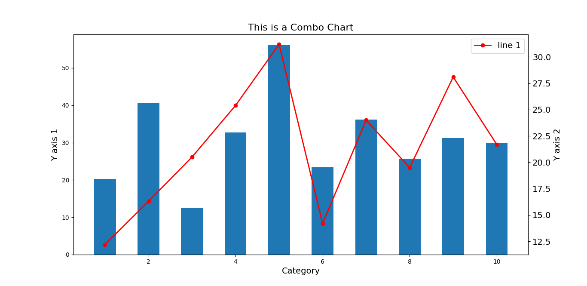
## **Kết hợp nhiều biểu đồ khác loại**

### Giới thiệu

Trong thực tế, đôi khi chúng ta cần biểu diễn nhiều thông tin cùng lúc trên nhiều bộ dữ liệu bằng cách sử dụng kết hợp nhiều biểu đồ khác nhau. Do đó, việc biểu diễn các biểu đồ khác nhau cùng lúc sẽ giúp tiết kiệm không gian trình bày, giúp người xem dễ dàng nắm bắt thông tin tổng hợp trên cùng một hình ảnh (ví dụ dashboard) tăng khả năng truyền tải thông tin và giải thích thông tin.

### Nhiều biểu đồ khác loại trong cùng một khung nhìn



* Dạng biểu này rất dễ gây rắc rối, phức tạp cho người xem nên chỉ sử dụng khi các loại biểu đồ có tác dụng bổ trợ thông tin cho nhau cùng lúc.
* Nếu 2 biểu đồ biểu diễn các thuộc tính khác nhau và không có cùng tỉ lệ trục Y thì nên biểu diễn tối đa 2 biểu đồ và cần sinh thêm trục Y phụ bên phải.

#### Ý nghĩa sử dụng

* Trực quan các bộ dữ liệu nhằm mô tả các đối tượng khác nhau nhưng cùng có chung mục tiêu mô tả hoặc bổ trợ thông tin cho nhau cùng lúc.
* Giúp người xem dễ dàng nắm thông tin tổng hợp, giải thích thông tin của một biểu đồ bằng thông tin từ biểu đồ khác.

#### Sử dụng thư viện Matplotlib

* Import thư viện cần thiết

import matplotlib.pyplot as plt

* Lấy các bộ dữ liệu cần thiết:

x = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]  
y1 = [12.2, 16.3, 20.5, 25.4, 31.2, 14.2, 24, 19.5, 28.1, 21.7]  
y2 = [2000, 4000, 1200, 3200, 5600, 2300, 3600, 2500, 3100, 3000]

* Vẽ biểu đồ thứ nhất

plt.bar(x, y2, label = 'line 2', width = 0.5)

* Tạo Axes để vẽ biểu đồ thứ hai

axes1 = plt.gca()  
axes2 = axes1.twinx()

* Vẽ biểu đồ thứ hai

axes2.plot(x, y1, label = 'line 1', linewidth = 2, c = 'r', marker = 'o')

* Có thể thêm nhiều biểu đồ khác vào các vị trí khác nhau

fig = plt.gcf()  
axes3 = fig.add\_axes([left, bottom, width, height])  
axes3.plot()

Ví dụ:

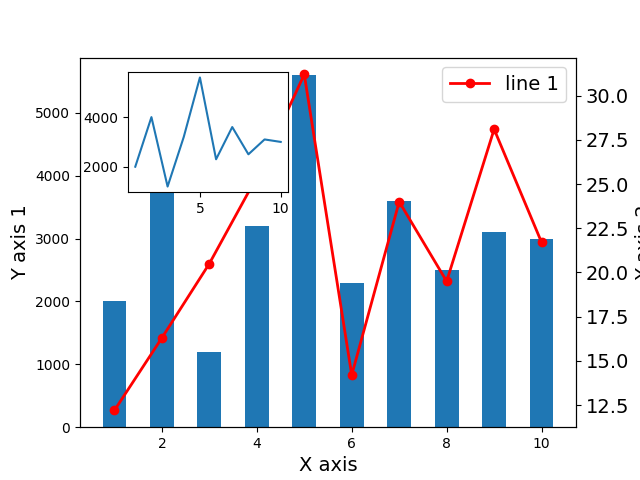
axes3 = fig.add\_axes([0.2, 0.6, 0.25, 0.25])  
axes3.plot(x, y2)

Lưu ý: cân bằng giữa việc thể hiện cùng lúc nhiều thông với mức độ rắc rối, phức tạp về mặt hình ảnh mà người xem cần xử lý.

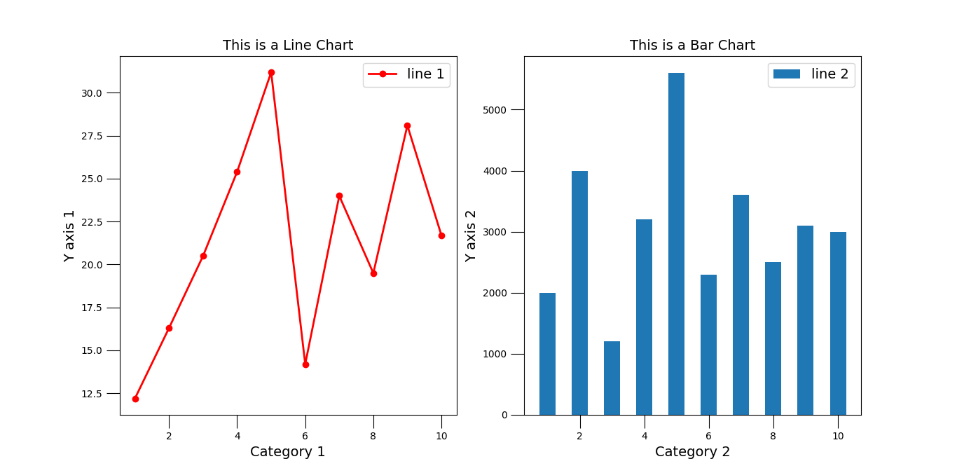
* Cài đặt các thuộc tính

axes1.set\_xlabel('X axis', fontsize = 14)  
axes1.set\_ylabel('Y axis 1', fontsize = 14)  
axes2.set\_ylabel('Y axis 2', fontsize = 14)  
plt.legend(fontsize = 14)  
plt.xticks(fontsize = 14)  
plt.yticks(fontsize = 14)  
plt.title('This is a Combo Chart', fontsize = 16)  
plt.show()

Kết quả:



### Nhiều biểu đồ khác loại trên cùng một Figure



* Có thể biểu diễn đa dạng các bộ dữ liệu và thể hiện nhiều thông tin cùng lúc trên một hình ảnh.
* Được ứng dụng rộng rãi trong các hệ thống giám sát, quản lý tập trung, các màn hình dashboard.
* Tuy nhiên, việc biểu diễn quá nhiều biểu đồ cùng lúc cũng có thể gây khó khăn cho người xem trong việc xử lý và trích xuất thông tin. Do đó, cần chọn các loại biểu đồ và cách bố trí phù hợp để tránh gây rắc rối cho người xem.
  + Các biểu đồ cùng loại, mô tả cùng thuộc tính có thể bố trí trên cùng một cột để người xem dễ dàng so sánh.
  + Các biểu đồ khác loại nhưng có tác dụng suy diễn thông tin, bổ trợ cho nhau có thể trình bày trên cùng một hàng.
  + Các dữ liệu cùng thuộc tính, có tính chu kì hoặc mô tả các đối tượng khác nhau, có thể bố trí trên cùng một cột để dễ dàng so sánh, tìm quy luật.
  + Các dữ liệu dạng time series có thể bố trí cùng một hàng.

#### Ý nghĩa sử dụng

* Trực quan hóa nhiều bộ dữ liệu mô tả nhiều thuộc tính cùng lúc.
* Giúp người xem dễ dàng so sánh giá trị, xu hướng, độ chênh lệch, giá trị thuộc tính giữa các bộ dữ liệu.
* Có thể dùng để kiểm tra tính quy luật, tính đồng nhất, và giải thích thông tin giữa các bộ dữ liệu.

#### Sử dụng thư viện Matplotlib

* Import thư viện cần thiết

import matplotlib.pyplot as plt

* Lấy các bộ dữ liệu:

x1 = ['P1','P2','P3','P4','P5']  
y1 = [12.2, 16.3, 20.5, 25.4, 31.2]  
y2 = [20.3, 22.6, 31.4, 33.7, 39.2]  
y3 = [16.3, 17.6, 24.4, 26.7, 33.2]

* Khai báo ma trận các Axes

fig, ax = plt.subplots(r, c)

trong đó:

r: số hàng

c: số cột

ví dụ:

fig, ax = plt.subplots(2, 2)

* Vẽ các biểu đồ trên mỗi Axes

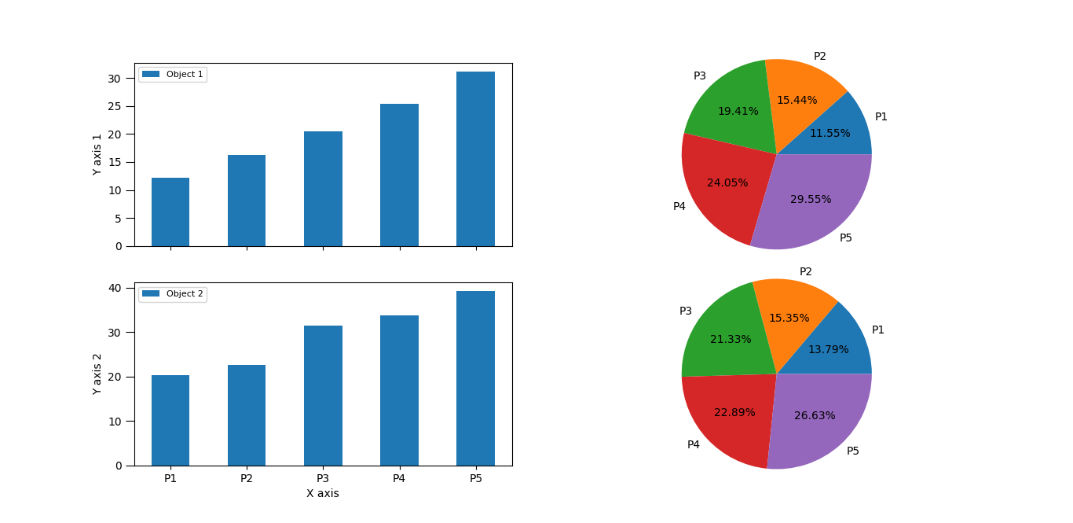
ax[i].plot(x1, y1, label = 'Object 1')

Nếu ax là một ma trận thì truy cập từng phần tử như đối với ma trận thông thường (ví dụ a[i][j]).

Ví dụ:

ax[0][0].bar(x1, y1, label = 'Object 1', width = 0.5)  
ax[1][0].bar(x1, y2, label = 'Object 2', width = 0.5)  
ax[0][1].pie(y1, labels = x1, autopct='%1.2f%%', radius=1.3)  
ax[1][1].pie(y2, labels = x1, autopct='%1.2f%%', radius=1.3)  
ax[0][0].set\_ylabel('Y axis 1', fontsize = 10)  
ax[1][0].set\_xlabel('X axis', fontsize = 10)  
ax[1][0].set\_ylabel('Y axis 2', fontsize = 10)  
  
ax[0][0].legend(fontsize = 8, loc = 'upper left')  
ax[0][0].tick\_params(axis='y', size = 8)  
ax[0][0].set\_xticklabels([])  
  
ax[1][0].legend(fontsize = 8, loc = 'upper left')  
ax[1][0].tick\_params(axis='y', size = 8)  
ax[0][0].set\_xticklabels([])  
  
plt.show()

Kết quả:



### Tổng kết

Qua bài đọc này, chúng ta đã tìm hiểu được phương pháp trực quan hóa dữ liệu bằng cách kết hợp nhiều loại biểu đồ khác nhau trên một Axes hay trên một Figure nhằm tăng khả năng so sánh, tìm quy luật, trích xuất và giải thích thông tin cho người xem.