

# STATISTICA E ANALISI DEI DATI

Capitolo 2 – Statistica Descrittiva

---

Dott. Stefano Cirillo  
Dott. Luigi Di Biasi

a.a. 2025-2026

# STATISTICA

---

## LA STATISTICA NELLA RICERCA

Raccolta dei dati

Elaborazione

Descrizione

Una raccolta di dati non corretta, una loro presentazione inadeguata o un'analisi statistica non appropriata rendono impossibile la verifica dei risultati da parte di altri studiosi e il confronto con altre ricerche e analisi del settore

# STATISTICA

---

## LA STATISTICA NELLA RICERCA

### Descrittiva

- Si occupa della presentazione e sintesi dei dati

### Inferenziale

- Permette di trasferire le informazioni ottenute su un campione all'intera popolazione

La **variabile** è ciò che viene **osservato** o **misurato** e può assumere uno tra una serie definita di possibili **valori**

# STATISTICA DESCRITTIVA

---

- Cos'è la **statistica descrittiva**?

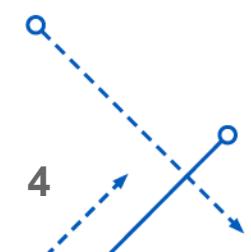
- Il mondo è pieno di **fenomeni osservabili**!
- In genere, siamo in grado di analizzare questi fenomeni **mediante l'osservazione del comportamento** dalle **caratteristiche «visibili (osservabili appunto)»** di un generico fenomeno.
  - Come **osserviamo «come si comportano» le caratteristiche** di un fenomeno?

Possiamo  
**misurare/contare/osser**vare i  
valori assunti da X



Possiamo associare una variabile ad ogni caratteristica osservabile di un fenomeno (**ad esempio ... X**)

In statistica descrittiva **le variabili** possono essere di **tre tipi**: Qualitative, Quantitative, Ordinabili



# STATISTICA DESCRITTIVA

---

- Cos'è la **statistica descrittiva**?

- è una branca della statistica che si concentra sulla **raccolta, sull'organizzazione, sull'analisi e sulla presentazione dei dati** in modo da **riassumere e descrivere le principali caratteristiche** di un insieme di dati.

- **Principali tecniche:**

L'obiettivo principale della statistica descrittiva è quello di **fornire una comprensione chiara e sintetica** dei dati

- Misure di centralità (media campionaria, mediana, moda ...);
- Misure di dispersione (varianza, deviazione standard...);
- Grafici e tavelle;
- Misura della forma della distribuzione;
- Percentili e quantili;
- Frequenze e percentuali;

# STATISTICA DESCRITTIVA

---

- Cos'è la **statistica descrittiva**?

- è una branca della statistica che si concentra sulla **raccolta, sull'organizzazione, sull'analisi e sulla presentazione dei dati** in modo da **riassumere e descrivere le principali caratteristiche** di un insieme di dati.

- **Principali tecniche:**

L'obiettivo principale della statistica descrittiva è quello di **fornire una comprensione chiara e sintetica** dei dati

- Misure di centralità (media campionaria, mediana, moda ...);
- Misure di dispersione (varianza, deviazione standard...);

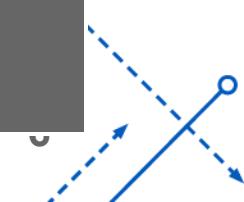
- Grafici e tavelle;



- Misura della forma della distribuzione;
- Percentili e quantili;
- Frequenze e percentuali;

PARTE 1: Rappresentazione di dati in contenuti in:

- Vettori;
- Fattori;
- Matrici;
- Data frame;
- Serie temporali (serie storiche).



# STATISTICA DESCRITTIVA

---

- Cos'è la **statistica descrittiva**?

- è una branca della statistica che si concentra sulla **raccolta, sull'organizzazione, sull'analisi e sulla presentazione dei dati** in modo da **riassumere e descrivere le principali caratteristiche** di un insieme di dati.

- **Principali tecniche:**

L'obiettivo principale della statistica descrittiva è quello di **fornire una comprensione chiara e sintetica** dei dati

- Misure di centralità (media campionaria, mediana, moda ...);
- Misure di dispersione (varianza, deviazione standard...);

- Grafici e tavelle;



- Misura della forma della distribuzione;

- Percentili e quantili;

- Frequenze e percentuali;

PARTE 2:

Costruzione di tavelle di frequenza e grafici di alta qualità.

# STATISTICA E ANALISI DEI DATI

Capitolo 2 – Grafici in R

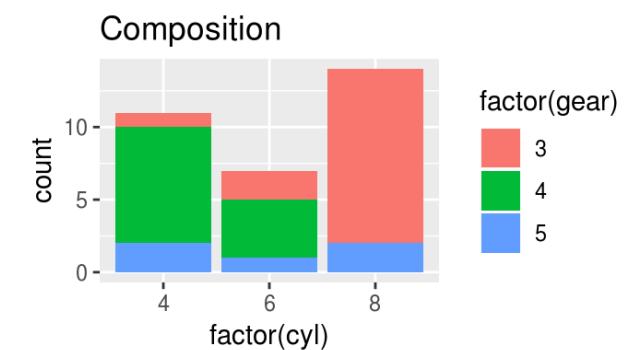
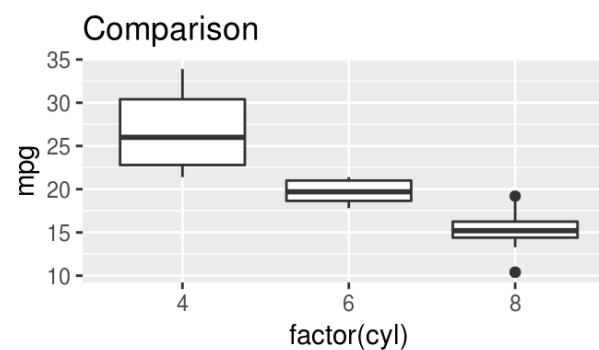
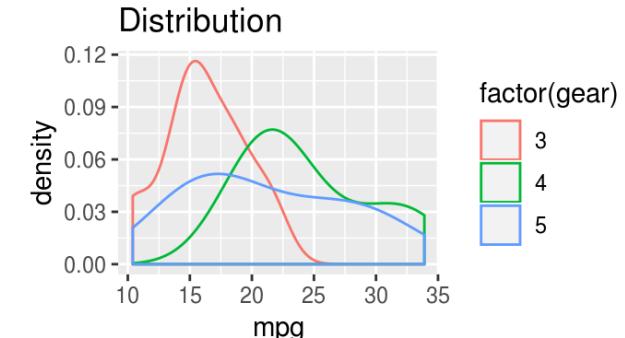
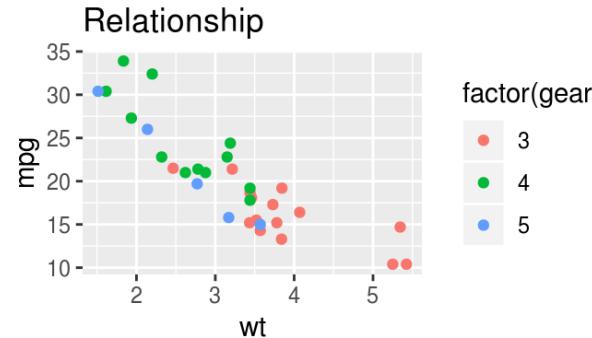
---

Dott. Stefano Cirillo  
Dott. Luigi Di Biasi

a.a. 2025-2026

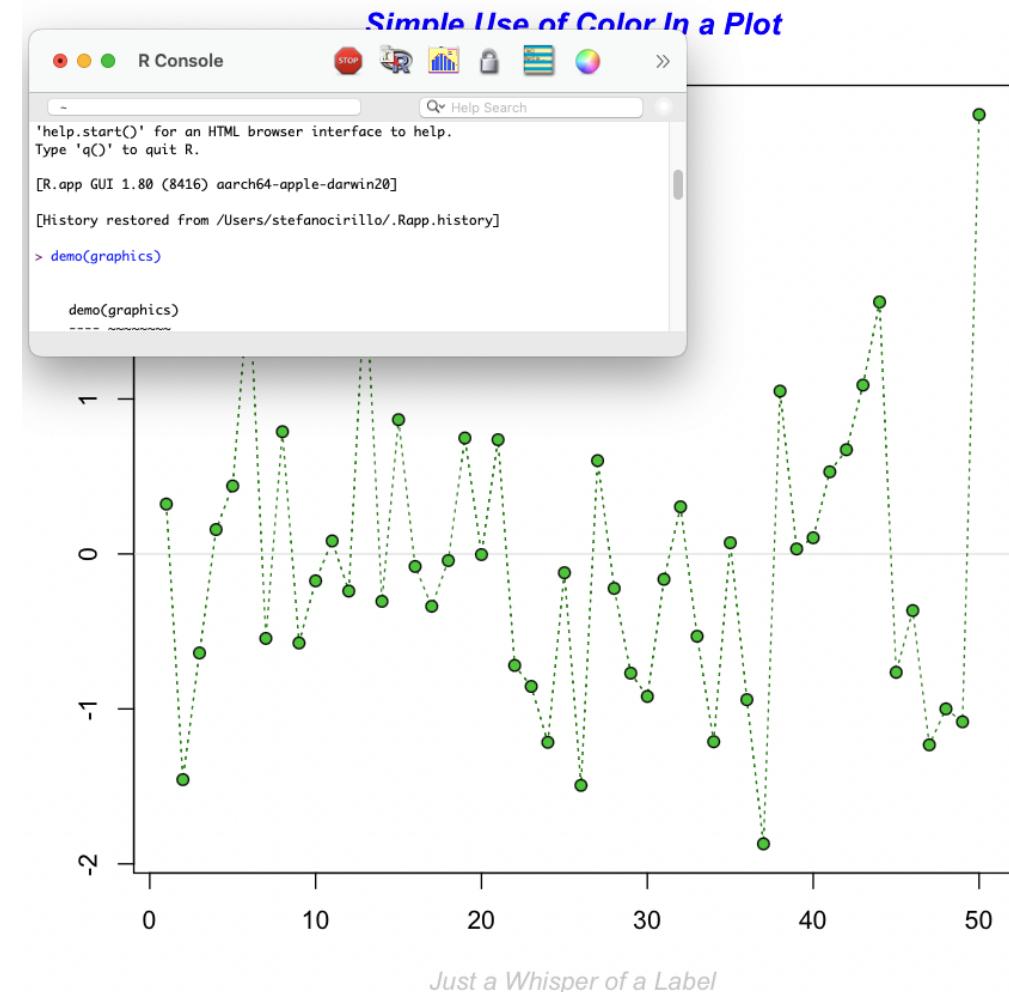
# L'AMBIENTE GRAFICO R

- R è dotato di un sofisticato **ambiente grafico** che permette di creare grafici per illustrare i risultati di elaborazioni statistiche
  - Esso permette di impostare parametri per modificare ogni aspetto di un grafico, simboli, colori ed esportare nei formati più comuni, sia **vettoriali** (quali .ps, .pdf) che **bitmap** (.bmp, .jpg)



# L'AMBIENTE GRAFICO R

- Per avere un'idea delle potenzialità offerte dall'ambiente grafico di R, basta visualizzare la **dimostrazione riassuntiva** che può essere ottenuta digitando il comando `demo(graphics)` e digitando ripetutamente il tasto “Invio” per visualizzare delle immagini in successione.
- Noi abbiamo visto tanti oggetti gestiti da R
  - Ognuno di essi è utile per rappresentare una particolare caratteristica o una vista sui dati del fenomeno che stiamo analizzando. Per ognuno di essi, vediamo come R permette di visualizzare graficamente i dati contenuti.



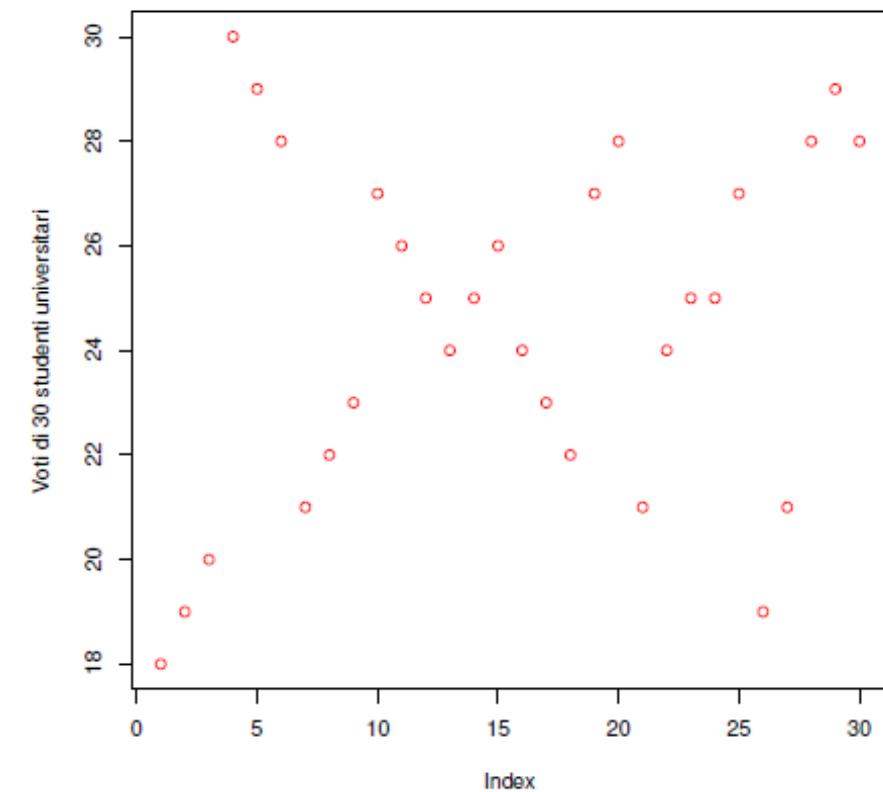
10

# GRAFICI PER VETTORI NUMERICI

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

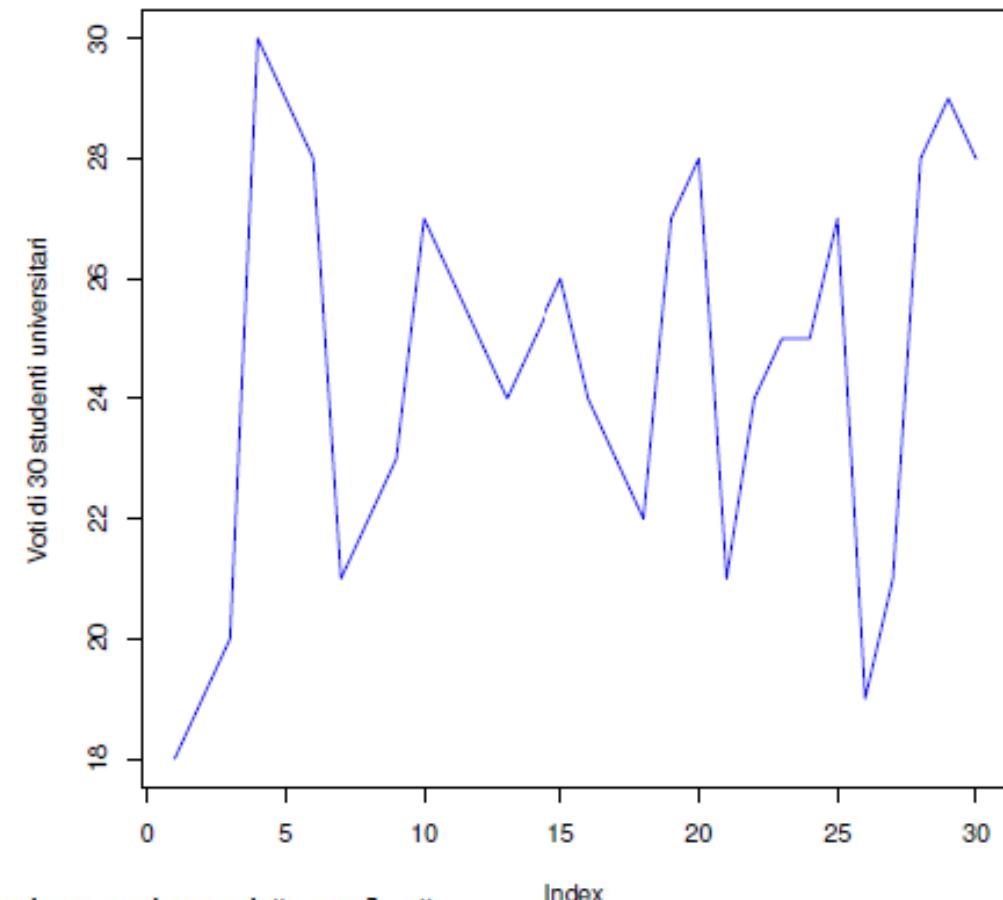
```
> voti<-c(18,19,20,30,29,28,21,22,23,27,26,25,24,25,  
+ 26,24,23,22,27,28,21,24,25,25,27,19,21,28,29,28)  
>  
> plot(voti,ylab="Voti di 30 studenti universitari",col="red")
```

- ...otteniamo un grafico che illustra il voto riportato da ognuno dei 30 studenti individuati dalle loro posizioni nel vettore

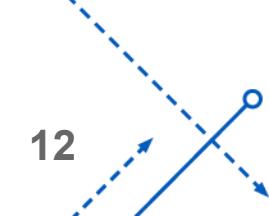


# GRAFICI PER VETTORI NUMERICI

- Possiamo connettere i punti mediante linee per creare una **serie storica** dei voti
- A tal fine, utilizziamo la funzione **plot()** con l'opzione **type = l**, che permette di creare delle linee interconnesse.



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

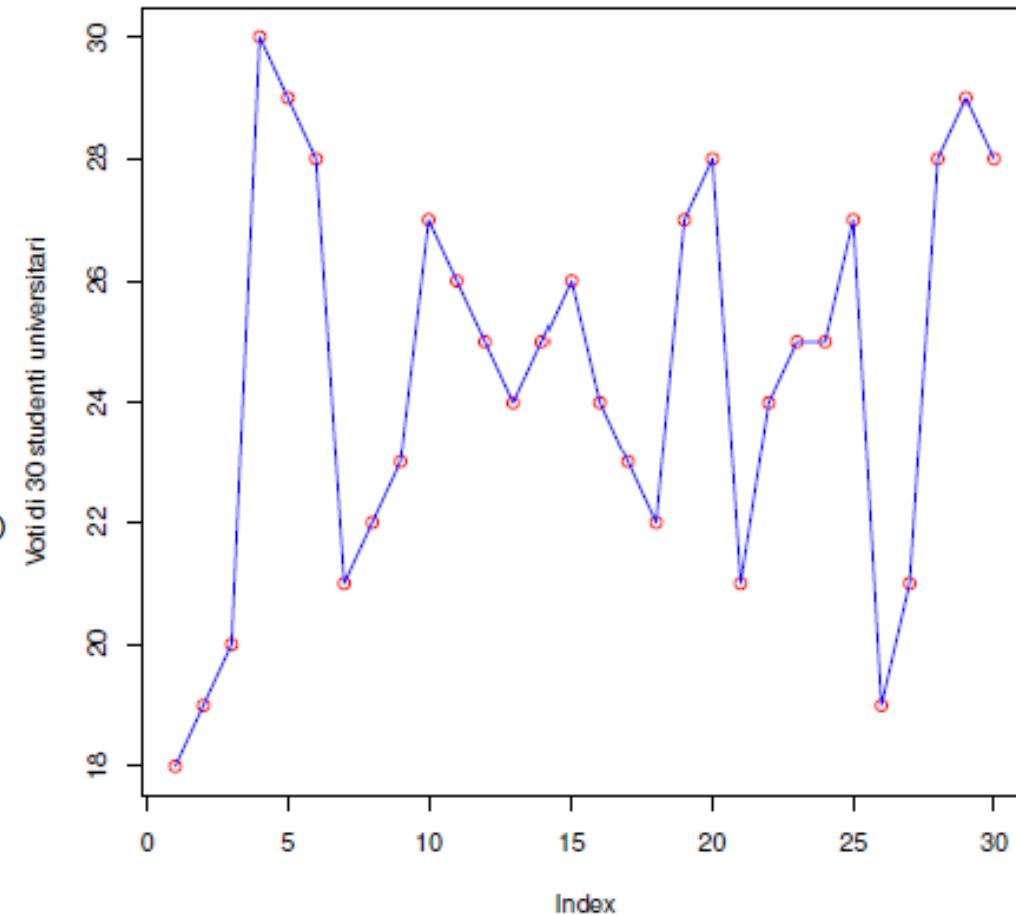


# GRAFICI PER VETTORI NUMERICI

- Possiamo anche procedere in modo diverso e utilizzare il comando **lines()**
  - Il comando **lines()** in R viene utilizzato per aggiungere linee a un grafico esistente
- È particolarmente utile quando vuoi sovrapporre linee a un grafico precedentemente disegnato con funzioni come **plot()**

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

- Nel codice sopra:
  - **plot()** crea il grafico iniziale con dei punti (type = "p").
  - **lines()** aggiunge una linea continua rossa (type = "l") che segue i punti precedentemente tracciati



# GRAFICI PER VETTORI NUMERICI

- Consideriamo un esempio più complesso in cui partendo da un dataset reale vogliamo filtrare i dati e rappresentare il loro andamento storico
- Consideriamo il dataset fornito da USA social security administration (<https://github.com/hadley/babynames>)
  - Vogliamo rappresentare come è cambiato l'utilizzo dei nomi 'Alice', 'Mary' e 'Mae' nei bambini nati tra il 1880 ed il 2020

```
install.packages("babynames")  
  
library(stats)  
library(babynames)  
library(dplyr)
```

babynames				
A tibble: 1924665 × 5				
year	sex	name	n	prop
<dbl>	<chr>	<chr>	<int>	<dbl>
1880	F	Mary	7065	0.07238359
1880	F	Anna	2604	0.02667896
1880	F	Emma	2003	0.02052149
1880	F	Elizabeth	1939	0.01986579
1880	F	Minnie	1746	0.01788843
1880	F	Margaret	1578	0.01616720

<https://www.delftstack.com/it/howto/r/add-line-to-plot-in-r/>

# GRAFICI PER VETTORI NUMERICI

- Vogliamo rappresentare come è cambiato l'utilizzo dei nomi 'Alice', 'Mary' e 'Mae' nei bambini nati tra il 1880 ed il 2020

```
install.packages("babynames")
```

```
library(stats)
library(babynames)
library(dplyr)
```

```
babynames
```

A tibble: 1924665 × 5				
year	sex	name	n	prop
<dbl>	<chr>	<chr>	<int>	<dbl>
1880	F	Mary	7065	0.07238359
1880	F	Anna	2604	0.02667896
1880	F	Emma	2003	0.02052149
1880	F	Elizabeth	1939	0.01986579
1880	F	Minnie	1746	0.01788843
1880	F	Margaret	1578	0.01616720

<https://www.delftstack.com/it/howto/r/add-line-to-plot-in-r/>

- Step 1: Installare la libreria dei dataset

- Step 2: Importare le librerie necessarie per l'analisi

- La libreria **stats** è parte delle librerie base di R e fornisce un insieme di funzioni statistiche per analisi dati.
- Contiene strumenti per eseguire test statistici, calcolare distribuzioni, regressioni, clustering, analisi delle serie temporali, e altro.
- La libreria **babynames** contiene un dataset storico di nomi di bambini negli Stati Uniti.
  - Il dataset è derivato dalle statistiche della Social Security Administration (SSA) e include dati dal 1880 al presente
- La libreria **dplyr** è uno dei pacchetti centrali del **tidyverse**, e fornisce un set di strumenti per manipolare dati in modo efficiente.
  - Consente di eseguire operazioni di trasformazione, filtraggio, raggruppamento e sintesi di dataset con una sintassi intuitiva.

# GRAFICI PER VETTORI NUMERICI

- Vogliamo rappresentare come è cambiato l'utilizzo dei nomi 'Alice', 'Mary' e 'Mae' nei bambini nati tra il 1880 ed il 2020

babynames				
A tibble: 1924665 × 5				
year	sex	name	n	prop
1880	F	Mary	7065	0.07238359
1880	F	Anna	2604	0.02667896
1880	F	Emma	2003	0.02052149
1880	F	Elizabeth	1939	0.01986579
1880	F	Minnie	1746	0.01788843
1880	F	Margaret	1578	0.01616720

- Step 3: Analizzare le informazioni/dati a nostra disposizione

- Quali sono le informazioni?
- Quanti sono i dati (osservazioni) a nostra disposizione?
- Di che tipo sono le informazioni?
- Quali dati mi interessano per l'analisi?

# GRAFICI PER VETTORI NUMERICI

- Vogliamo rappresentare come è cambiato l'utilizzo dei nomi 'Alice', 'Mary' e 'Mae' nei bambini nati tra il 1880 ed il 2020

```
dat <- babynames %>% filter(name %in% c("Alice")) %>% filter(sex=="F")
dat2 <- babynames %>% filter(name %in% c("Mary")) %>% filter(sex=="F")
dat3 <- babynames %>% filter(name %in% c("Mae")) %>% filter(sex=="F")
head(dat)
head(dat2)
head(dat3)
```

			year	sex	name	n	prop
			<dbl>	<chr>	<chr>	<int>	<dbl>
			1880	F	Mary	7065	0.07238359
			1881	F	Mary	6919	0.069999140
1880	F	Alice	1882	F			
1881	F	Alice	1883	F			
1882	F	Alice	1884	F			
1883	F	Alice	1885	F			
1884	F	Alice	1732	0.01258			
1885	F	Alice	1681	0.01184			

A tibble: 6 × 5

			year	sex	name	n	prop
			<dbl>	<chr>	<chr>	<int>	<dbl>
			1880	F	Mae	344	0.00352441
			1881	F	Mae	314	0.00317637
			1882	F	Mae	406	0.00350923
			1883	F	Mae	423	0.00352327
			1884	F	Mae	542	0.00393935
			1885	F	Mae	578	0.00407191

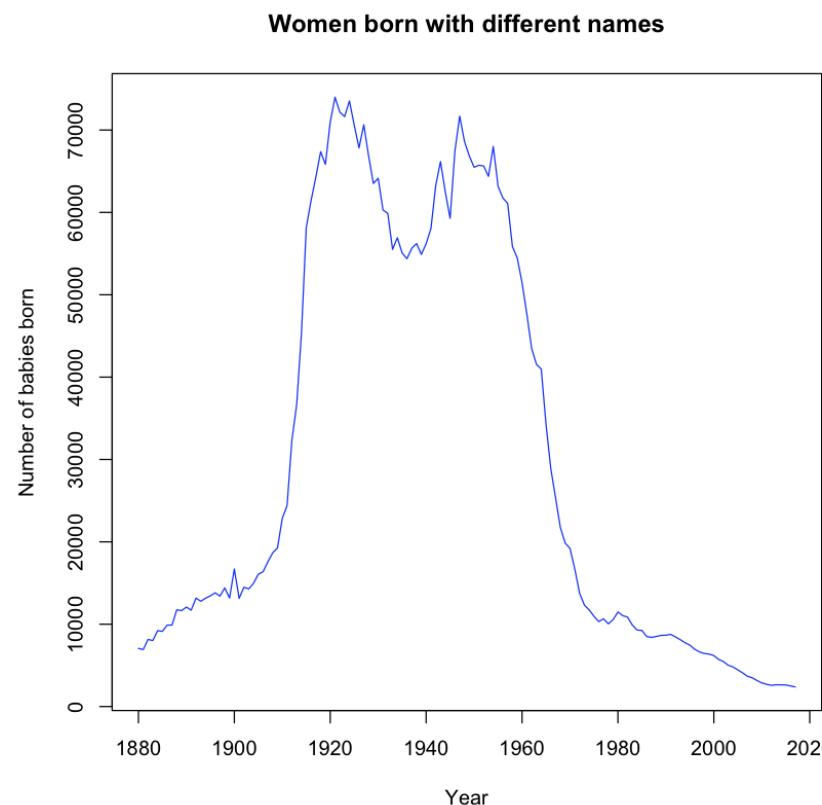
- Step 4: Selezionare le informazioni da rappresentare

- **%>%:** È l'operatore pipe, fornito da dplyr, che passa l'output del comando a sinistra come input al comando a destra.
- **filter(name %in% c("Alice")):** Filtra il dataset babynames per includere solo le righe in cui la colonna name ha il valore 'Alice', 'Mary' o 'Mae'.
  - La funzione %in% verifica se il valore della colonna name è contenuto nel vettore specificato, che in questo caso è solo c("Alice").
- **filter(sex == "F"):** Filtra ulteriormente il dataset per includere solo le righe in cui la colonna sex ha il valore "F" (femminile).

# GRAFICI PER VETTORI NUMERICI

- Vogliamo rappresentare come è cambiato l'utilizzo dei nomi 'Alice', 'Mary' e 'Mae' nei bambini nati tra il 1880 ed il 2020

```
plot(dat2$year, dat2$n, type = "l", col = "blue",
  main = "Women born with different names",
  xlab = "Year",
  ylab = "Number of babies born")
```

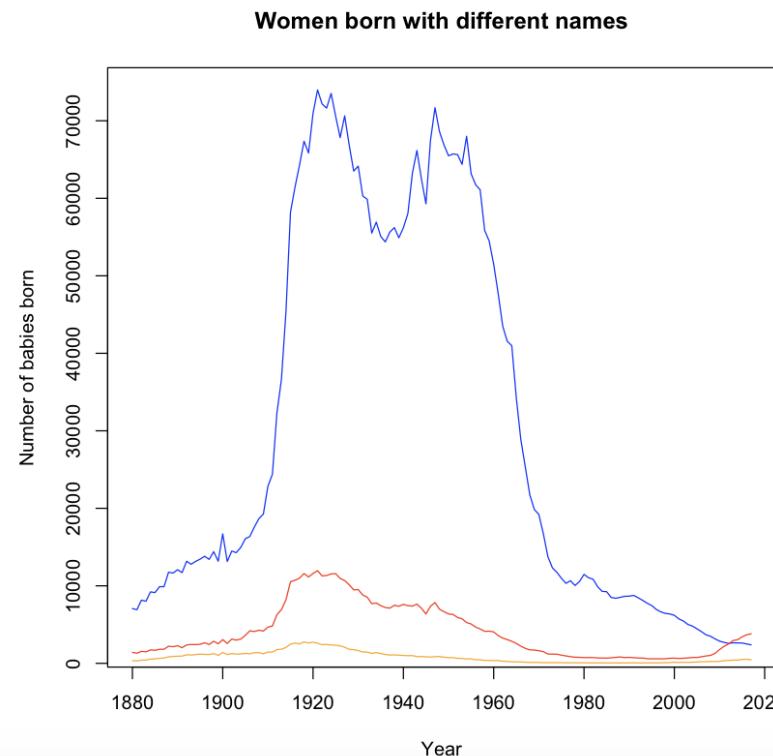


- Step 5: Rappresentare gli insiemi di osservazioni
  - Crea un grafico iniziale con gli anni (`dat2$year`) sull'asse x e il numero di bambine nate (`dat2$n`) sull'asse y.
  - `type = "l"` specifica che il grafico è di tipo "linea" (line plot).`col = "blue"` imposta la linea del grafico di colore blu, che rappresenterà il nome "Mary".
  - `main = "Women born with different names"` imposta il titolo del grafico.
  - `xlab = "Year" e ylab = "Number of babies born"` impostano le etichette per gli assi x e y, rispettivamente.

# GRAFICI PER VETTORI NUMERICI

- Vogliamo rappresentare come è cambiato l'utilizzo dei nomi 'Alice', 'Mary' e 'Mae' nei bambini nati tra il 1880 ed il 2020

```
plot(dat2$year, dat2$n, type = "l", col = "blue",
      main = "Women born with different names",
      xlab = "Year",
      ylab = "Number of babies born")
lines(dat$year, dat$n, col = "red")
lines(dat3$year, dat3$n, col = "orange")
```



- Step 5: Rappresentare gli insiemi di osservazioni
  - Aggiunge una seconda linea al grafico, utilizzando i dati contenuti in dat, dove **dat\$year** rappresenta gli anni e **dat\$n** il numero di bambine nate con il nome "Alice". La linea è colorata in rosso (**col = "red"**).
  - Aggiunge una terza linea al grafico, con i dati di dat3, per il nome "Mae". Anche in questo caso, **dat3\$year** rappresenta gli anni e **dat3\$n** il numero di bambine nate. La linea è colorata in arancione (**col = "orange"**).

# STATISTICA E ANALISI DEI DATI

Capitolo 2 – Serie Temporali

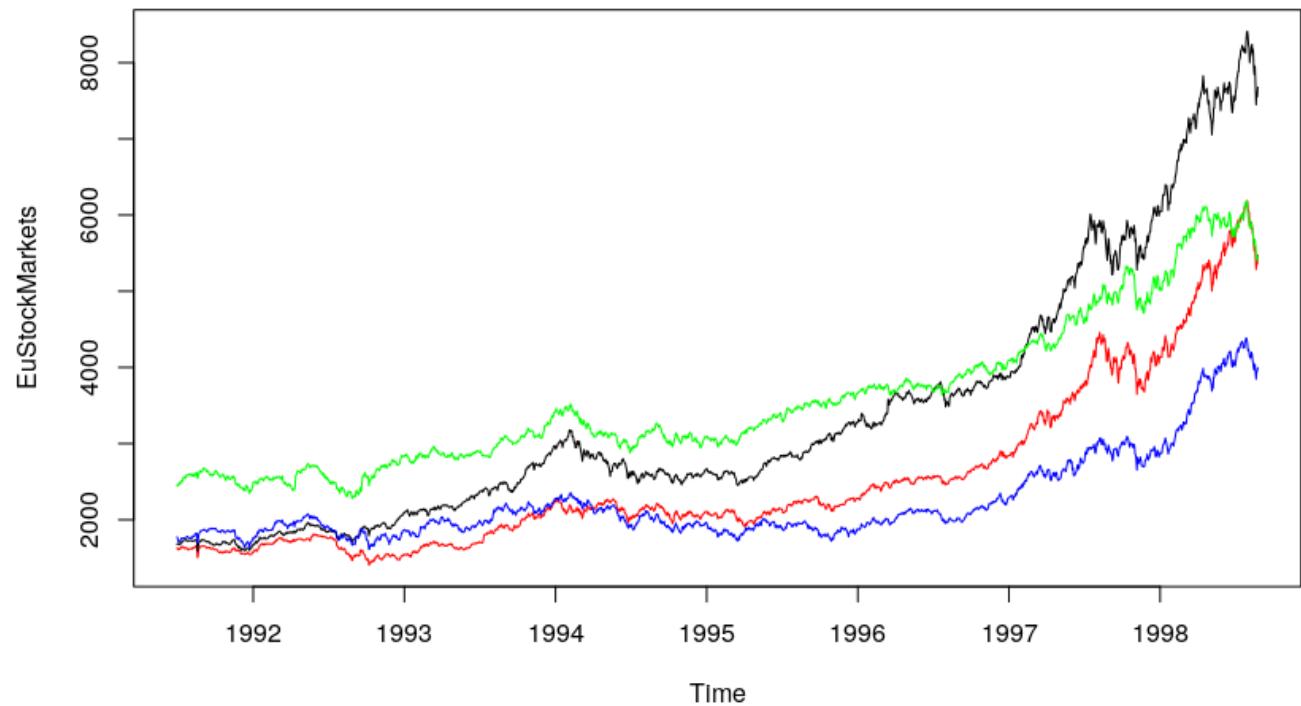
---

Dott. Stefano Cirillo  
Dott. Luigi Di Biasi

a.a. 2023-2024

# SERIE TEMPORALI

- Una **serie temporale** è una sequenza di osservazioni di una variabile raccolte o registrate ad intervalli di tempo regolari.
- L'obiettivo è analizzare i dati per identificare pattern o tendenze che possano aiutare a fare previsioni sul futuro (forecasting)
- **Esempi comuni:**
  - Prezzi di chiusura di un'azione
  - Temperatura media mensile
  - Produzione industriale annuale
  - ...



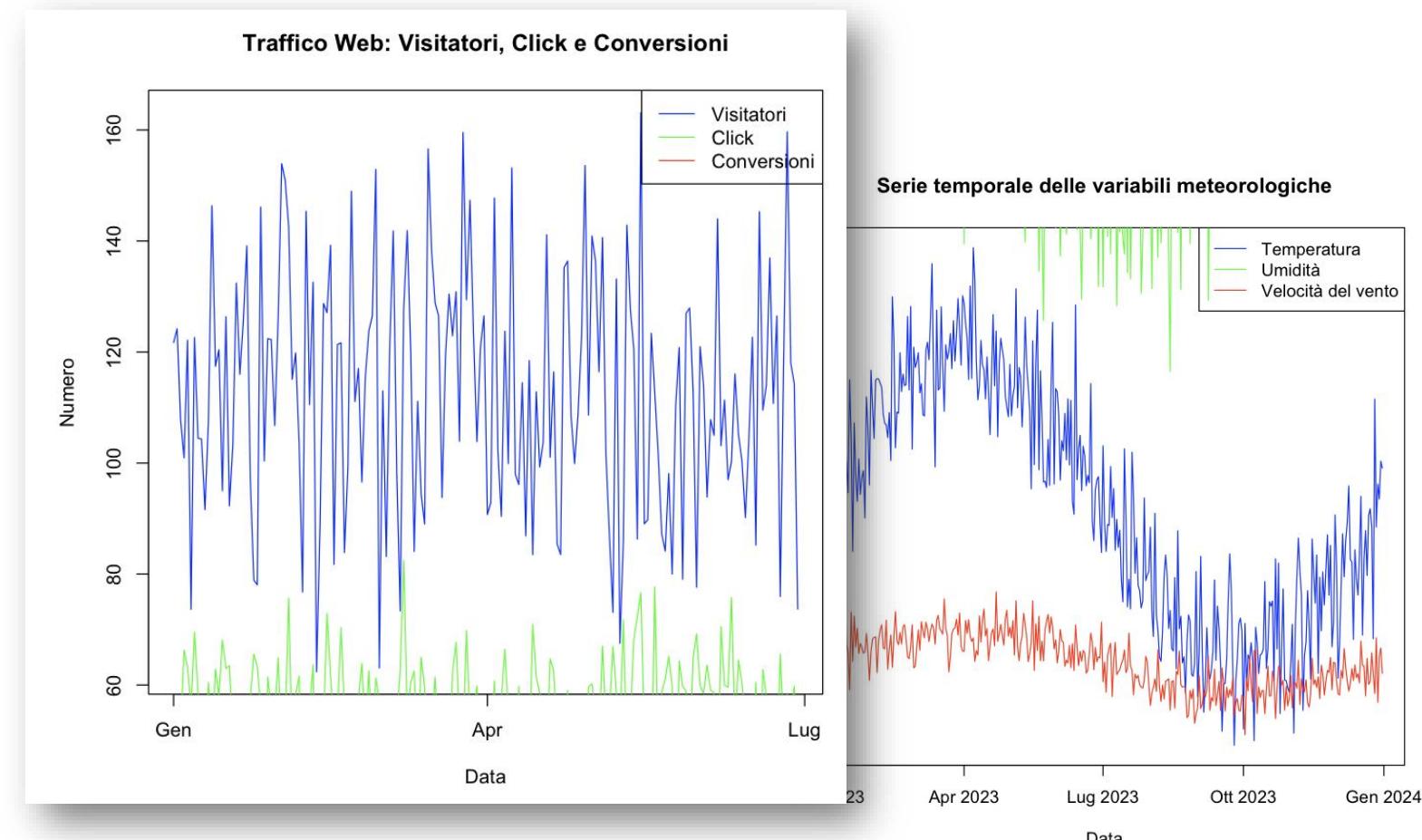
# SERIE TEMPORALI UNIVARIATE

- Una **serie temporale univariata** analizza una sola variabile misurata nel tempo.
- L'attenzione è sulla dinamica temporale di una singola quantità.
- **Esempi comuni:**
  - Grafico del prezzo dell'azione di una singola società (solo una variabile: il prezzo) nel tempo
  - Monitorare la temperatura giornaliera o mensile in una specifica località è un esempio classico di serie temporale univariata. Puoi visualizzare le variazioni di temperatura nel corso del tempo
  - Consumo energetico giornaliero
  - Tasso di cambio valuta
  - ....



# SERIE TEMPORALI MULTIVARIATE

- Una **serie temporale multivariata** considera più variabili misurata nel tempo.
- L'attenzione è sulla dinamica temporale di una singola quantità.
- **Esempi comuni:**
  - Dati meteorologici (Temperatura, Umidità e Velocità del vento)
  - Dati economici (PIL, inflazione e disoccupazione)
  - Dati di traffico su un sito web (Visitatori, Click e Conversioni)
  - ....



# SERIE TEMPORALI MULTIVARIATE

Caratteristica	Serie Temporale Univariata	Serie Temporale Multivariata	Aspetti Comuni
<b>Analisi di tendenze</b>	Si concentra sulle tendenze di una sola variabile	Analizza le tendenze e le relazioni tra più variabili	Si possono usare tecniche di analisi simili (es. analisi di trend)
<b>Numero di variabili</b>	Una sola variabile osservata nel tempo	Due o più variabili osservate simultaneamente nel tempo	Basate su misurazioni nel tempo
<b>Applicazioni comuni</b>	Finanza, meteorologia, economia	Economia, medicina, studi ambientali	Utilizzate in vari campi per previsioni e analisi storiche
<b>Complessità</b>	Più semplice da analizzare e interpretare	Più complessa, richiede tecniche di analisi che considerano la correlazione tra variabili	Richiedono metodi statistici per estrarre insight dal tempo
<b>Visualizzazione</b>	Grafico a linea singola	Grafici con più linee, o utilizzo di matrici di covarianza	Grafici temporali utilizzano in genere gli assi tempo-valore

# SERIE TEMPORALI

---

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

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

- Dove:
  - **x** : valori della serie (vettore nel caso univariato, matrice o data frame nel caso multivariato);
  - **start**: istante iniziale temporale (ad esempio, 2010);
  - **frequency** : numero di osservazioni nell'unità di tempo;
  - **deltat** : la distanza temporale tra le osservazioni;
  - **end** : istante finale.

# SERIE TEMPORALI

---

- La funzione `plot(y)` permette successivamente di rappresentare il grafico della serie temporale creata.
- Il parametro opzionale `start` serve per stabilire **l'istante temporale** a cui deve essere riferita la prima osservazione
- Ad esempio, supponiamo che i valori contenuti in `x` siano una serie storica annuale e che la prima osservazione sia da riferire al 2010. Possiamo usare:

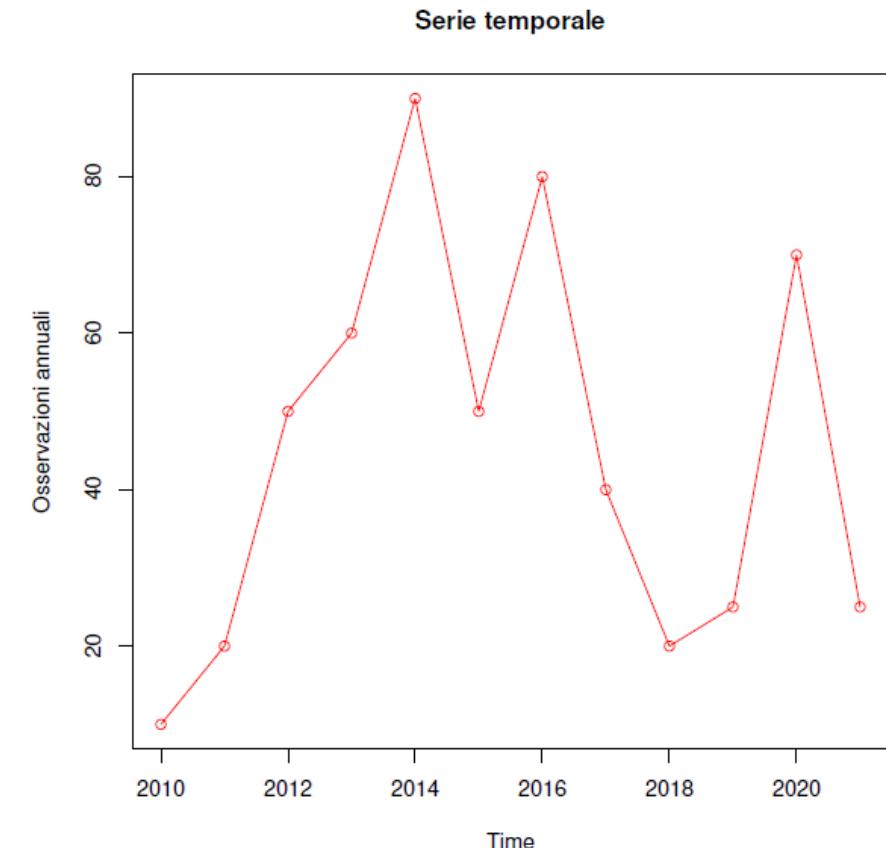
```
y <- ts(x, start=2010, frequency=1)
```

- R calcola automaticamente l'anno finale in base alla lunghezza del vettore `x` e gli anni sono rappresentati sull'asse delle ascisse nel grafico realizzato con la funzione `plot(y)`.
- Vediamo un esempio...

# SERIE TEMPORALI

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

- In questo caso la frequenza delle osservazioni è annuale



# SERIE TEMPORALI

---

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

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

- Oppure, equivalentemente:

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

- Da notare che **frequency** rappresenta il reciproco di **deltat**, ossia fornisce il numero di osservazioni per unità di tempo. Nel caso semestrale delle osservazioni, la frequenza è 2 e la distanza temporale tra le osservazioni è 0.5.

# SERIE TEMPORALI

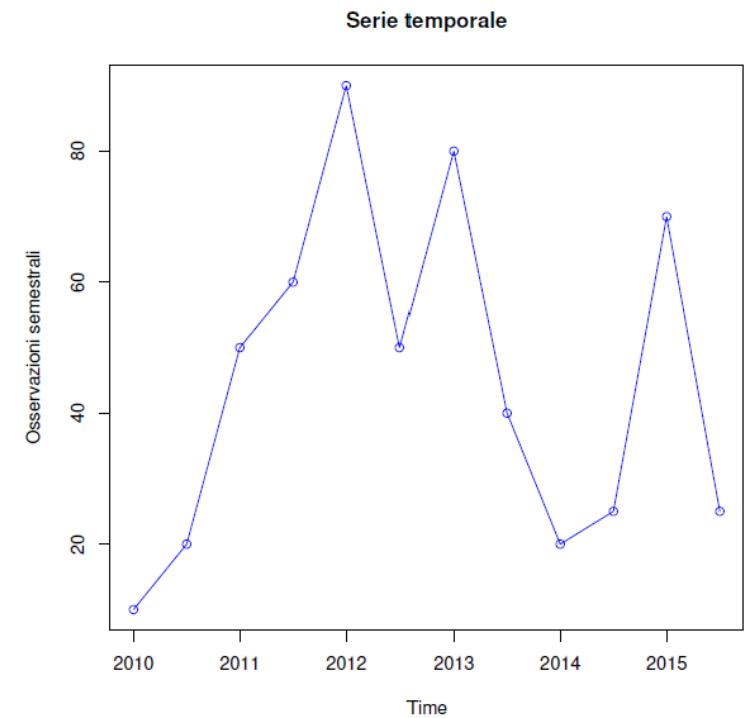
- Ad esempio, il seguente codice:

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

- o equivalentemente il seguente...

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

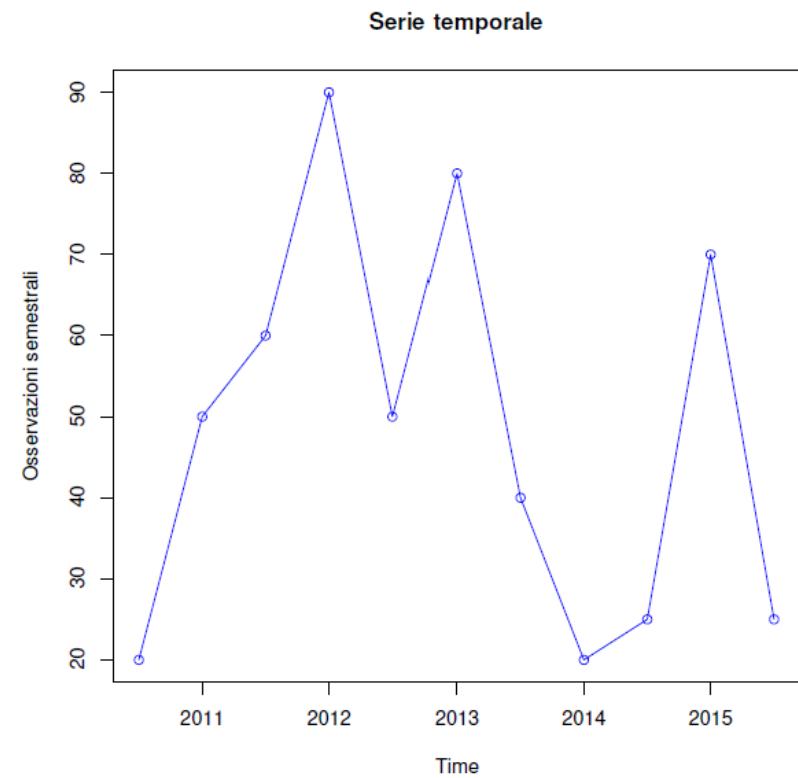
- ... producono un grafico con frequenza delle osservazioni semestrale



# SERIE TEMPORALI

- Se la prima osservazione corrisponde al secondo semestre dell'anno 2010 si può procedere nel seguente modo:

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



# SERIE TEMPORALI

---

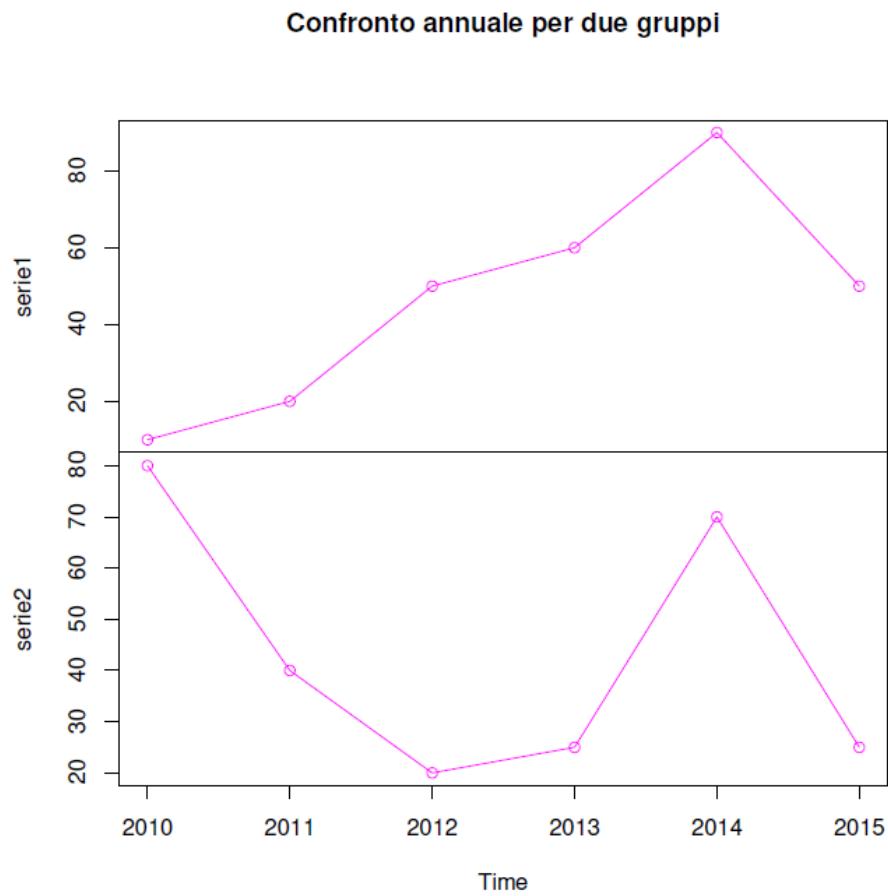
- E se le osservazioni sono **mensili**?
- Consideriamo delle osservazioni mensili che partono dal quarto mese del 2010:

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

# SERIE TEMPORALI

- È possibile inoltre confrontare sullo stesso grafico diverse serie temporali rappresentate su differenti colonne (osservazioni) di un data frame:

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

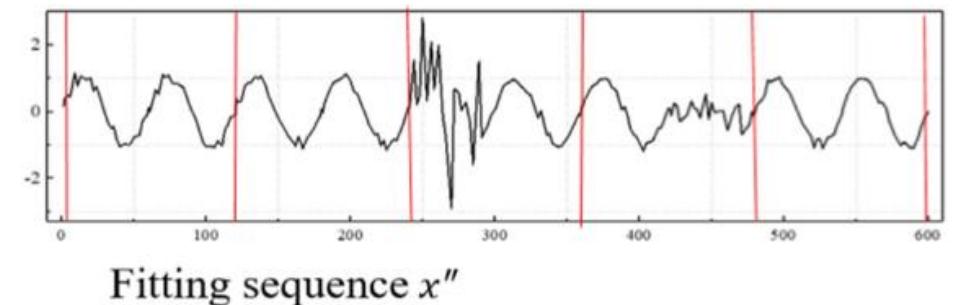


# WINDOW

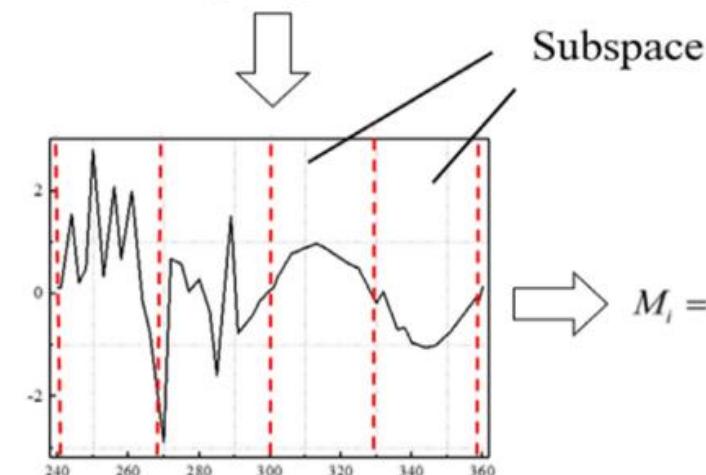
- La funzione `window()` è utile per lavorare con serie temporali quando si vuole isolare o analizzare un sottoinsieme di dati temporali
- Viene principalmente applicata agli oggetti di tipo `ts` (time series), tipici nell'analisi di dati temporali

`window(x, start, end)`

- `x`: L'oggetto di tipo `ts` (serie temporale) da cui estrarre la sotto-serie.
- `start`: Il punto di inizio della sotto-serie (in termini di tempo).
- `end`: Il punto di fine della sotto-serie.



Fitting sequence  $x''$



Subsequence  $X_i$

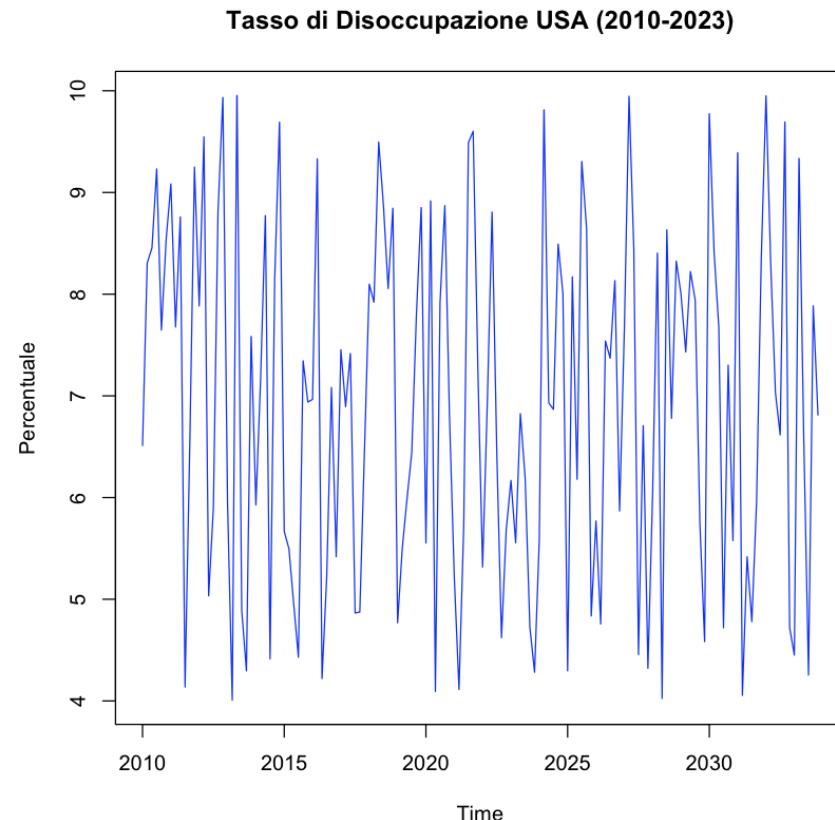
$$M_i = \begin{bmatrix} 11 & 0.69 & 3.52 & 14.77 \\ 11 & -2.95 & 3.09 & 8.46 \\ 8 & 0.63 & 0.85 & 1.37 \\ 12 & -6.3 & 1.18 & 4.45 \end{bmatrix}$$

Pattern matrix  $M_i$

# WINDOW

- Esempio: Supponiamo di avere una serie temporale dei tassi di disoccupazione mensili negli Stati Uniti dal 2010 al 2023

```
# Dati ipotetici di tassi di disoccupazione mensili dal 2010 al 2023  
disoccupazione <- ts(c(runif(144, 4, 10)), start = c(2010, 1), frequency = 12)  
  
# Visualizzazione della serie temporale  
plot(disoccupazione, main = "Tasso di Disoccupazione USA (2010-2023)",  
      ylab = "Percentuale", col = "blue")
```

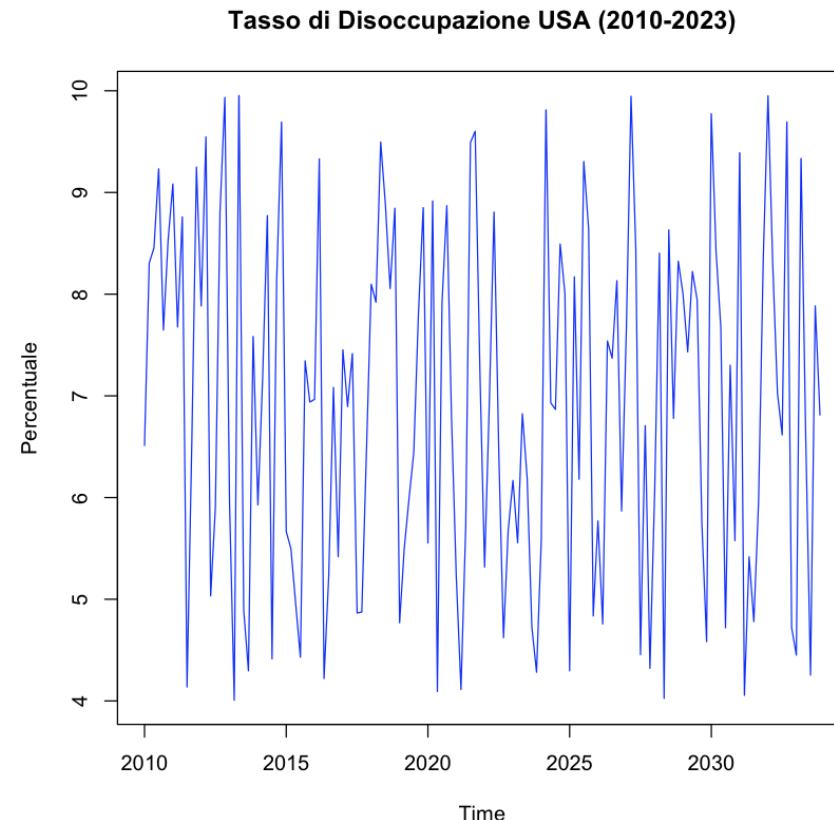


# WINDOW

- Esempio: Supponiamo di avere una serie temporale dei tassi di disoccupazione mensili negli Stati Uniti dal 2010 al 2023

- **runif()** genera numeri casuali distribuiti uniformemente nell'intervallo **[min,max]**, dove ogni valore ha la stessa probabilità di essere selezionato
  - *min=4*
  - *max=10*
  - *Numero di valori = 144*

```
# Dati ipotetici di tassi di disoccupazione mensili dal 2010 al 2023  
disoccupazione <- ts(c(runif(144, 4, 10)), start = c(2010, 1), frequency = 12)  
  
# Visualizzazione della serie temporale  
plot(disoccupazione, main = "Tasso di Disoccupazione USA (2010-2023)",  
      ylab = "Percentuale", col = "blue")
```

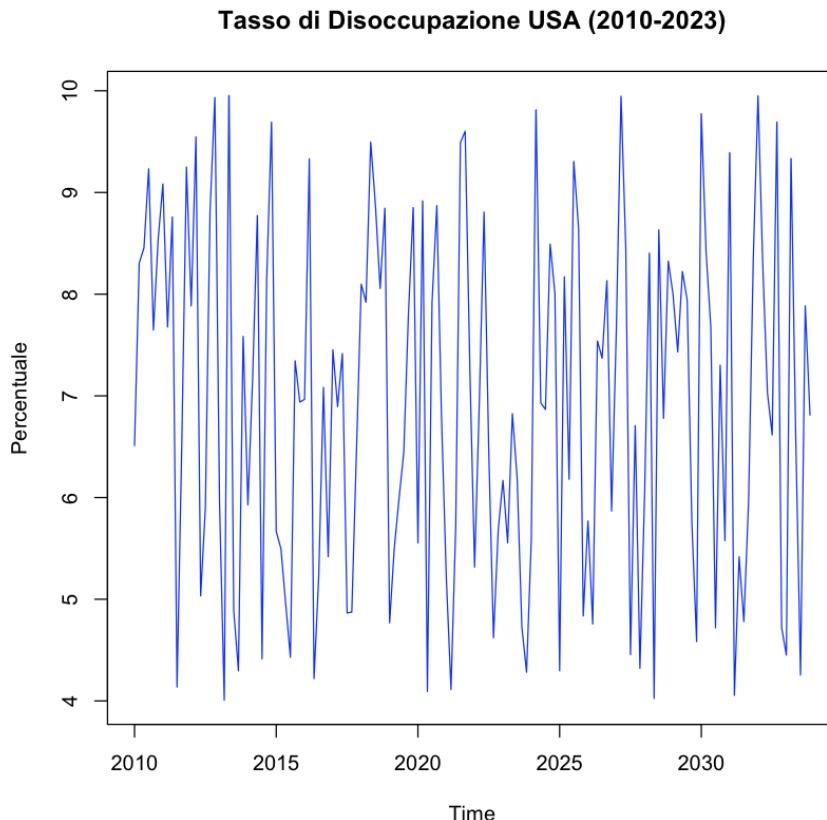


# WINDOW

- Esempio: Supponiamo di avere una serie temporale dei tassi di disoccupazione mensili negli Stati Uniti dal 2010 al 2023
  - **runif()** genera numeri casuali distribuiti uniformemente nell'intervallo **[min,max]**, dove ogni valore ha la stessa probabilità di essere selezionato
    - *min=4*
    - *max=10*
    - *Numero di valori = 144*
  - **frequency = 12** indica che la serie temporale ha 12 osservazioni per anno, quindi i dati sono mensili

```
# Dati ipotetici di tassi di disoccupazione mensili dal 2010 al 2023
disoccupazione <- ts(c(runif(144, 4, 10)), start = c(2010, 1), frequency = 12)

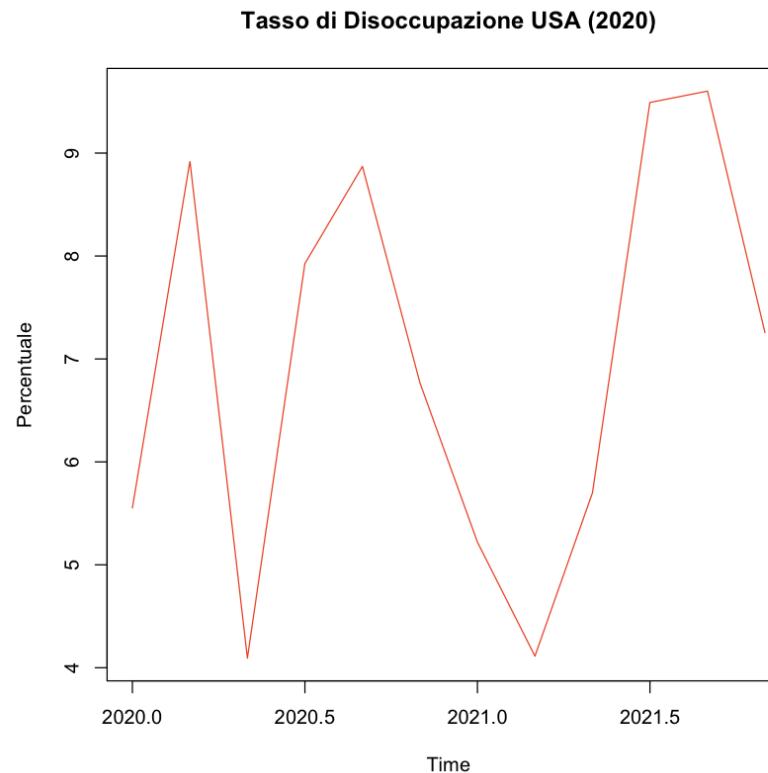
# Visualizzazione della serie temporale
plot(disoccupazione, main = "Tasso di Disoccupazione USA (2010-2023)",
      ylab = "Percentuale", col = "blue")
```



# WINDOW

- Esempio: Supponiamo di avere una serie temporale dei tassi di disoccupazione mensili negli Stati Uniti dal 2010 al 2023
- Supponiamo di voler analizzare solo i dati relativi al periodo della crisi economica del 2020 dovuta alla pandemia, quindi estraiamo i dati solo per l'anno 2020

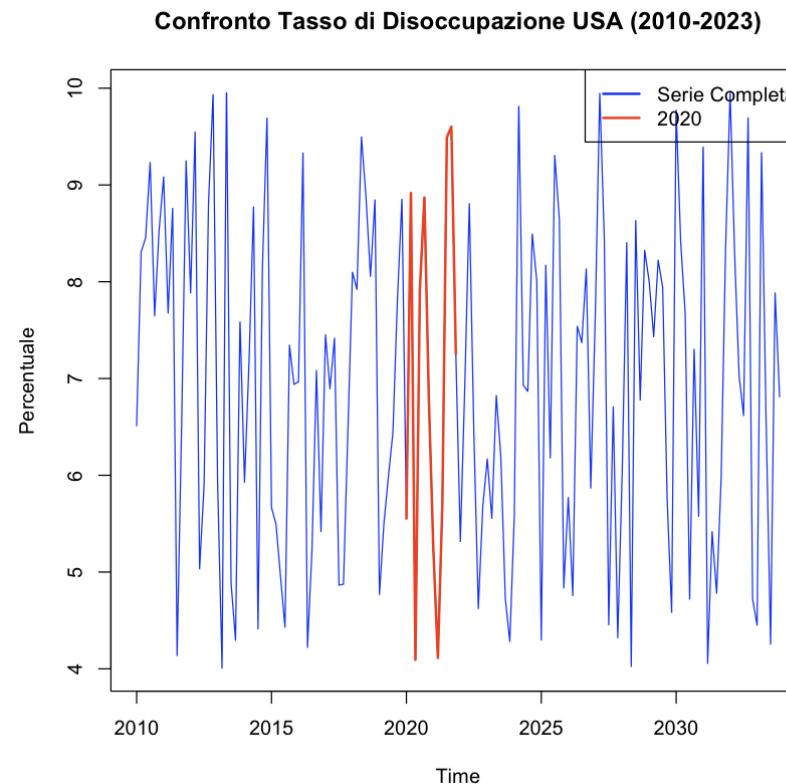
```
# Estrarre i dati solo per l'anno 2020  
disoccupazione_2020 <- window(disoccupazione, start = c(2020, 1), end = c(2020, 12))  
  
# Visualizzazione della sotto-serie  
plot(disoccupazione_2020, main = "Tasso di Disoccupazione USA (2020)",  
      ylab = "Percentuale", col = "red")
```



# WINDOW

- Esempio: Supponiamo di avere una serie temporale dei tassi di disoccupazione mensili negli Stati Uniti dal 2010 al 2023
- Supponiamo di voler analizzare solo i dati relativi al periodo della crisi economica del 2020 dovuta alla pandemia, quindi estraiamo i dati solo per l'anno 2020
- Grazie alla funzione **window** siamo in grado di isolare una parte della TS

```
# Grafico della serie completa e della sotto-serie per il 2020
plot(disoccupazione, main = "Confronto Tasso di Disoccupazione USA (2010-2023)",
      ylab = "Percentuale", col = "blue")
lines(disoccupazione_2020, col = "red", lwd = 2)
legend("topright", legend = c("Serie Completa", "2020"), col = c("blue", "red"),
      lwd = 2)
```



# TSSTUDIO

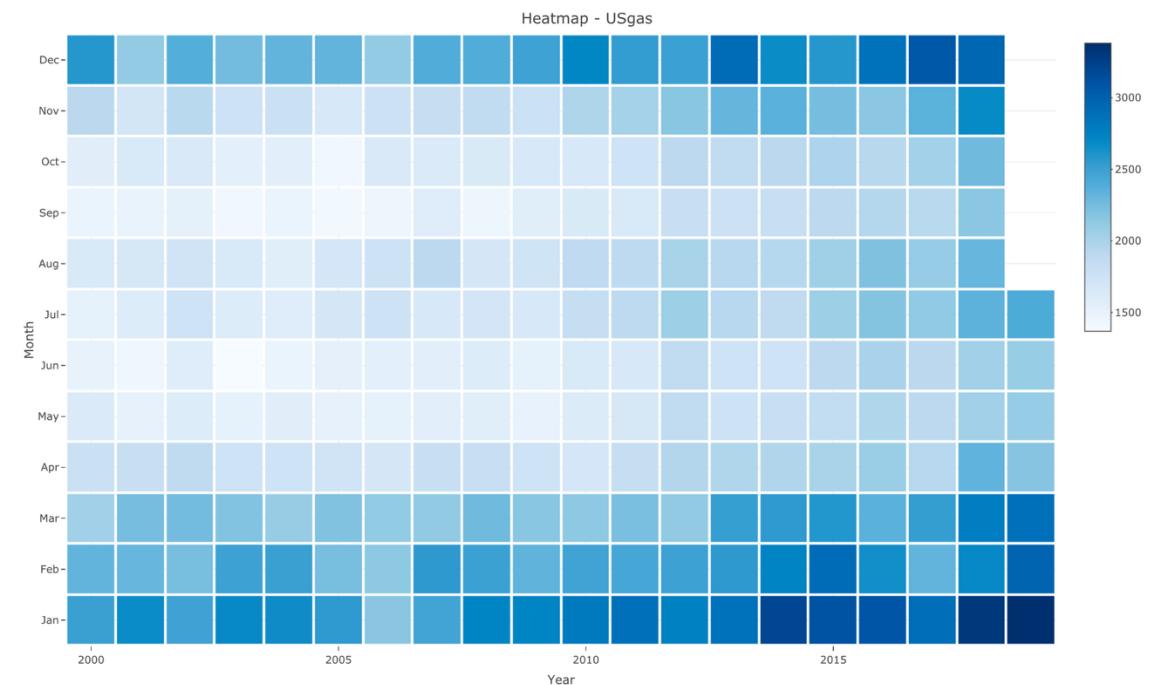
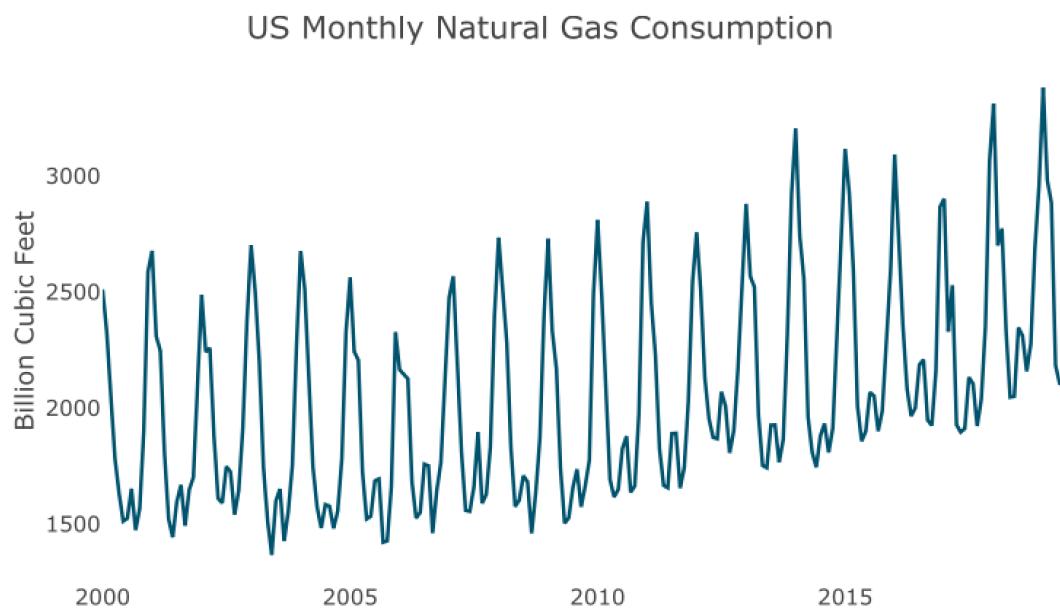
---

- TSstudio è una libreria avanzata per R, specializzata nell'analisi e nella visualizzazione di serie temporali.
- Questa libreria è stata progettata per fornire agli analisti di dati, ricercatori e professionisti un insieme completo di strumenti per lavorare con dati temporali in modo efficiente e intuitivo.
- Cos'è TSstudio:
  - **Toolkit integrato:** TSstudio si presenta come un toolkit completo che riunisce diverse funzionalità per l'analisi delle serie temporali in un'unica piattaforma
  - **Interfaccia user-friendly:** Offre un'interfaccia utente più accessibile per operazioni complesse di analisi delle serie temporali, rendendo queste tecniche più accessibili anche a utenti meno esperti in programmazione R.
  - **Ponte tra analisi e visualizzazione:** TSstudio colma il divario tra l'analisi statistica rigorosa e la visualizzazione intuitiva dei dati, permettendo agli utenti di esplorare, analizzare e presentare i risultati in modo efficace.



# TSSTUDIO

- Visualizzazione interattiva:
    - Crea grafici interattivi di serie temporali con la funzione `ts_plot()`.
    - Permette zoom, pan e selezione di intervalli specifici nei grafici.
    - Genera heatmap per visualizzare pattern complessi con `ts_heatmap()`.

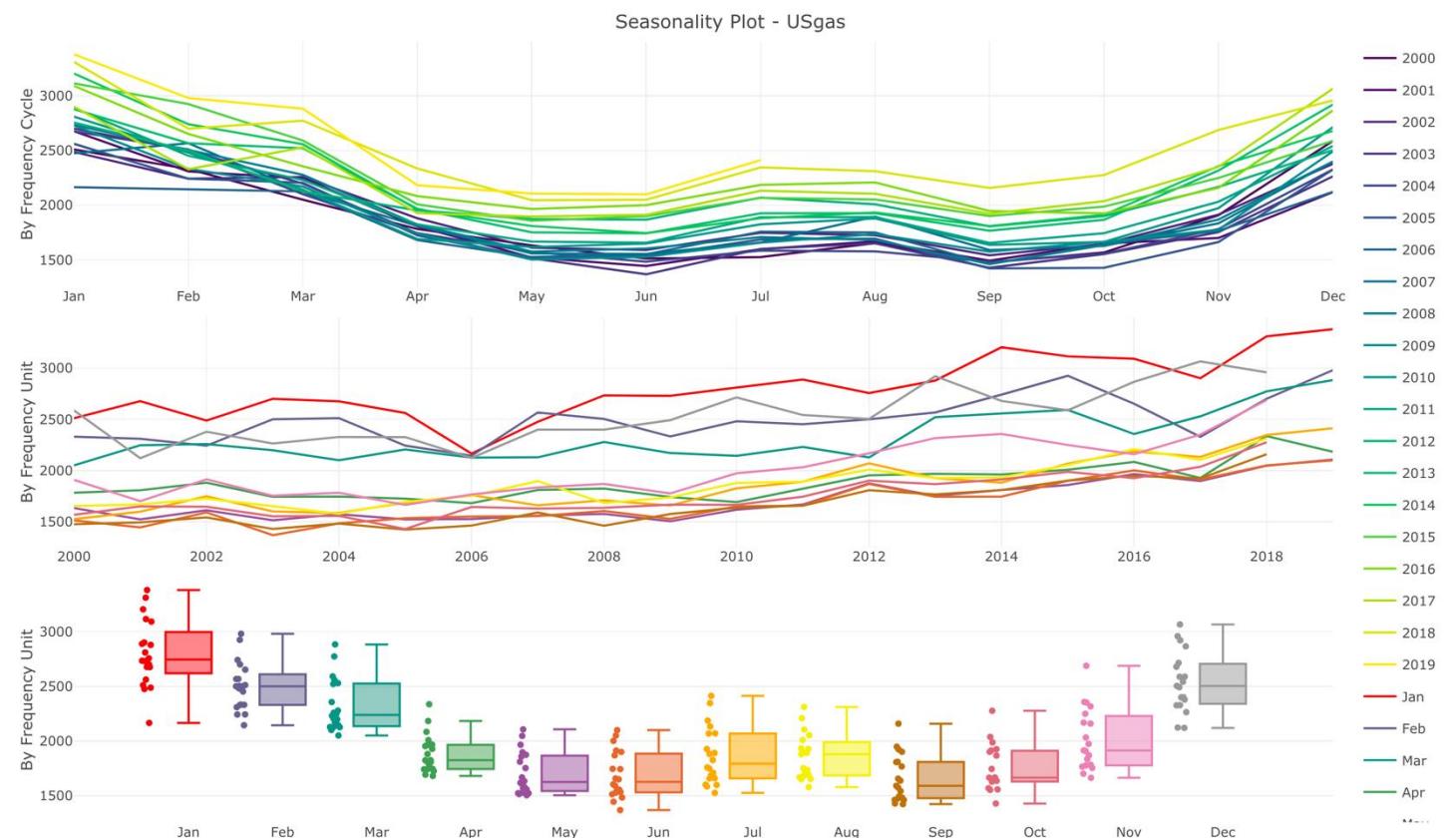


# TSSTUDIO

- Visualizzazione interattiva:
  - Crea grafici interattivi di serie temporali con la funzione `ts_plot()`.
  - Permette zoom, pan e selezione di intervalli specifici nei grafici.
  - Genera heatmap per visualizzare pattern complessi con `ts_heatmap()`.
- Analisi della stagionalità:
  - Offre strumenti avanzati per l'analisi dei pattern stagionali con `ts_seasonal()`.
  - Produce grafici stagionali, subseries plot, e polar plot per una comprensione approfondita della stagionalità.
- Decomposizione delle serie:
  - Decomponete le serie temporali in componenti di trend, stagionalità e residui con `ts_decompose()`.
  - Supporta vari metodi di decomposizione, inclusi STL e X-11.
- Modellazione e previsione:
  - Facilita la creazione di modelli predittivi con funzioni come `ts_forecast()`.
  - Supporta vari algoritmi di previsione, tra cui ARIMA, ETS, e modelli di machine learning.

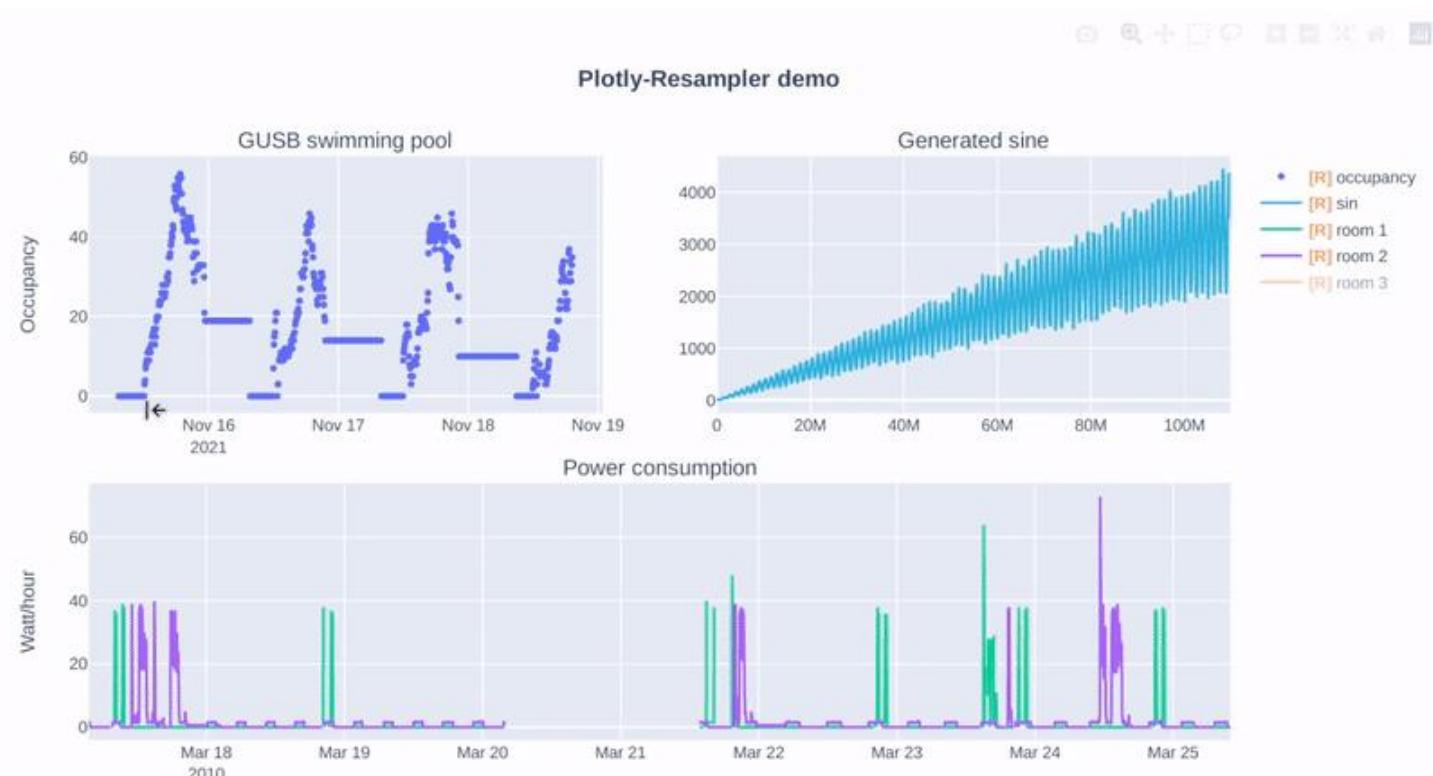


- Analisi della stagionalità:
  - Offre strumenti avanzati per l'analisi dei pattern stagionali con `ts_seasonal()`.
  - Produce grafici stagionali, subseries plot, e polar plot per una comprensione approfondita della stagionalità.
- Decomposizione delle serie:
  - Decomponete le serie temporali in componenti di trend, stagionalità e residui con `ts_decompose()`.
  - Supporta vari metodi di decomposizione, inclusi STL e X-11.
- Modellazione e previsione:
  - Facilita la creazione di modelli predittivi con funzioni come `ts_split()` e librerie di forecast come il pacchetto `forecasting`
  - Supporta vari algoritmi di previsione, tra cui ARIMA, ETS, e modelli di machine learning.



# INSTALLAZIONE TSSTUDIO

- Il pacchetto [TSstudio](#) fornisce un set di strumenti per l'analisi descrittiva e predittiva dei dati delle serie temporali
- Ciò include funzioni di utilità per la preelaborazione dei dati delle serie temporali, funzioni di visualizzazione interattiva basate sul motore del pacchetto [plotly](#) e un set di strumenti per l'addestramento e la valutazione dei modelli di previsione delle serie temporali dai pacchetti [forecast](#), [forecastHybrid](#) e [bsts](#)
- Il pacchetto TSstudio è disponibile su GitHub (<https://github.com/RamiKrispin/TSstudio>)



```
install.packages("TSstudio")
```



- Oppure:

```
# install.packages("devtools")
devtools::install_github("RamiKrispin/TSstudio")
```



# ESEMPIO SERIE TEMPORALE

- Vogliamo analizzare i dati reali sui prezzi gas negli USA per identificare tendenze e comportamenti stagionali

- Questo tipo di analisi può essere utile:
  - Per aziende che devono pianificare strategie di approvvigionamento energetico
  - Per clienti che vogliono sottoscrivere contratti di fornitura al miglior prezzo

## • Step 1: Identificazione dei dati

- Il dataset USgas contiene i dati mensili sui consumi di gas naturale negli Stati Uniti, espressi in **miliardi di piedi cubici**. È un oggetto di serie temporale.

```
library(TSstudio)
data(USgas)
USgas
```

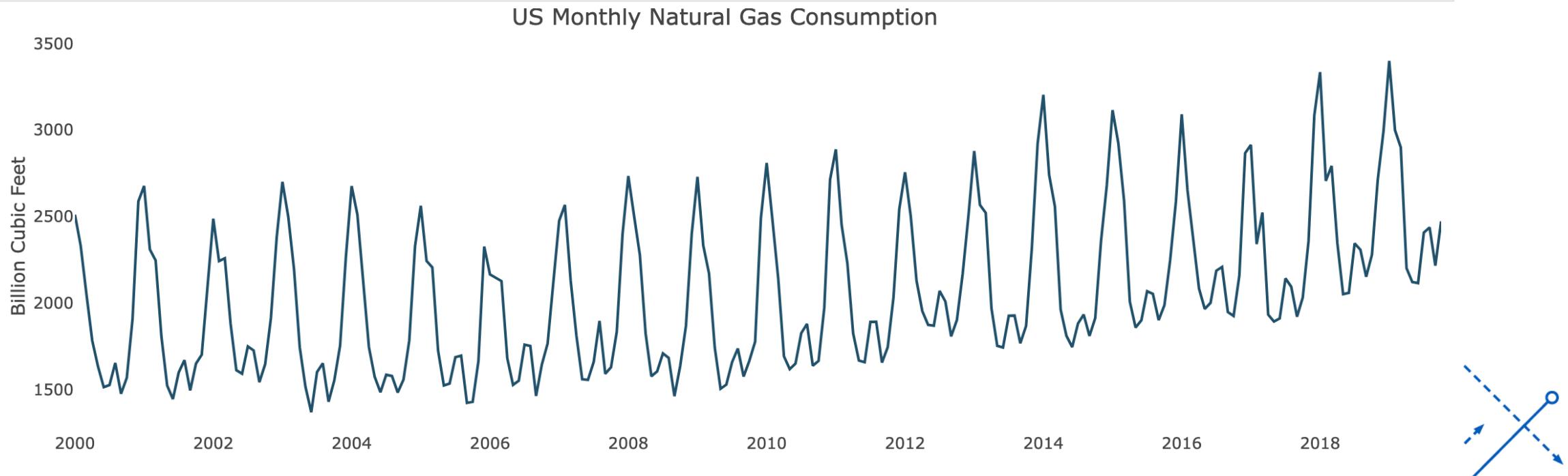
A Time Series: 20 × 12

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
2000	2510.5	2330.7	2050.6	1783.3	1632.9	1513.1	1525.6	1653.1	1475.0	1567.8	1908.5	2587.5
2001	2677.0	2309.5	2246.6	1807.2	1522.4	1444.4	1598.1	1669.2	1494.1	1649.1	1701.0	2120.2
2002	2487.6	2242.4	2258.4	1881.0	1611.5	1591.4	1748.4	1725.7	1542.2	1645.9	1913.6	2378.9
2003	2700.5	2500.3	2197.9	1743.5	1514.7	1368.4	1600.5	1651.6	1428.6	1553.2	1753.6	2263.7
2004	2675.8	2511.1	2100.9	1745.2	1573.0	1483.7	1584.9	1578.0	1482.2	1557.2	1782.8	2327.7
2005	2561.9	2243.0	2205.8	1724.9	1522.6	1534.1	1686.6	1695.1	1422.5	1428.2	1663.4	2326.4
2006	2165.3	2144.4	2126.4	1681.0	1526.3	1550.9	1758.7	1751.7	1462.1	1644.2	1765.4	2122.8
2007	2475.6	2567.0	2128.8	1810.1	1559.1	1555.2	1659.9	1896.1	1590.5	1627.8	1834.5	2399.2
2008	2734.0	2503.4	2278.2	1823.9	1576.4	1604.2	1708.6	1682.9	1460.9	1635.8	1868.9	2399.7
2009	2729.7	2332.5	2170.7	1741.3	1504.0	1527.8	1658.0	1736.5	1575.0	1666.5	1776.2	2491.9
2010	2809.8	2481.0	2142.9	1691.8	1617.3	1649.5	1825.8	1878.9	1637.5	1664.9	1973.3	2714.1
2011	2888.6	2452.4	2230.5	1825.0	1667.4	1657.3	1890.5	1891.8	1655.6	1744.5	2031.9	2541.9
2012	2756.2	2500.7	2127.8	1953.1	1873.8	1868.4	2069.8	2008.8	1807.2	1901.1	2167.8	2503.9
2013	2878.8	2567.2	2521.1	1967.5	1752.5	1742.9	1926.3	1927.4	1767.0	1866.8	2316.9	2920.8
2014	3204.1	2741.2	2557.9	1961.7	1810.2	1745.4	1881.0	1933.1	1809.3	1912.8	2357.5	2679.2
2015	3115.0	2925.2	2591.3	2007.9	1858.1	1899.9	2067.7	2052.7	1901.3	1987.3	2249.1	2588.2
2016	3091.7	2652.3	2356.3	2083.8	1965.8	2000.7	2186.6	2208.4	1947.8	1925.2	2159.4	2866.3
2017	2914.2	2340.6	2523.7	1932.5	1892.5	1910.9	2142.1	2094.3	1920.9	2032.0	2357.7	3084.5
2018	3335.0	2705.9	2792.6	2346.3	2050.9	2058.7	2344.6	2307.7	2151.5	2279.1	2709.9	2993.1
2019	3399.9	2999.2	2899.9	2201.1	2121.0	2115.2	2407.5	2437.2	2215.6	2472.3		

# ESEMPIO SERIE TEMPORALE

- **Step 2a:** Creazione del Grafico: la funzione **ts\_plot()** per tracciare un grafico della serie temporale del dataset USgas.
  - **title:** il titolo del grafico, "US Monthly Natural Gas Consumption", che descrive il contenuto della serie temporale.
  - **Ytitle:** Etichetta l'asse delle ordinate (y-axis) con "Billion Cubic Feet", l'unità di misura dei dati

```
# Ploting time series object
ts_plot(USgas,
        title = "US Monthly Natural Gas Consumption",
        Ytitle = "Billion Cubic Feet")
```



# ESEMPIO SERIE TEMPORALE

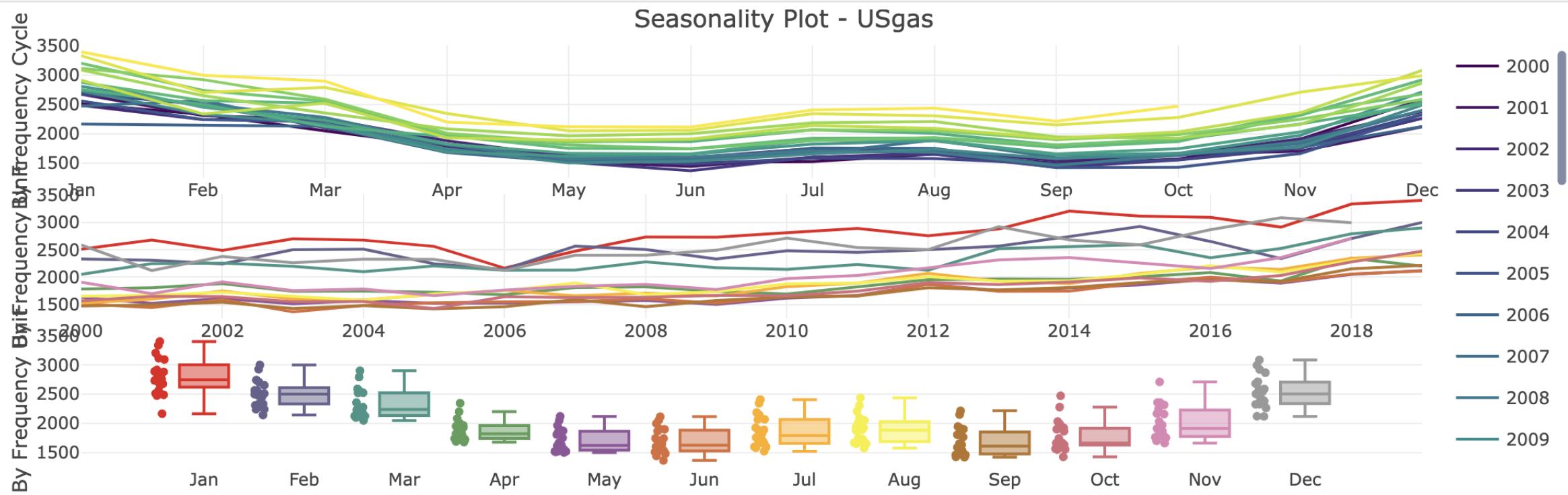
---

- **Step 2b:** Creazione del Grafico: la funzione **ts\_seasonal()** per tracciare un grafico della serie temporale del dataset Usgas **stagionale**
  - Questa analisi completa sarà particolarmente utile per:
    - Identificare pattern stagionali ricorrenti nel consumo di gas.
    - Rilevare eventuali cambiamenti nei pattern stagionali nel corso degli anni.
    - Individuare anomalie o outlier stagionali.
    - Comprendere la variabilità del consumo all'interno di ciascuna stagione.

# ESEMPIO SERIE TEMPORALE

- L'opzione type = "all" assicura che tutti questi tipi di grafici e analisi vengano prodotti, offrendo una visione completa e multidimensionale della stagionalità nel consumo di gas naturale negli Stati Uniti

```
# Seasonal plot  
ts_seasonal(USgas, type = "all")
```



# STATISTICA E ANALISI DEI DATI

Capitolo 2 – Altri tipi di Grafici

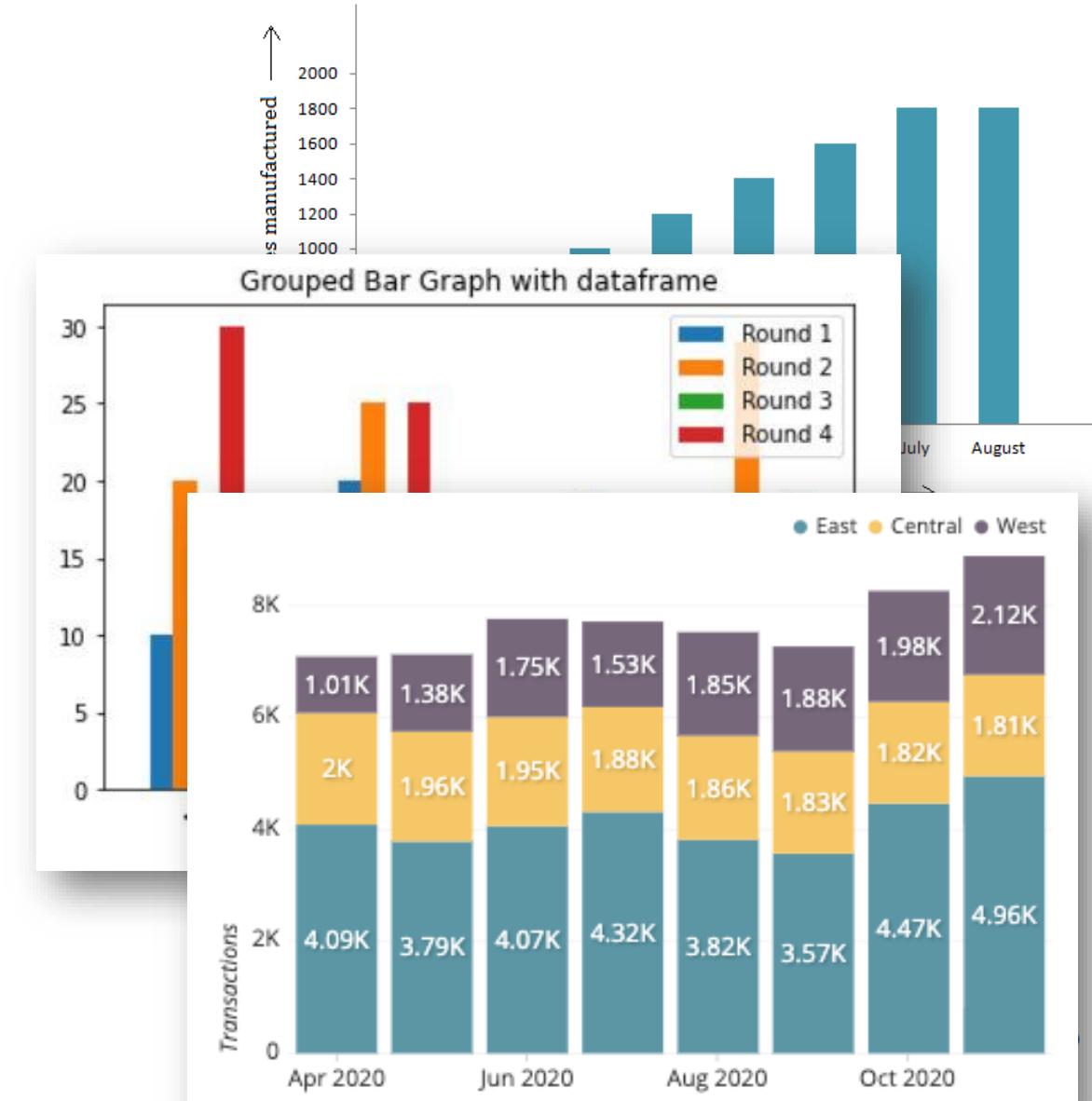
---

Dott. Stefano Cirillo  
Dott. Luigi Di Biasi

a.a. 2025-2026

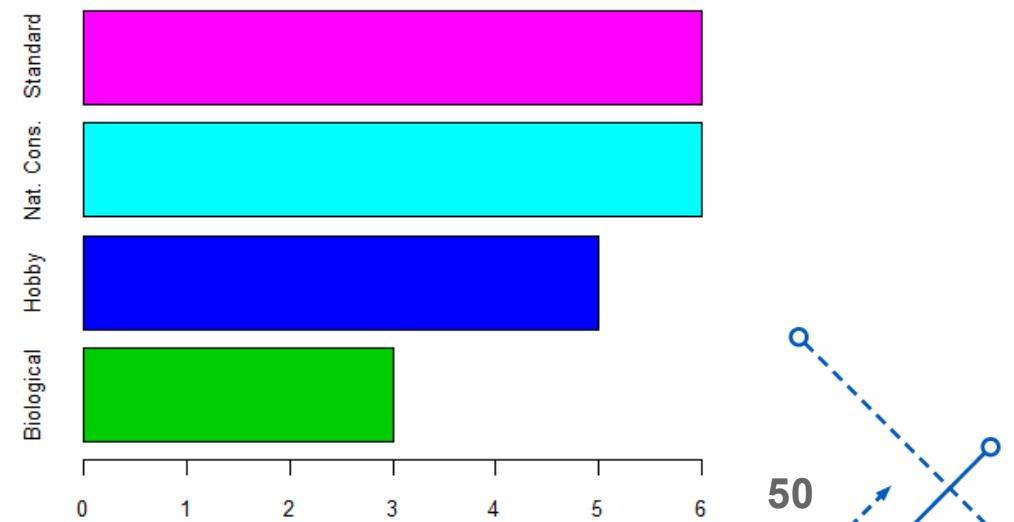
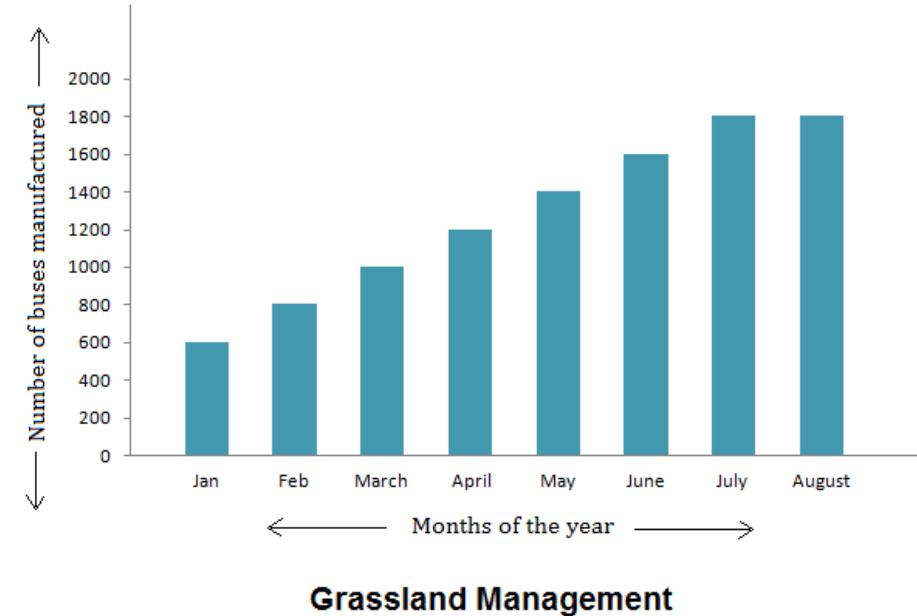
# BARPLOT

- Un BarPlot è una rappresentazione grafica che utilizza barre di lunghezza variabile per mostrare le frequenze o altre misure quantitative di dati categoriali
- Utilizzato per dati categoriali (nominali o ordinali) e per rappresentare aggregazioni di dati.
- Utilizzare i BarPlot
  - **Comparazione:** Ideale per confrontare valori tra diverse categorie (es. vendite per prodotto)
  - **Visualizzazione di Dati Aggregati:** Utile per mostrare medie, somme o conteggi all'interno di categorie
  - **Rappresentazione di Distribuzioni:** I BarPlot possono essere usati per mostrare la distribuzione di dati categoriali



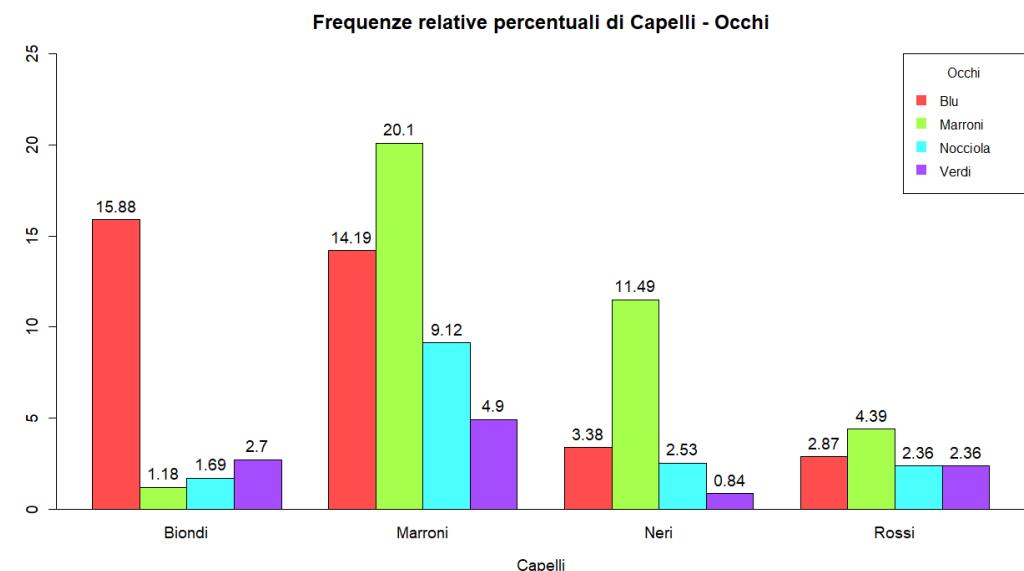
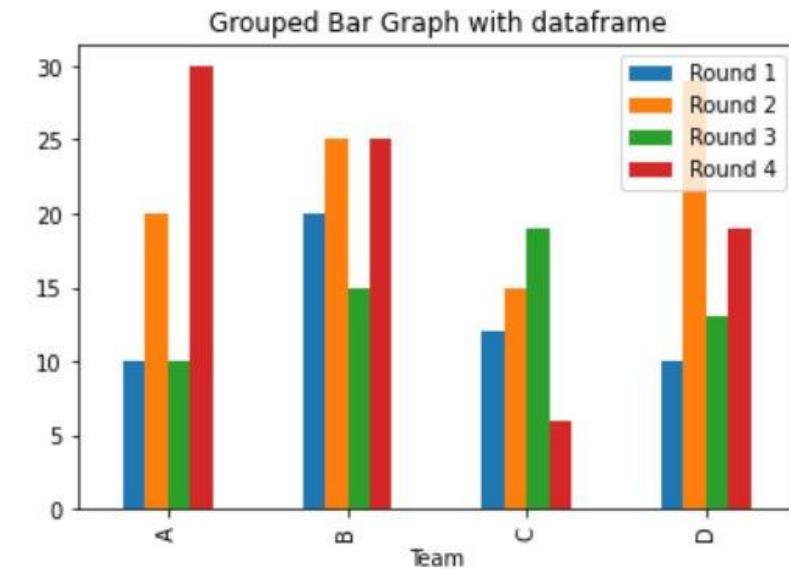
# BARPLOT

- Un BarPlot è una rappresentazione grafica che utilizza barre di lunghezza variabile per mostrare le frequenze o altre misure quantitative di dati categoriali
- Utilizzato per dati categoriali (nominali o ordinali) e per rappresentare aggregazioni di dati.
- Tipo 1:
  - BarPlot Tradizionali



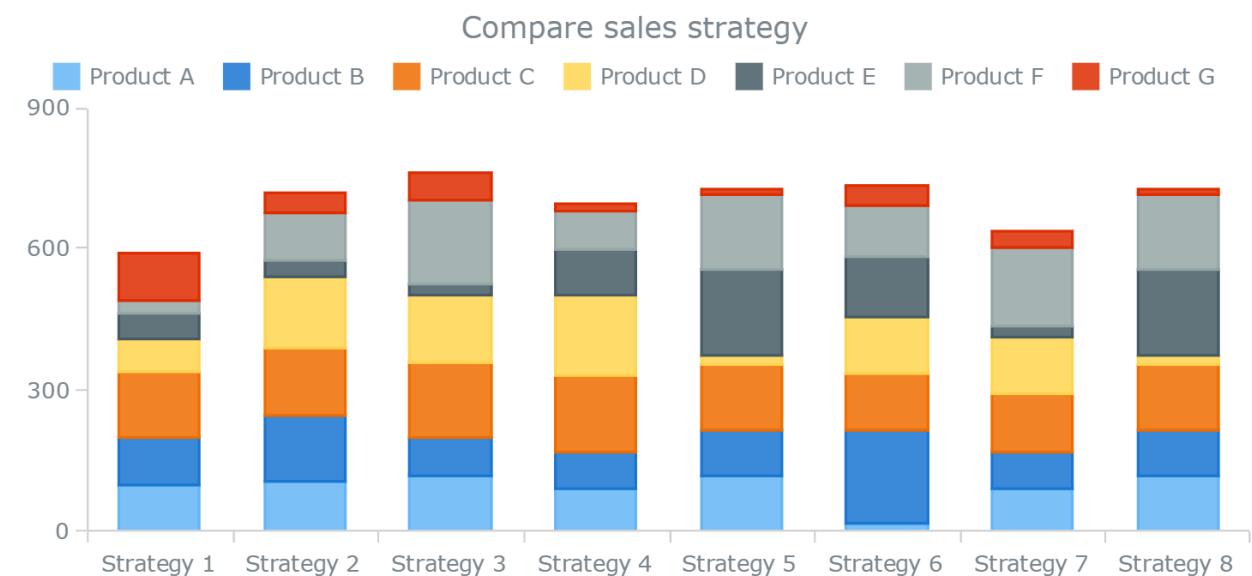
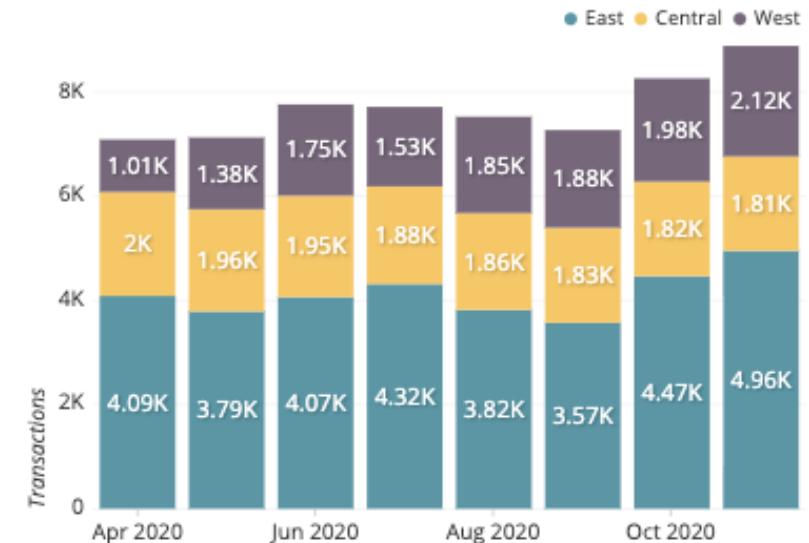
# BARPLOT

- Un BarPlot è una rappresentazione grafica che utilizza barre di lunghezza variabile per mostrare le frequenze o altre misure quantitative di dati categoriali
- Utilizzato per dati categoriali (nominali o ordinali) e per rappresentare aggregazioni di dati.
- Tipo 1:
  - BarPlot Tradizionali
- Tipo 2:
  - Grouped BarPlot



# BARPLOT

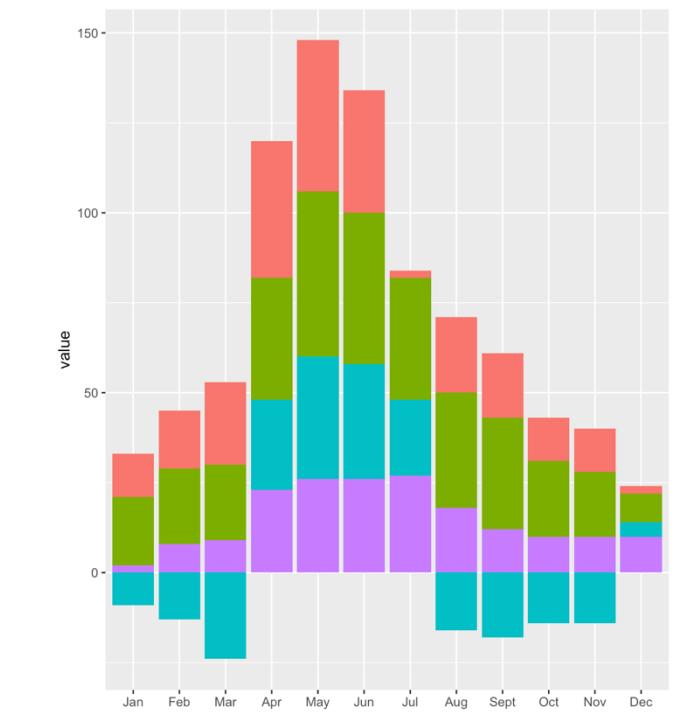
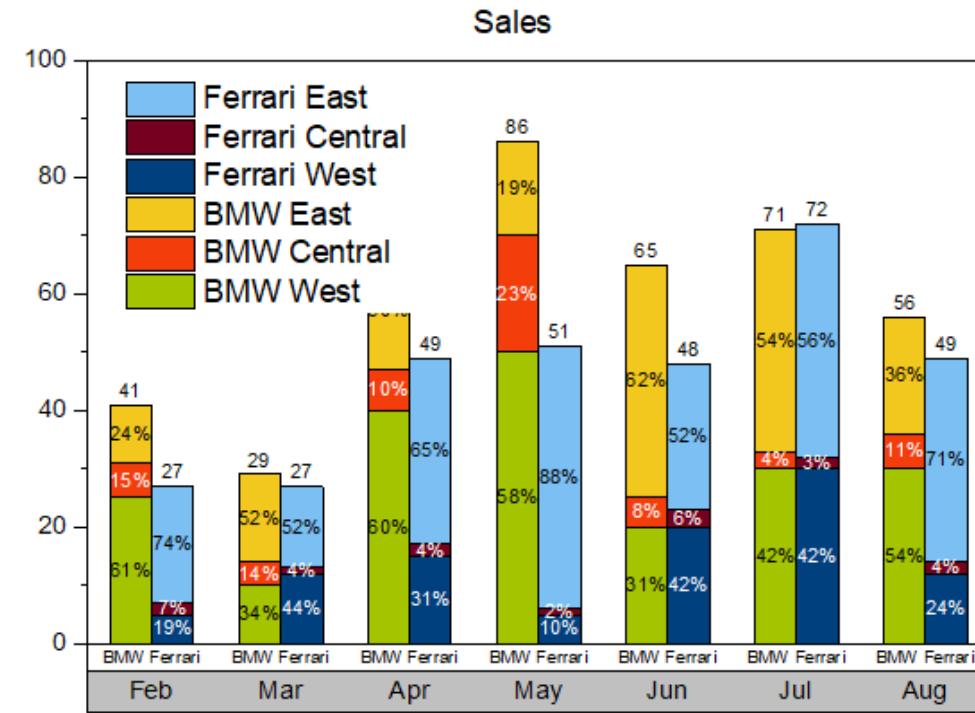
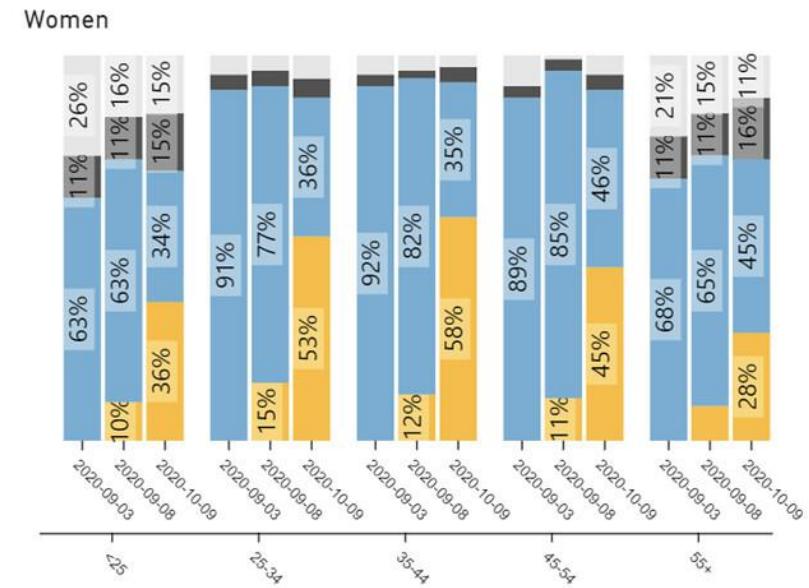
- Un BarPlot è una rappresentazione grafica che utilizza barre di lunghezza variabile per mostrare le frequenze o altre misure quantitative di dati categoriali
- Utilizzato per dati categoriali (nominali o ordinali) e per rappresentare aggregazioni di dati.
- Tipo 1:
  - BarPlot Tradizionali
- Tipo 2:
  - Grouped BarPlot
- Tipo 3:
  - Stacked BarPlot



# BARPLOT

- Un BarPlot è una rappresentazione grafica che utilizza barre di lunghezza variabile per mostrare le frequenze o altre misure quantitative di dati categoriali
- Utilizzato per dati categoriali (nominali o ordinali) e per rappresentare aggregazioni di dati.

- Tipo 1:
  - BarPlot Tradizionali
- Tipo 2:
  - Grouped BarPlot
- Tipo 3:
  - Stacked BarPlot
- ...
- Tipo *n*: ...



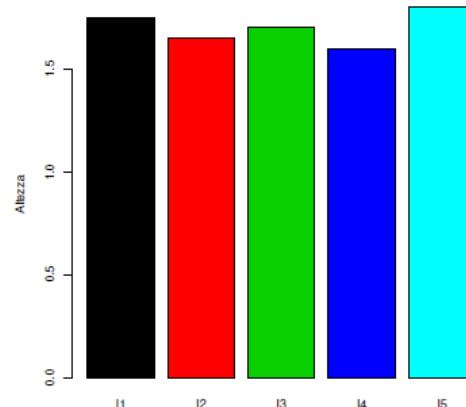
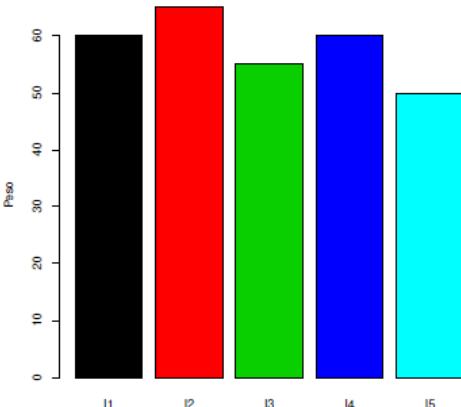
# BARPLOT

- Supponiamo ora di disporre di una **matrice di dati numerici**.
- A partire da essa è possibile creare dei vettori contenenti gli elementi delle singole colonne
  - Utilizzando poi la funzione barplot() è possibile creare dei grafici a barre
- Supponiamo di avere i dati di peso, altezza e presenza di una particolare caratteristica per 5 individui
- Con il seguente codice otteniamo i grafici relativi alle prime due colonne

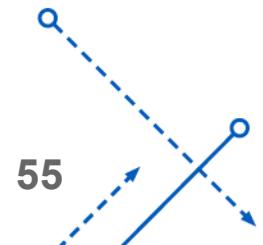
```
> m<-cbind(c(60,65,55,60,50), c(1.75,1.65,1.70,1.60,1.80),  
+ c(1,0,0,1,1))  
> rownames(m)<-c("I1","I2","I3","I4","I5")  
> colnames(m)<-c("Peso","Altezza","Presenza")  
> m  
    Peso   Altezza  Presenza  
I1     60      1.75       1  
I2     65      1.65       0  
I3     55      1.70       0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

# BARPLOT

- Supponiamo ora di disporre di una **matrice di dati numerici**.
- A partire da essa è possibile creare dei vettori contenenti gli elementi delle singole colonne
  - Utilizzando poi la funzione barplot() è possibile creare dei grafici a barre
- Supponiamo di avere i dati di peso, altezza e presenza di una particolare caratteristica per 5 individui
- Con il seguente codice otteniamo i grafici relativi alle prime due colonne



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



# BARPLOT

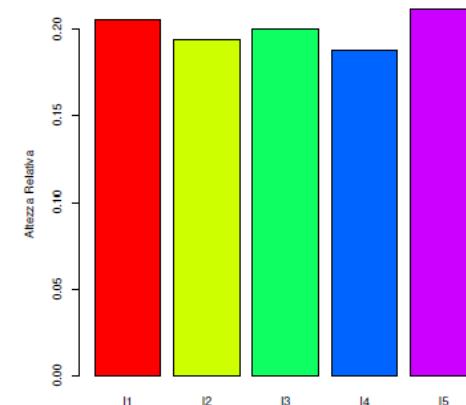
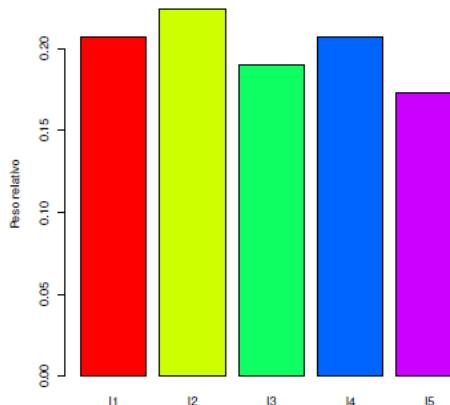
- Le unità di misura in una matrice dei dati sono importanti e spesso occorre procedere ad una standardizzazione dei dati per avere dei numeri puri, ossia privi di unità di misura.
- **Un semplice modo per standardizzare consiste nel dividere gli elementi di ogni colonna per la somma dei valori della colonna presa in considerazione.**
- Da notare che la somma degli elementi di ogni colonna della matrice **mrel** è unitaria

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

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

# GRAFICI PER DATA FRAME

- Le unità di misura in una matrice dei dati sono importanti e spesso occorre procedere ad una standardizzazione dei dati per avere dei numeri puri, ossia privi di unità di misura.
- Un semplice modo per standardizzare consiste nel dividere gli elementi di ogni colonna per la somma dei valori della colonna presa in considerazione.**
- Da notare che la somma degli elementi di ogni colonna della matrice **mrel** è unitaria



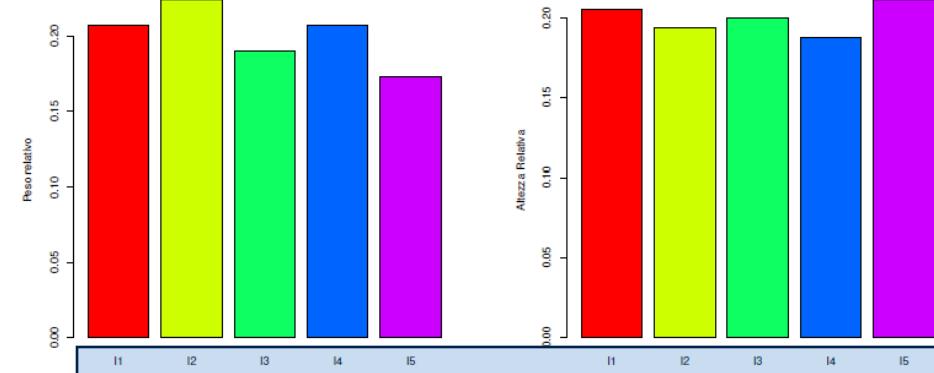
```
> peso<-c(60,65,55,60,50)
> altezza<-c(1.75,1.65,1.70,1.60,1.80)
> presenza<- c(1,0,0,1,1)
>
> mrel<-cbind(peso/sum(peso),altezza/sum(altezza),
+ presenza/sum(presenza))
> colnames(mrel)<-c("Peso","Altezza","Presenza")
> rownames(mrel)<-c("I1","I2","I3","I4","I5")
>
> mrel
      Peso    Altezza 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

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

# GRAFICI

- Se invece si è interessati a grafici a barre per le **percentuali** basta moltiplicare per 100 tutti gli elementi della matrice **mrel**
- Lo stesso grafico di prima si può ottenere usando un data frame invece di una matrice

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



Il parametro **names.arg** definisce un vettore di nomi che appaiono sotto ogni barra del grafico. Esso viene usato anche per settare i colori

# BARPLOT VS GROUPED VS STACKED

- Supponiamo ora di disporre di un insieme di informazioni relative ad una fornitura di frutta che è stata fatta ad un ristorante nel dal 2021 al 2023
  - Vogliamo tenere traccia della quantità di frutta e per ogni tipo di frutta per analizzare i consumi del ristorante

```
# Creazione del dataframe
df <- data.frame(Frutta = c("Mele", "Banane", "Arance", "Pere", "Fragole"),
                  Anno2021 = c(10, 15, 7, 12, 9),
                  Anno2022 = c(8, 12, 10, 14, 6),
                  Anno2023 = c(6, 9, 12, 11, 8))
```

```
df
```

Frutta	Anno2021	Anno2022	Anno2023
<chr>	<dbl>	<dbl>	<dbl>
Mele	10	8	6
Banane	15	12	9
Arance	7	10	12
Pere	12	14	11
Fragole	9	6	8

# BARPLOT VS GROUPED VS STACKED

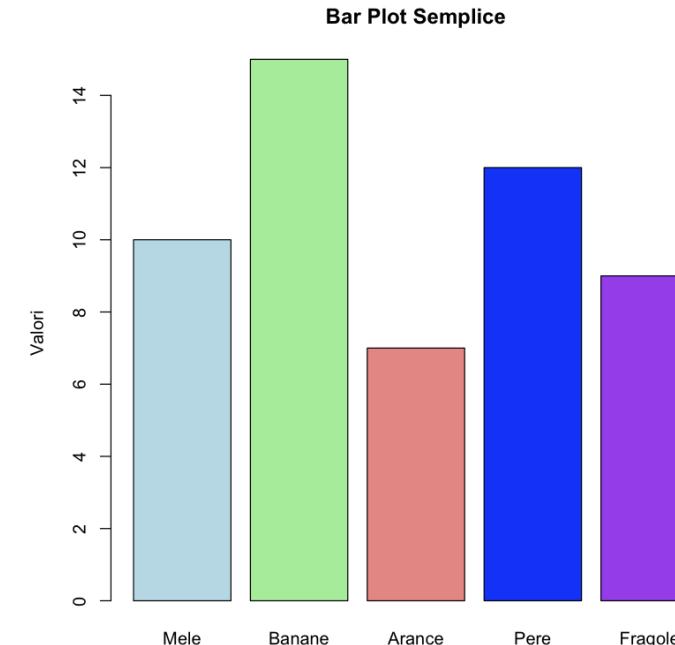
- Supponiamo ora di disporre di un insieme di informazioni relative ad una fornitura di frutta che è stata fatta ad un ristorante nel dal 2021 al 2023
  - Vogliamo tenere traccia della quantità di frutta e per ogni tipo di frutta per analizzare i consumi del ristorante

```
# Creazione del dataframe
df <- data.frame(Frutta = c("Mele", "Banane", "Arance", "Pere", "Fragole"),
                  Anno2021 = c(10, 15, 7, 12, 9),
                  Anno2022 = c(8, 12, 10, 14, 6),
                  Anno2023 = c(6, 9, 12, 11, 8))

df

# Creazione del Bar Plot
barplot(df$Anno2021, col = c("lightblue", "lightgreen", "lightcoral", "blue", "purple"),
         main = "Bar Plot Semplice", ylab = "Valori", names.arg=df$Frutta)
```

A data.frame: 5 x 4			
Frutta	Anno2021	Anno2022	Anno2023
<chr>	<dbl>	<dbl>	<dbl>
Mele	10	8	6
Banane	15	12	9
Arance	7	10	12
Pere	12	14	11
Fragole	9	6	8



60

# BARPLOT VS GROUPED VS STACKED

- Supponiamo ora di disporre di un insieme di informazioni relative ad una fornitura di frutta che è stata fatta ad un ristorante nel dal 2021 al 2023
  - Vogliamo tenere traccia della quantità di frutta e per ogni tipo di frutta per analizzare i consumi del ristorante

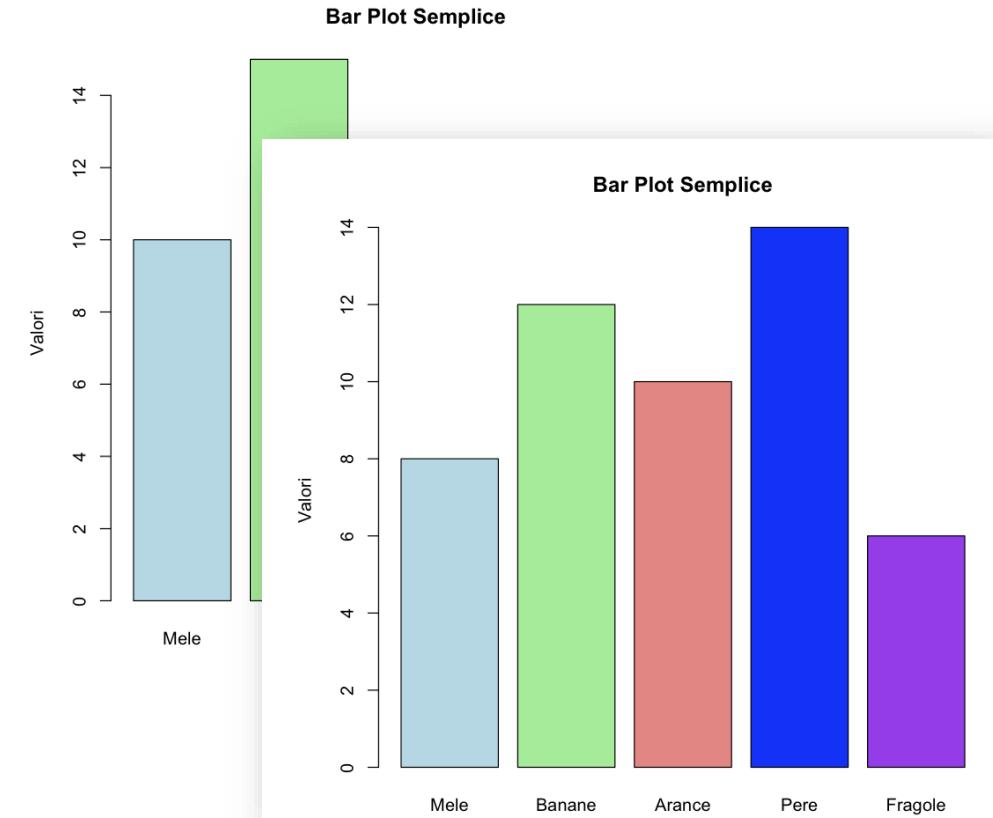
```
# Creazione del dataframe
df <- data.frame(Frutta = c("Mele", "Banane", "Arance", "Pere", "Fragole"),
                  Anno2021 = c(10, 15, 7, 12, 9),
                  Anno2022 = c(8, 12, 10, 14, 6),
                  Anno2023 = c(6, 9, 12, 11, 8))

df

# Creazione del Bar Plot
barplot(df$Anno2021, col = c("lightblue", "lightgreen", "lightcoral", "blue", "purple"),
         main = "Bar Plot Semplice", ylab = "Valori", names.arg=df$Frutta)

barplot(df$Anno2022, col = c("lightblue", "lightgreen", "lightcoral", "blue", "purple"),
         main = "Bar Plot Semplice", ylab = "Valori", names.arg=df$Frutta)
```

A data.frame: 5 × 4			
Frutta	Anno2021	Anno2022	Anno2023
<chr>	<dbl>	<dbl>	<dbl>
Mele	10	8	6
Banane	15	12	9
Arance	7	10	12
Pere	12	14	11
Fragole	9	6	8



# BARPLOT VS GROUPED VS STACKED

- Supponiamo ora di disporre di un insieme di informazioni relative ad una fornitura di frutta che è stata fatta ad un ristorante nel dal 2021 al 2023
  - Vogliamo tenere traccia della quantità di frutta e per ogni tipo di frutta per analizzare i consumi del ristorante

```
# Creazione del dataframe
df <- data.frame(Frutta = c("Mele", "Banane", "Arance", "Pere", "Fragole"),
                  Anno2021 = c(10, 15, 7, 12, 9),
                  Anno2022 = c(8, 12, 10, 14, 6),
                  Anno2023 = c(6, 9, 12, 11, 8))

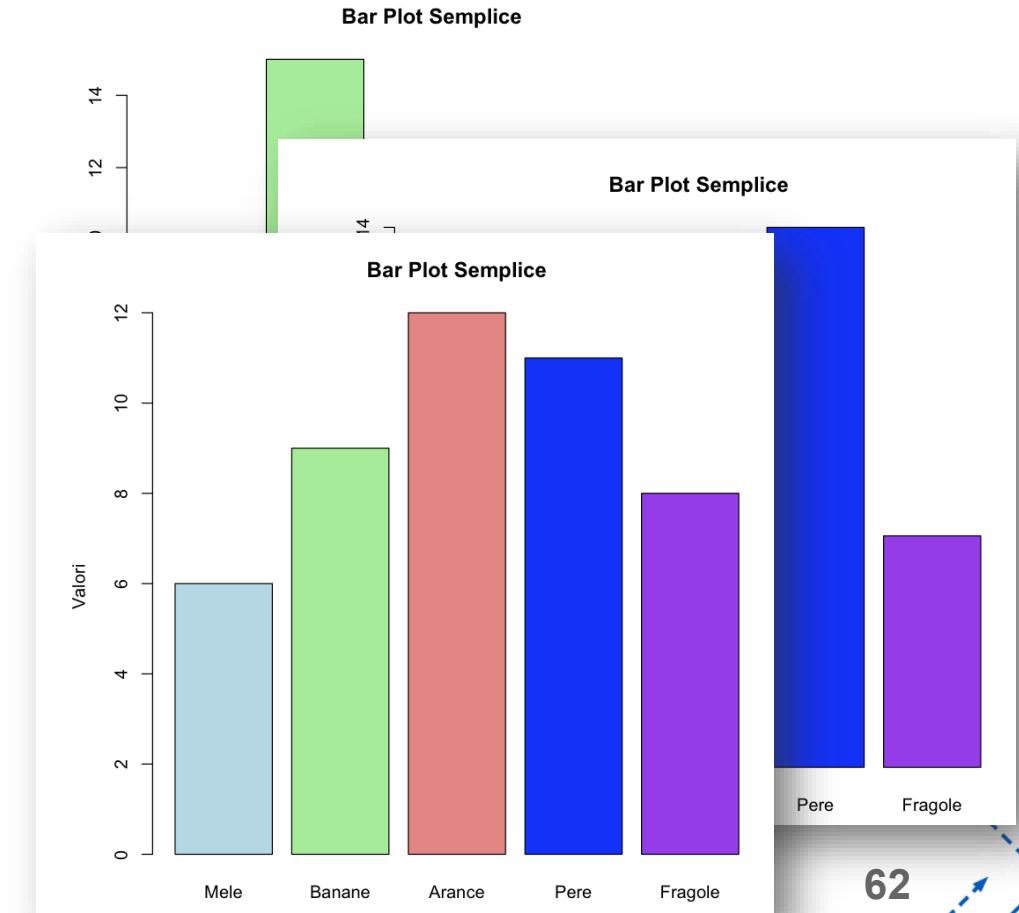
df

# Creazione del Bar Plot
barplot(df$Anno2021, col = c("lightblue", "lightgreen", "lightcoral", "blue", "purple"),
         main = "Bar Plot Semplice", ylab = "Valori", names.arg=df$Frutta)

barplot(df$Anno2022, col = c("lightblue", "lightgreen", "lightcoral", "blue", "purple"),
         main = "Bar Plot Semplice", ylab = "Valori", names.arg=df$Frutta)

barplot(df$Anno2023, col = c("lightblue", "lightgreen", "lightcoral", "blue", "purple"),
         main = "Bar Plot Semplice", ylab = "Valori", names.arg=df$Frutta)

A data.frame: 5 x 4
Frutta  Anno2021  Anno2022  Anno2023
<chr>   <dbl>    <dbl>    <dbl>
  Mele     10       8       6
  Banane   15      12       9
  Arance    7      10      12
  Pere     12      14      11
  Fragole   9       6       8
```



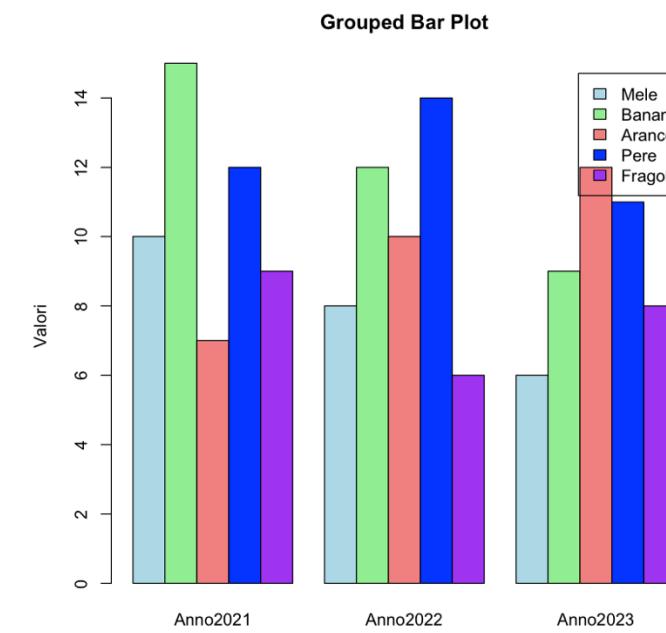
# BARPLOT VS GROUPED VS STACKED

- Supponiamo ora di disporre di un insieme di informazioni relative ad una fornitura di frutta che è stata fatta ad un ristorante nel dal 2021 al 2023
  - Vogliamo tenere traccia della quantità di frutta e per ogni tipo di frutta per analizzare i consumi del ristorante

```
# Creazione del dataframe
df <- data.frame(Frutta = c("Mele", "Banane", "Arance", "Pere", "Fragole"),
                  Anno2021 = c(10, 15, 7, 12, 9),
                  Anno2022 = c(8, 12, 10, 14, 6),
                  Anno2023 = c(6, 9, 12, 11, 8))
df

A data.frame: 5 x 4
Frutta  Anno2021  Anno2022  Anno2023
<chr>   <dbl>     <dbl>     <dbl>
Mele      10        8        6
Banane    15       12        9
Arance     7       10       12
Pere      12       14       11
Fragole    9        6        8

# Creazione del Grouped Bar Plot (corretto)
barplot(as.matrix(df[,2:4]), beside = TRUE,
        col = c("lightblue", "lightgreen", "lightcoral", "blue", "purple"),
        main = "Grouped Bar Plot", ylab = "Valori", legend.text = df$Frutta)
```

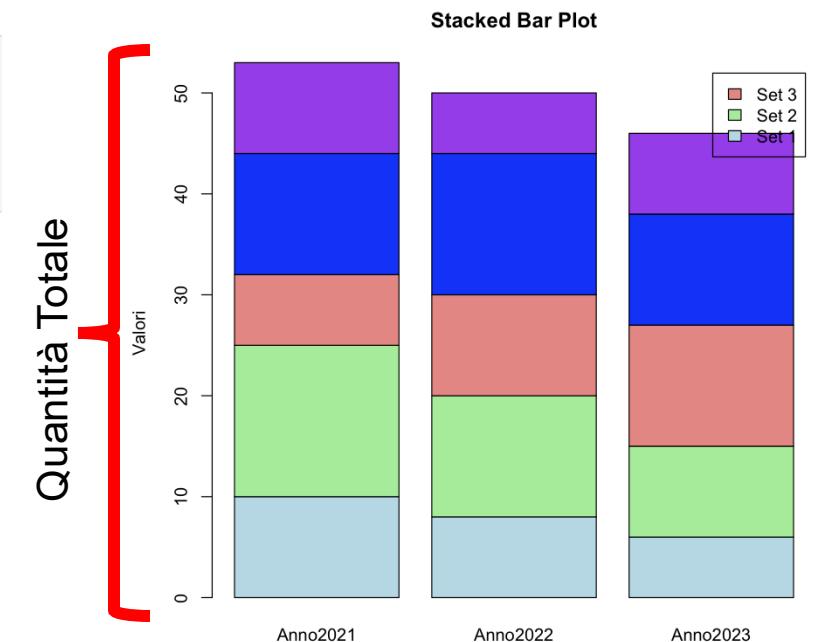


# BARPLOT VS GROUPED VS STACKED

- Supponiamo ora di disporre di un insieme di informazioni relative ad una fornitura di frutta che è stata fatta ad un ristorante nel dal 2021 al 2023
  - Vogliamo tenere traccia della quantità di frutta e per ogni tipo di frutta per analizzare i consumi del ristorante

```
# Creazione dello Stacked Bar Plot
barplot(as.matrix(df[,2:4]),
       col = c("lightblue", "lightgreen", "lightcoral", "blue", "purple"),
       main = "Stacked Bar Plot", ylab = "Valori", legend.text = c("Set 1", "Set 2", "Set 3"))
```

Frutta	A data.frame: 5 x 4		
	Anno2021	Anno2022	Anno2023
<chr>	<dbl>	<dbl>	<dbl>
Mele	10	8	6
Banane	15	12	9
Arance	7	10	12
Pere	12	14	11
Fragole	9	6	8

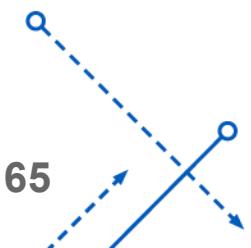


# SCATTERPLOT

---

## Grafici per coppie di variabili: scatterplot

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

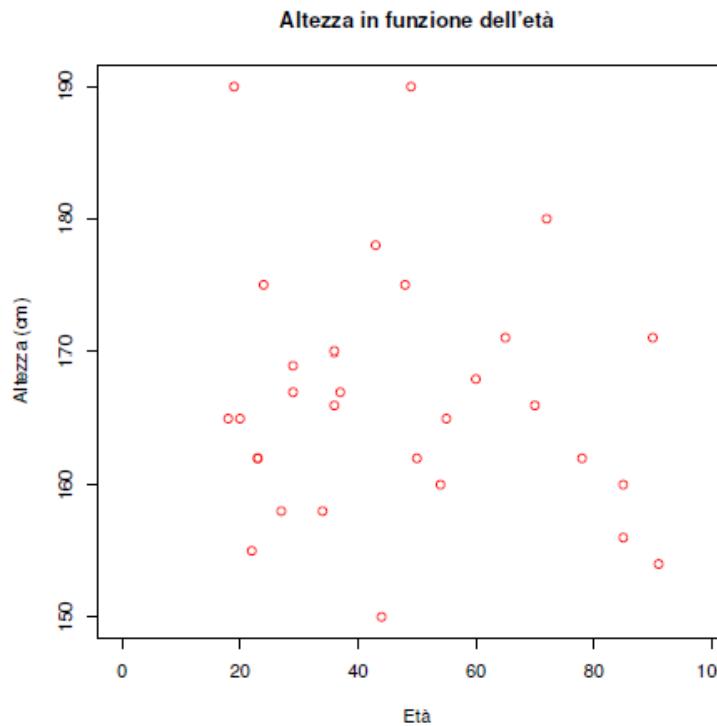


# SCATTERPLOT

## Grafici per coppie di variabili: scatterplot

- Realizziamo lo scatterplot **considerando l'età come variabile indipendente ed l'altezza come variabile dipendente**, disegnando 30 punti in uno spazio euclideo

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



# SCATTERPLOT

---

## Grafici per coppie di variabili: scatterplot

- Definiamo un data frame contenente tutte le informazioni dei 30 individui

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

# SCATTERPLOT

## Grafici per coppie di variabili: scatterplot

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

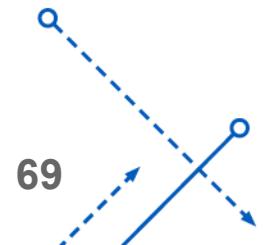
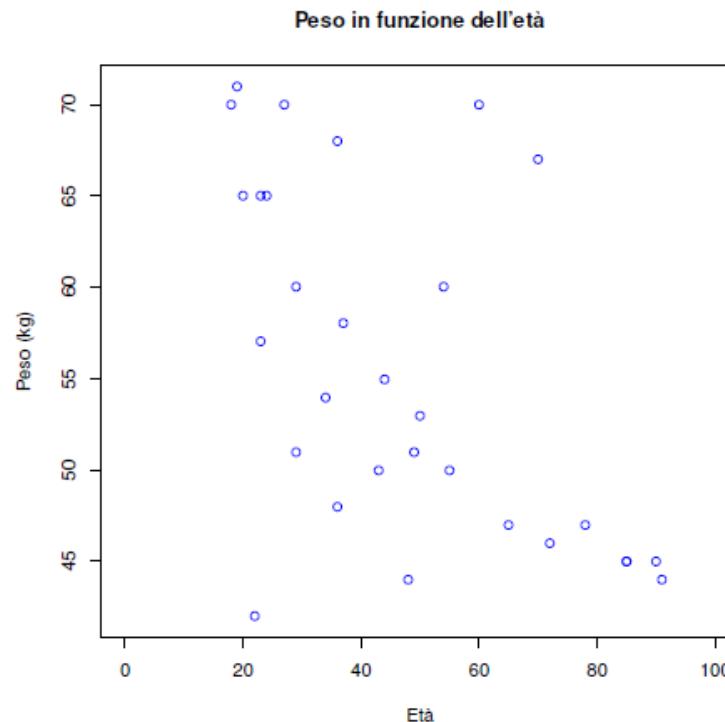
Età	44	55	36	72	29	24	27	50	49	22
Altezza	150	165	170	180	167	175	158	162	190	155
Peso	55	50	48	46	60	65	70	53	51	42
Età	34	29	48	43	54	23	36	60	65	85
Altezza	158	169	175	178	160	162	166	168	171	160
Peso	54	51	44	50	60	65	68	70	47	45
Età	85	91	18	19	20	70	37	23	90	78
Altezza	156	154	165	190	165	166	167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

# SCATTERPLOT

## Grafici per coppie di variabili: scatterplot

- Realizziamo lo scatterplot **considerando l'età come variabile indipendente ed il peso come variabile dipendente**, disegnando 30 punti in uno spazio euclideo

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

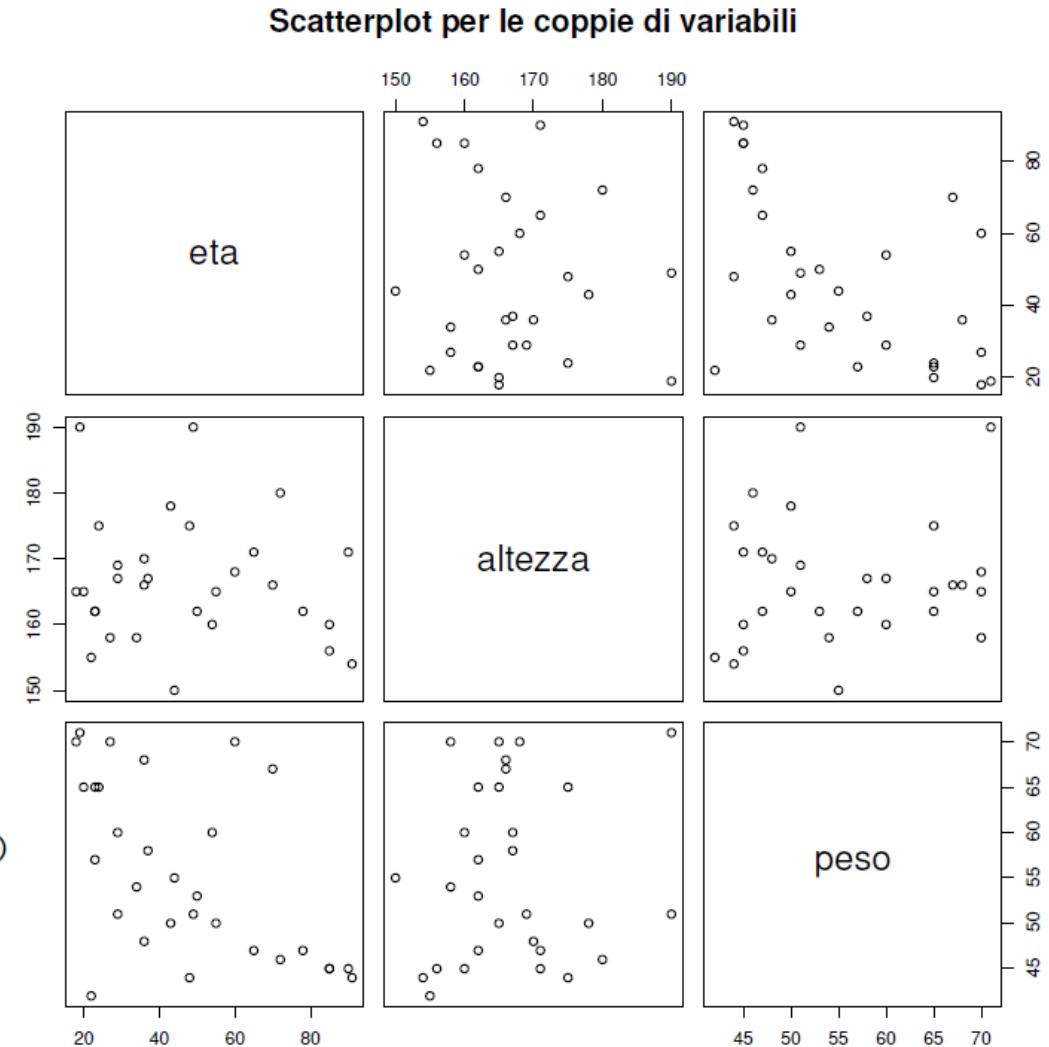


# SCATTERPLOT

## Grafici per coppie di variabili: scatterplot

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

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



# DOMANDE?

