

Lezione 3: Plotting

Davide Evangelista e Dario Lanzoni
`dario.lanzoni3@studio.unibo.it`

Università di Bologna

6 Marzo 2025

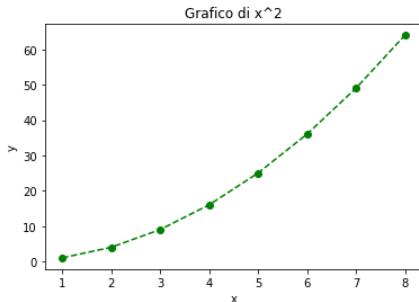
- Per rappresentare i vettori delle osservazioni e controllare come sono distribuiti i dati nel nostro dataset, usiamo Matplotlib.
- Si importa con il comando `import matplotlib.pyplot as plt`

La funzione `plot()` prende come input molti argomenti:

- Un vettore `x` rappresentante le ascisse dei punti da plottare.
- Un vettore `y` rappresentante le ordinate dei punti da plottare.
- `color` che indica il colore del grafico.
- `marker` che indica il marker che rappresenta i punti. [ref]
- `linestyle` che indica il tipo di linea che vogliamo usare. [ref]
- `plt.show()` alla fine del codice permette di visualizzare il grafico. [ref]

Plot Esempio

```
> x = np.arange(1,9)
> y = x**2
> plot(x, y, color='green', marker='o', linestyle='dashed')
> plt.xlabel('x')
> plt.ylabel('y')
> plt.title('Grafico di x^2')
> plt.show()
```



- La funzione `plot()` disegna il nuovo grafico nella stessa figura.

```
> x = np.arange(1,9)
> y = x**2
> y1 = x + 5
> plt.plot(x, y, 'ko')
> plt.plot(x,y1,'ro')
> plt.xlabel('x')
> plt.ylabel('y')
> plt.legend(['black','red'])
> plt.title('Esempio')
> plt.show()
```

Plot(s)

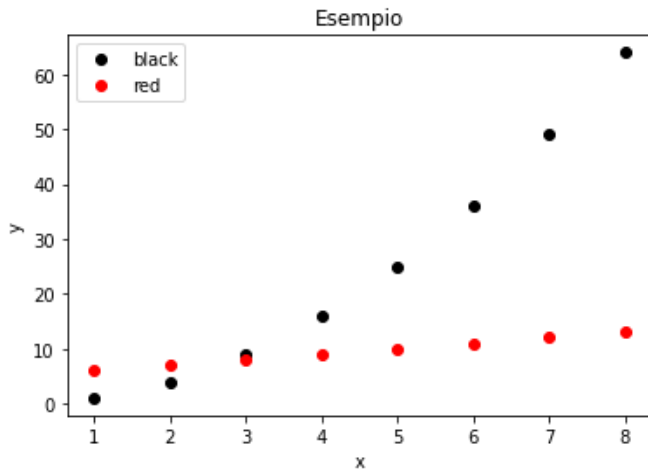


Figure e Axes

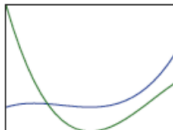
- Nella libreria Matplotlib, la rappresentazione di più grafici insieme viene organizzata attraverso Figure e Axes.
- Una **Figure** è una finestra o una regione di disegno che può contenere uno o più grafici.
- Gli **Axes** sono i sistemi di coordinate all'interno della Figure, utilizzati per disegnare uno o più grafici, eventualmente di diverso tipo (Axes.plot, Axes.step, Axes.bar, Axes.hist, Axes.scatter...).
- Una Figure può contenere più Axes, e ogni Axes rappresenta un singolo grafico.
- Per accedere e manipolare gli Axes all'interno di una figura, si usa il comando:

```
> ax[i,j]
```

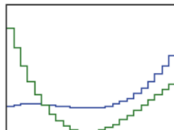
dove *i* rappresenta l'indice della riga e *j* l'indice della colonna, in una griglia di Axes.

Tipi di Axes plot

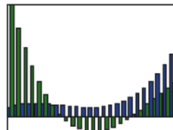
Axes.plot



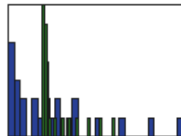
Axes.step



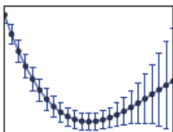
Axes.bar



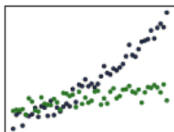
Axes.hist



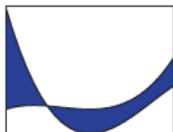
Axes.errorbar



Axes.scatter



Axes.fill_between



Axes.quiver

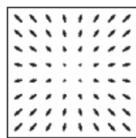


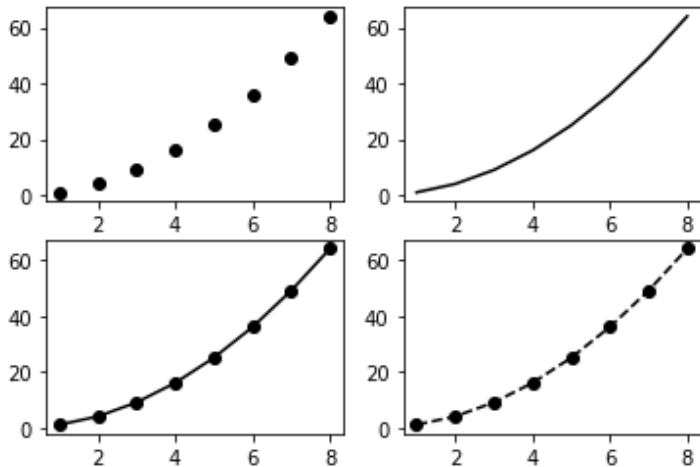
Figura: Tipi di plot che si possono fare con Axes in una figura.

- Per disegnare più plot nella stessa schermata si usa la funzione `plt.subplots(nrows=rnum, ncols=cnum)`, dove `rnum` rappresenta il numero di plot che si vuole visualizzare in riga, mentre `cnum` rappresenta il numero di plot che si vuole visualizzare in colonna.

```
> fig, ax = plt.subplots(nrows=2, ncols=2)
> ax[0,0].plot(x, y, 'ko')
> ax[0,1].plot(x, y, 'k-')
> ax[1,0].plot(x, y, 'k-o')
> ax[1,1].plot(x, y, 'k--o')

> plt.show()
```

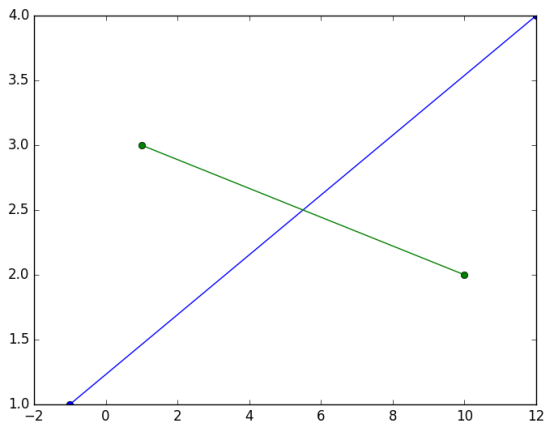

Subfigure



- Per disegnare una retta si usa il comando `axline(xy1, xy2=None, slope=None)`
 - `xy1, xy2` punti per cui passa la retta.
 - `slope` coefficiente angolare retta (se non si specifica `xy2`)
- Si possono disegnare i segmenti prendendo in input le coordinate dei due punti estremi.

```
> x1, y1 = [-1, 12], [1, 4]
> x2, y2 = [1, 10], [3, 2]
> plt.plot(x1, y1, x2, y2, marker = 'o')
> plt.show()
```

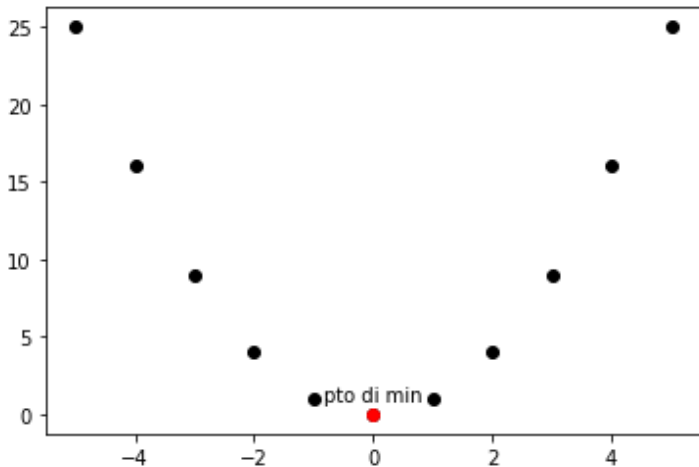
Segmenti e linee



La funzione `text()` serve per aggiungere dei label sul grafico.
Utile quando si vuole aggiungere il nome ad uno specifico punto del grafico.

```
> x = np.arange(-5,6)
> y = x**2

> plt.plot(x,y,'ko')
> plt.plot(0,0,'ro')
> plt.text(0,0.8, 'pto di min', horizontalalignment='center')
> plt.show()
```



Estrazioni casuali: `random.choice()`

- Dato un vettore `v` la funzione `random.choice()` estrae da `v` (con reinserimento) `n` valori.

```
> v = np.arange(1, 11)  
> np.random.choice(v, 10)  
[1] 2 9 5 9 1 9 2 6 5 9
```
- Per estrarli senza reinserimento

```
> np.random.choice(v,8, replace=False)  
[1] 2 9 1 7 3 6 8 4
```
- È possibile specificare la probabilità di estrarre ogni singolo elemento.

```
> v = np.arange(1,4)  
> np.random.choice(v, 10, replace=True, p=(0.7 , 0.2,  
0.1))  
[1] 1 1 2 3 1 1 2 3 1 1
```

Estrazioni casuali: normale Gaussiana

La funzione `random.normal()` prende in input

- la media
- la deviazione standard
- la lunghezza del vettore di output

restituisce un vettore che ha dimensione scelta e contiene elementi estratti con una distribuzione Gaussiana con media e deviazione assegnate.

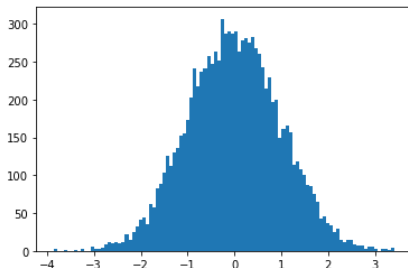
```
> x = random.normal(loc=0.0, scale=1.0, size=10)
> x # Normale standard
[1] 0.35315767 0.81645816 0.27591868 -1.09573384
1.63935188 0.80787089 -0.07381423 0.63730397
-0.61066481 -0.89062814
```

- `random.randn()` stessa cosa senza specificare media e deviazione standard (0,1).

Istogrammi

Per disegnare un istogramma si utilizza la funzione `hist()` che prende come input un vettore (del quale plottare le frequenze) e un parametro opzionale `bins`, che indica in quanti intervalli dividere i valori dell'array in input.

```
> x = np.random.randn(10000)
> plt.hist(x , bins=100)
```



Barplot & Piechart

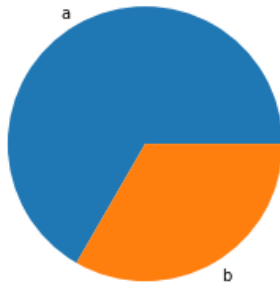
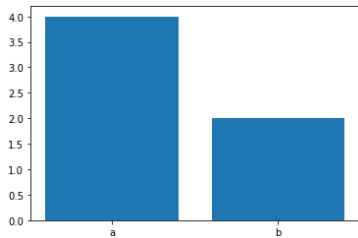
Per rappresentare un vettore di tipo qualitativo si utilizzano le seguenti funzioni

- `bar()` per rappresentare dei grafici a barre
- `piechart()` per rappresentare dei grafici a torta

Prima è necessario contare le occorrenze di ogni possibile valore.

```
> x = np.array(("a" , "a" , "a" , "b" , "a" , "b" ))  
> unique = np.unique(x)  
> count = [np.sum(x==el) for el in unique]  
> plt.bar(unique,count)  
> plt.pie(count,labels=unique)
```

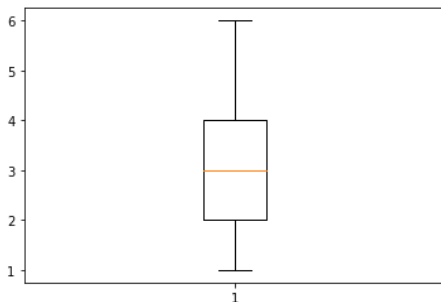
Barplot & Piechart



Boxplot

Il `boxplot()`, risulta utile per visualizzare il range dei dati.

```
> x = np.array((3, 3, 3, 2, 1, 4, 6, 2, 4, 1, 6, 4, 2, 1,  
5, 4))  
> plt.boxplot(x)
```



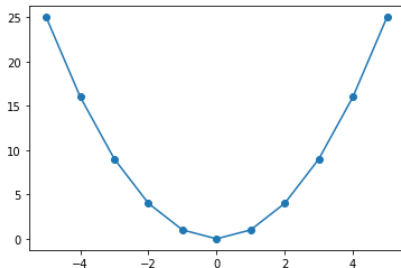
- Plot con Pyplot molto flessibili ma la sintassi è molto verbosa.
- La libreria Seaborn permette di creare grafici più complessi.
- Ottima integrazione con i DataFrame in Pandas
- Per una lista completa dei possibili grafici si veda la Python-Graph Gallery.

```
import seaborn as sns
```

Esempio Seaborn

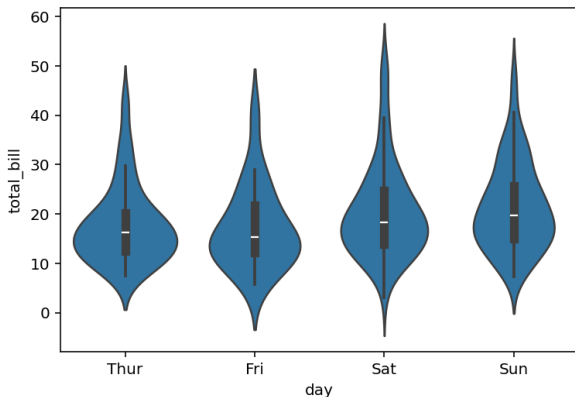
Disegniamo il grafico della funzione $f(x) = x^2$ sull'intervallo $[-5, 5]$.
Per passare i dati a ggplot2 è necessario salvarli in un dataframe.

```
> df = pd.DataFrame({'x_axis': np.arange(-5,6),  
  'y_axis': np.arange(-5,6)**2})  
> plt.plot('x_axis', 'y_axis',  
data=df, linestyle='-', marker='o')  
> plt.show()
```



Esempio Seaborn (I)

```
# load data  
> df = sns.load_dataset('tips')  
# plot  
> sns.violinplot(x=df["day"], y=df["total_bill"], data=df)
```



Esempio Seaborn (II)

- Il *violin plot* visto sopra è utile per mostrare la distribuzione dei dati in un dataset.
- Nello specifico, scegliamo le variabili da analizzare attraverso le espressioni `x=df[...]` e `y=df[...]`.
- Una volta scelte le variabili, visualizziamo la distribuzione di densità della variabile `y` in corrispondenza delle categorie della variabile `x`.
- All'interno della densità è presente un boxplot, che rappresenta sinteticamente i quartili, la mediana e i valori estremi della distribuzione.
- Le distribuzioni a destra e a sinistra dei boxplot sono le stesse, ma vengono visualizzate in questo modo per rendere più chiara la distribuzione dei dati.