - Poligoni (Polygons)
- **Funzionalità principali:**
  - Lettura e scrittura di shapefile: Supporto per formati geospaziali comuni, come Shapefile e GeoJSON
  - Proiezioni geografiche: Trasformazione e manipolazione delle coordinate geografiche (CRS - Coordinate Reference System)
  - Geometrie e attributi: Integra geometrie in dataframe semplificando l'analisi dei dati geografici

MANIPOLARE DATI GEOSPAZIALI

- Esempio: Rappresentazione della mappa di Alicante

```
> library(leaflet)
> library(sf)
Linking to GEOS 3.11.0, GDAL 3.5.3, PROJ 9.1.0; sf_use_s2() is TRUE
>
> # Read a Geojson or shapefile
> data_map <- read_sf("https://raw.githubusercontent.com/R-CoderDotCom/data/main/sample_geojson.geojson")
>
> # Transform to leaflet projection if needed
> data_map <- st_transform(data_map, crs = '+proj=longlat +datum=WGS84')
>
> leaflet() %>%
+     addTiles() %>%
+     setView(lng = -0.49, lat = 38.34, zoom = 14) %>%
+     addPolygons(data = data_map, color = "blue", stroke = 1, opacity = 0.8)
```

- La funzione read_sf() carica un file GeoJSON da un URL e lo converte in un oggetto sf, che è una rappresentazione spaziale di tipo Simple Features in R.
  - Nell'esempio viene letto un file GeoJSON che contiene dati geospaziali da un URL pubblico
    - I dati geospaziali sono pubblici in rete o possono essere creati con piattaforme come:

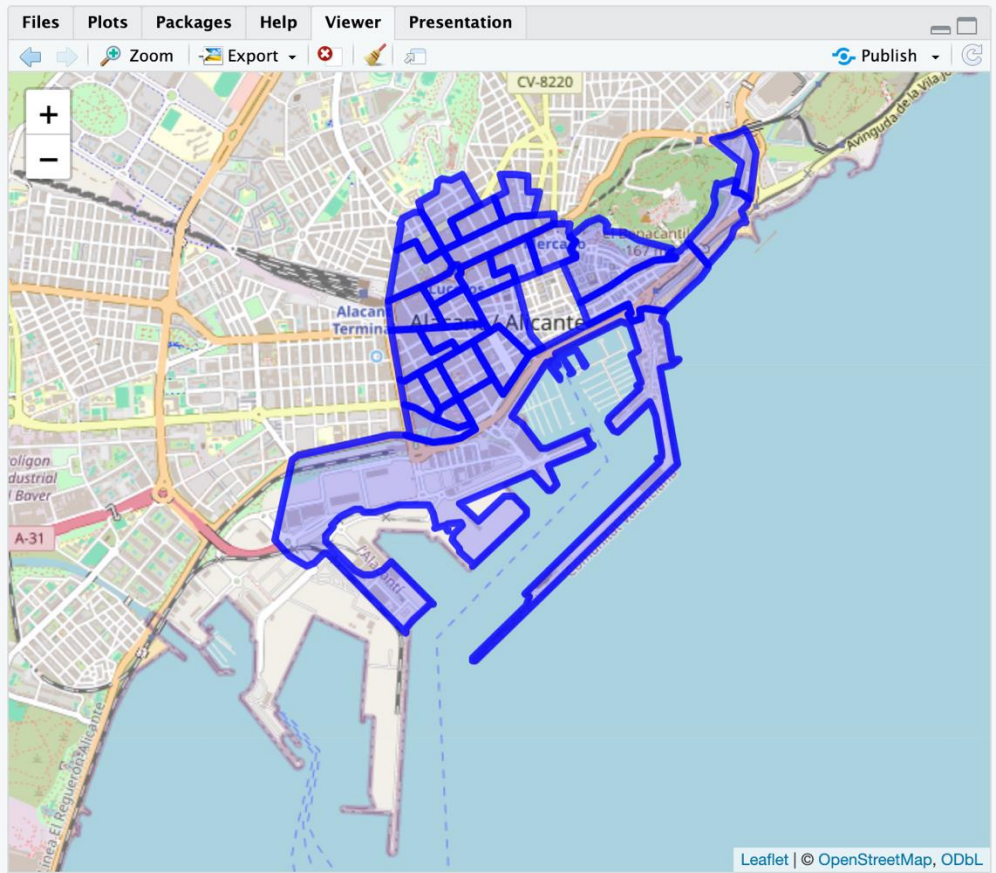
      GeoJson: https://geojson.io/

MANIPOLARE DATI GEOSPAZIALI

- Esempio: Rappresentazione della mappa di Alicante

```
> library(leaflet)
> library(sf)
Linking to GEOS 3.11.0, GDAL 3.5.3, PROJ 9.1.0; sf_use_s2() is TRUE
>
> # Read a Geojson or shapefile
> data_map <- read_sf("https://raw.githubusercontent.com/R-CoderDotCom/data/main/sample_geojson.geojson")
>
> # Transform to leaflet projection if needed
> data_map <- st_transform(data_map, crs = '+proj=longlat +datum=WGS84')
>
> leaflet() %>%
+     addTiles() %>%
+     setView(lng = -0.49, lat = 38.34, zoom = 14) %>%
+     addPolygons(data = data_map, color = "blue", stroke = 1, opacity = 0.8)
```

- La funzione st_transform() trasforma il sistema di riferimento delle coordinate (CRS) dei dati geospaziali



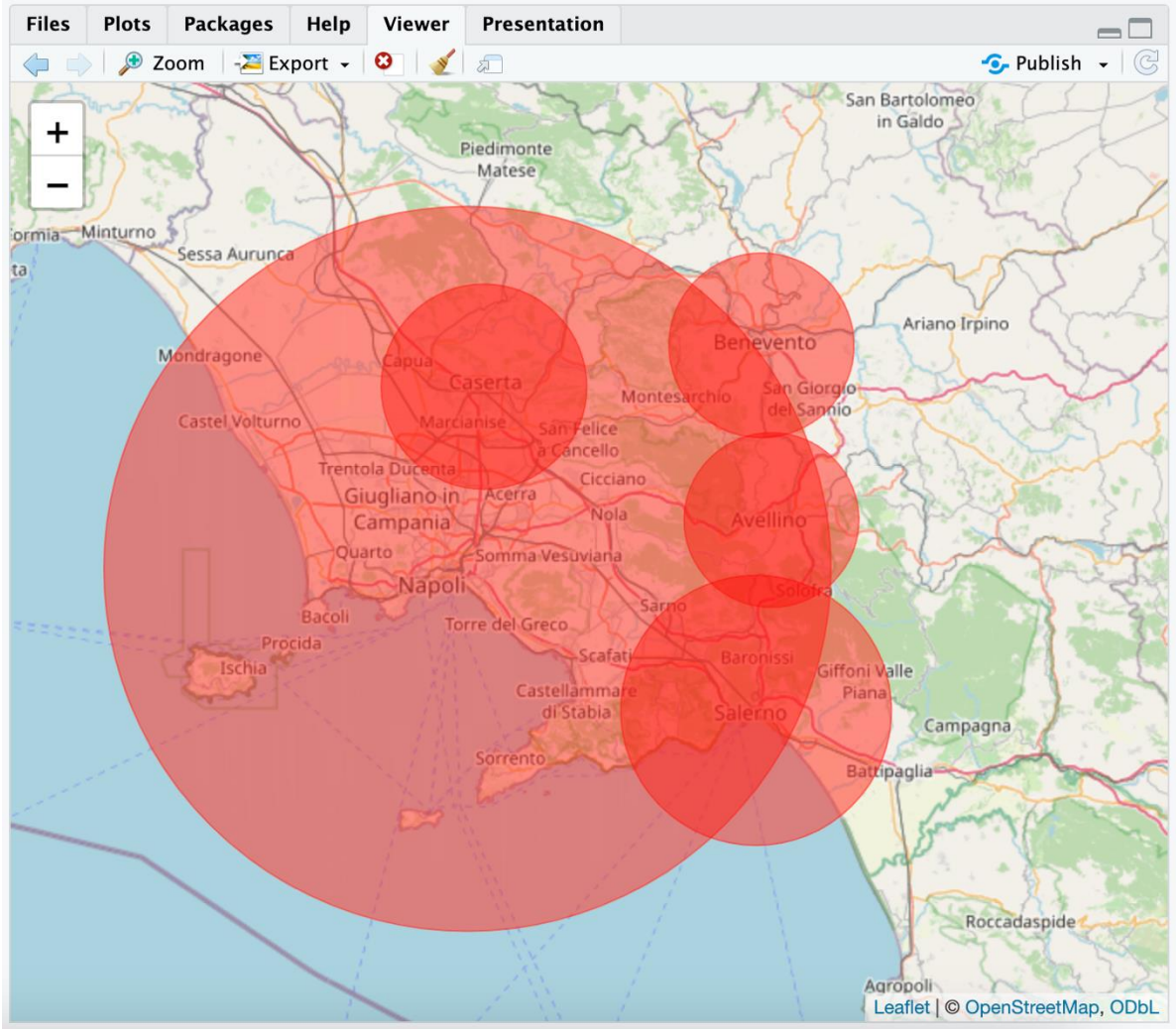
LEAFLET IN STATISTICA

- **Leaflet** è una libreria per creare mappe interattive, utile per la statistica geospaziale.
- **Vantaggi nell'uso di Leaflet per l'analisi statistica:**
  - **Visualizzazione geospaziale**: Permette di rappresentare dati statistici su mappe per un'analisi visiva più efficace.
  - **Interattività**: Le mappe possono essere zoomate, esplorate e arricchite con tooltip e pop-up, facilitando l'interpretazione dei dati.
- **Supporto per diversi tipi di dati georeferenziati**:
  - **Punti** (ad es. dati di localizzazione)
  - **Poligoni** (ad es. aree geografiche, distretti)
  - **Linee** (ad es. percorsi, confini)
- **Esempi di applicazione in statistica:**
  - **Mappe di densità**: Visualizzazione della densità di eventi o punti di interesse.
  - **Cerchi proporzionali**: Rappresentazione visiva di variabili quantitative (es. popolazione).
  - **Mappe tematiche**: Colorazione di aree geografiche in base a una variabile statistica (es. tasso di disoccupazione).
  - **Clustering di dati geolocalizzati**: Raggruppamento di punti in base alla loro densità.

LEAFLET IN STATISTICA

- Visualizziamo dati statistici su una mappa utilizzando cerchi proporzionali per rappresentare la grandezza dei valori di un dataset (ad es., popolazione per città)
    - Consideriamo le province Campane Salerno, Napoli, Avellino, Caserta e Benevento

```
> library(leaflet)
>
> # Dati fittizi: coordinate delle città e popolazione
> city_data <- data.frame(
+     city = c("Salerno", "Napoli", "Avellino", "Caserta", "Benevento"),
+     lat = c(40.6824, 40.8533, 40.9140, 41.0757, 41.1256),
+     lng = c(14.7681, 14.3056, 14.7928, 14.3321, 14.7768),
+     population = c(133970, 962003, 54877, 75930, 61231)
+ )
>
> # Mappa con cerchi proporzionali in base alla popolazione
> leaflet(city_data) %>%
+     addTiles() %>%
+     addCircles(
+         lng = ~lng, lat = ~lat,
+         # Raggio proporzionale alla radice quadrata della popolazione
+         weight = 1, radius = ~sqrt(population) * 50,
+         popup = ~paste(city, "<br>Popolazione:", population),
+         color = "red", fillColor = "red", fillOpacity = 0.5
```



DOMANDE?