CHƯƠNG 3:TRỰC QUAN HOÁ DỮ LIỆU

Môn học: Nhập môn Khoa học Dữ liệu

Giảng viên: Nguyễn Kiều Linh
Email: linhnk@ptit.edu.vn
Học viện Công nghệ Bưu chính Viễn thông

Hà Nội, năm 2023
http://www.ptit.edu.vn

Trực quan hoá dữ liệu

- **Trực quan hóa dữ liệu** (Data visualization) là thể hiện dữ liệu thành các dạng đồ họa như đồ thị, biểu đồ hay sử dụng các phương pháp, công cụ khác nhau để minh họa dữ liệu được tốt nhất.
- Mục đích là biến nguồn dữ liệu thành thông tin được thể hiện một cách trực quan, dễ quan sát, dễ hiểu, truyền đạt rõ ràng những hiểu biết đầy đủ (insights) từ dữ liệu đến người xem, người đọc.

Trực quan hoá dữ liệu

Graphical excellence is that which gives to the viewer the greatest number of ideas in the shortest time with the least ink in the smallest space.

Quote from The Visual Display of Quantitative Information by Edward R. Tufte

Trực quan hoá dữ liệu

Một số cách trình bày dữ liệu trực quan bằng các biểu đồ sau:

- Đồ thị dạng đường thẳng
- Đồ thị điểm rời rạc
- Trực quan hóa lỗi
- Đồ thị đường viền
- Histograms và mật độ
- Đồ thị ba chiều
- Dữ liệu địa lý

Thư viện trực quan hóa dữ liệu Matplotlib

- Matplotlib là thư viện trực quan hóa dữ liệu được xây dựng trên mảng NumPy và được thiết kế để hoạt động với ngăn xếp SciPy. Nó được John Hunter đưa ra vào năm 2002, cho phép vẽ sơ đồ kiểu MATLAB tương tác thông qua gnuplot từ dòng lệnh IPython.
- Một trong những tính năng quan trọng nhất của Matplotlib là khả năng hoạt động tốt với đa nền tảng nhiều hệ điều hành, điều này dẫn đến một số lượng lớn người dùng với sự phổ biến trong thế giới khoa học dữ liệu sử dụng.

Nạp thư viện matplotlib

Sử dụng viết tắt tiêu chuẩn khi nạp thư viện Matplotlib: *plt* bao gồm nhiều hàm sẽ được sử dụng thường xuyên trong suốt chương này.

```
import matplotlib as mpl
import matplotlib.pyplot as plt
```

Đồ thị dạng đường

- Đồ thị dạng đường thẳng có lẽ là đồ thị đơn giản nhất trong tất cả các đồ thị.
- Đầu tiên là việc thiết Notebook để vẽ đồ thị và nhập các hàm sẽ thực hiện bằng các câu lệnh dưới đây, sau đó tạo một hình - *figure* và một trục - *ax*:

```
In[1]: %matplotlib inline
       import matplotlib.pyplot as plt
       plt.style.use('seaborn-whitegrid')
       import numpy as np
In[2]: fig = plt.figure()
       ax = plt.axes()
```

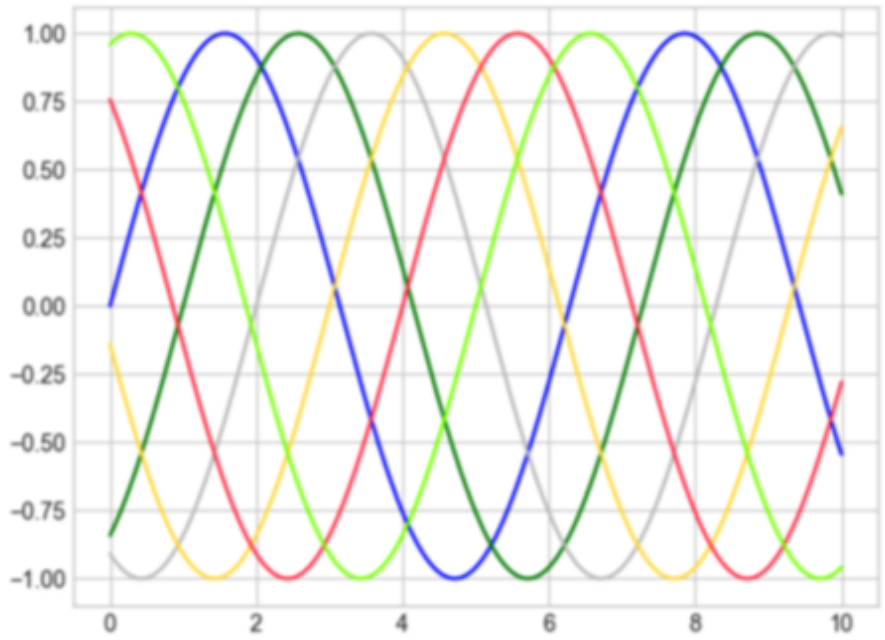
Điều chỉnh màu và kiểu dạng của đồ thị

Điều chỉnh có thể thực hiện đối với một biểu đồ là màu sắc (*colors*) và kiểu dáng của đồ thị (*styles*). Hàm *plt.plot()* chứa các đối số sử dụng để chỉ định các điều chỉnh này. Với màu sử dụng từ khóa *color*. Trong trường hợp không khai báo một màu cụ thể thì Matplotlib sẽ tự động lựa chọn một tập các màu mặc định.

```
plt.plot(x, np.sin(x - 0), color='blue') #Mau cu the theo ten
plt.plot(x, np.sin(x - 1), color='g') #Ma mau bang viet tat (rgbcmyk)
plt.plot(x, np.sin(x - 2), color='0.75') #Thang xam - Grayscale tu 0 den 1
plt.plot(x, np.sin(x - 3), color='#FFDD44') #Ma Hex (RRGGBB tu 00 - FF)
plt.plot(x, np.sin(x - 4), color=(1.0,0.2,0.3)) #RGB, gia tri giua 0 va 1
plt.plot(x, np.sin(x - 5), color='chartreuse'); #Ten mau HTML duoc ho tro
```

Điều chỉnh màu và kiểu dạng của đồ thị

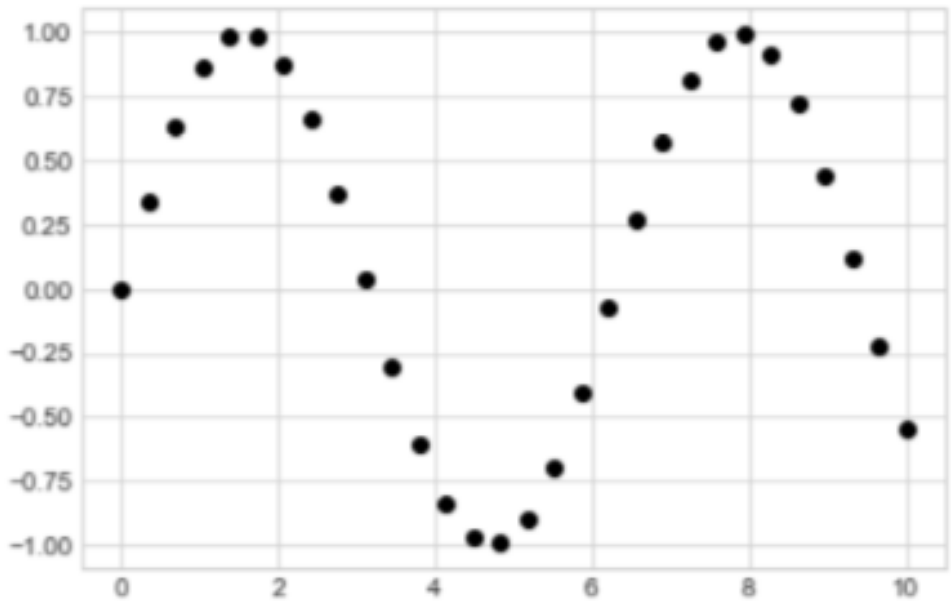
```
plt.plot(x, np.sin(x - 0), color='blue') #Mau cu the theo ten
plt.plot(x, np.sin(x - 1), color='g') #Ma mau bang viet tat (rgbcmyk)
plt.plot(x, np.sin(x - 2), color='0.75') #Thang xam - Grayscale tu 0 den 1
plt.plot(x, np.sin(x - 3), color='#FFDD44') #Ma Hex (RRGGBB tu 00 - FF)
plt.plot(x, np.sin(x - 4), color=(1.0,0.2,0.3)) #RGB, gia tri giua 0 va 1
plt.plot(x, np.sin(x - 5), color='chartreuse'); #Ten mau HTML duoc ho tro
```



Đồ thị điểm rời rạc

Đồ thị điểm rời rạc các điểm thay vì nối với nhau bằng đoạn thì được biểu diễn riêng lẻ bằng dấu chấm, hình tròn hoặc hình dạng khác bằng phương thức *plt.plot/ax.plot*:

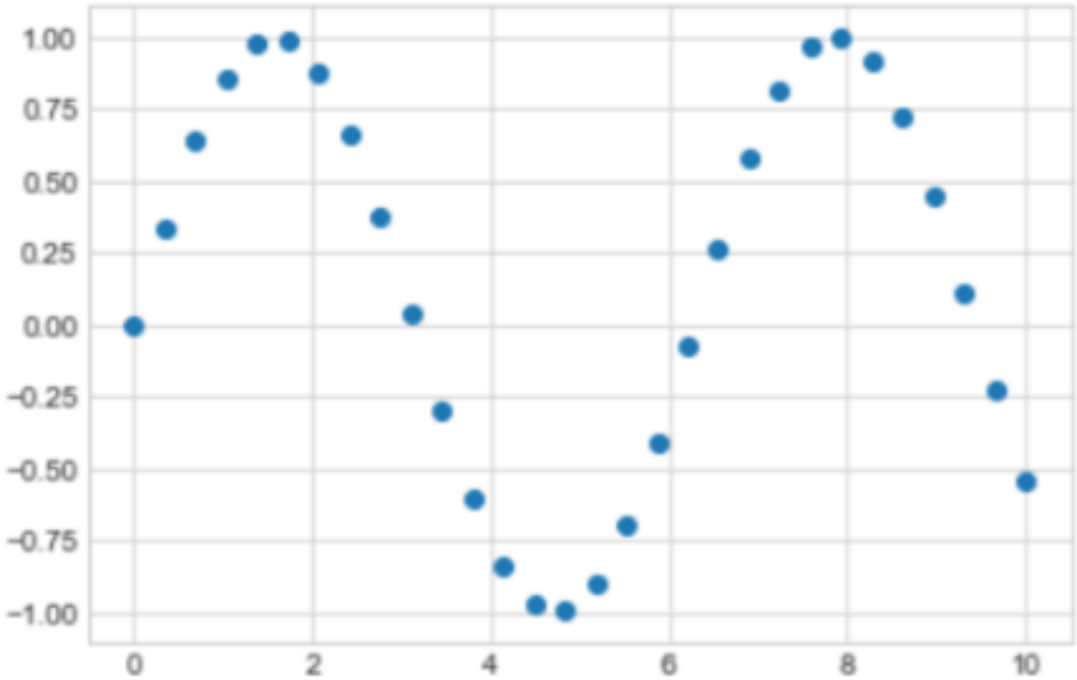
```
plt.plot(x, np.sin(x), 'o', color='black');
```



Đồ thị điểm rời rạc

Phương thức thứ hai, mạnh mẽ hơn để tạo đồ thị phân tán là hàm *plt.scatter*, được sử dụng rất giống với hàm *plt.plot*:

```
plt.scatter(x, y, marker='o');
```



Trực quan hóa lỗi

Đối với bất kỳ phép đo khoa học nào, việc tính toán chính xác các lỗi cũng quan trọng không kém việc báo cáo chính xác con số. Khi trực quan hóa dữ liệu và các kết quả, việc hiển thị các lỗi một cách hiệu quả có thể giúp đồ thị truyền tải thông tin đầy đủ hơn nhiều.

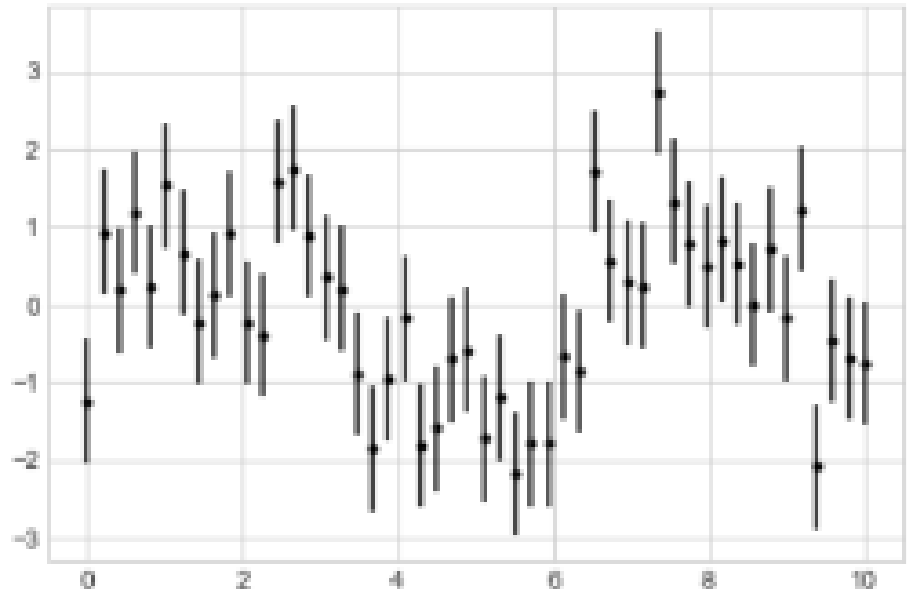
Trực quan bằng thanh lỗi

- Thanh lỗi (Error bar) được sử dụng làm đại diện cho sự biến đổi của một điểm dữ liệu. Điều này cung cấp về mức độ chính xác của điểm dữ liệu. Mức độ biến thiên càng nhiều thì điểm dữ liệu trong biểu đồ càng kèm chính xác.

Trực quan bằng thanh lỗi

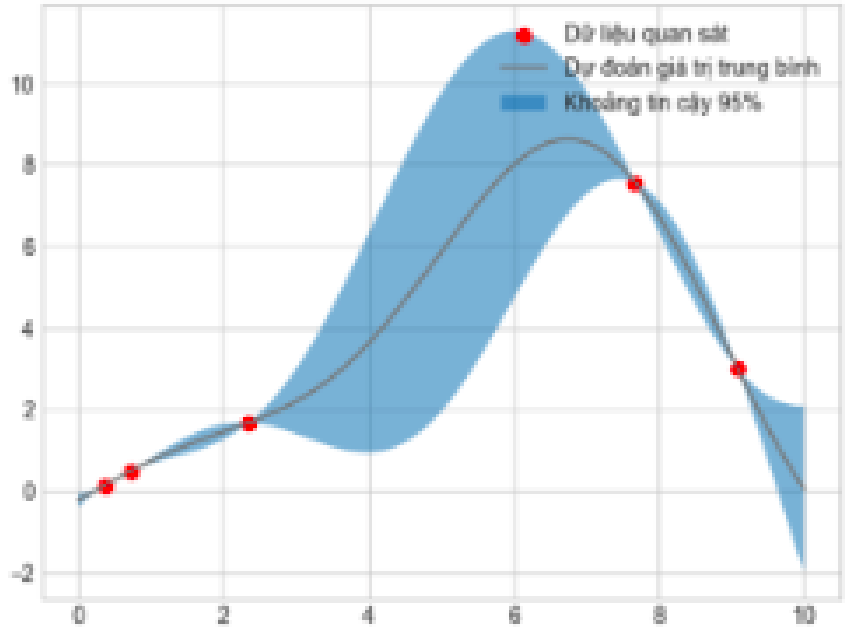
- Trong Python, trực quan lỗi bằng thanh được tạo bằng một lệnh gọi hàm *Matplotlib* duy nhất.

```
x = np.linspace(0, 10, 50)
dy = 0.8
y = np.sin(x) + dy * np.random.randn(50)
plt.errorbar(x, y, yerr=dy, fmt='.k');
```



Trực quan bằng miền lỗi liên tục

Trong một số trường hợp, mong muốn trực quan đồ thị lỗi trên miền liên tục, có thể kết hợp hàm *plt.plot* và *plt.fill_between*.



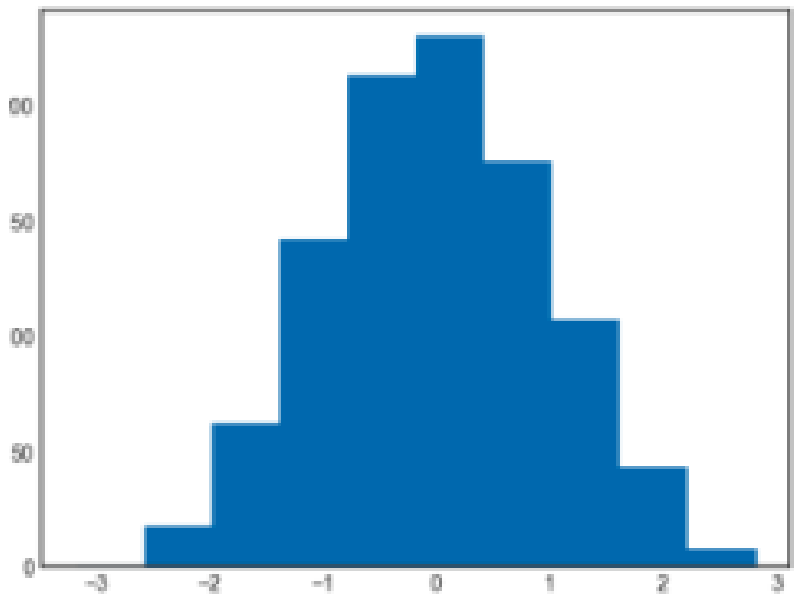
Đồ thị đường viền

- Đôi khi sẽ hữu ích khi hiển thị dữ liệu ba chiều ở không gian hai chiều bằng cách sử dụng các đường viền hoặc vùng mã màu chuyển tiếp khác nhau.
- Có ba hàm Matplotlib có thể hữu ích cho nhiệm vụ này: *plt.contour* cho các ô đường viền, *plt.contourf* cho các ô đường viền được tô màu và *plt.imshow* để hiển thị hình ảnh.

Histograms và mật độ

- Histograms là một biểu đồ đơn giản để hiểu tập dữ liệu, đồ thị cơ bản được trực quan hoá bằng một dòng mã lệnh, sau khi nhập một tập dữ liệu như sau:

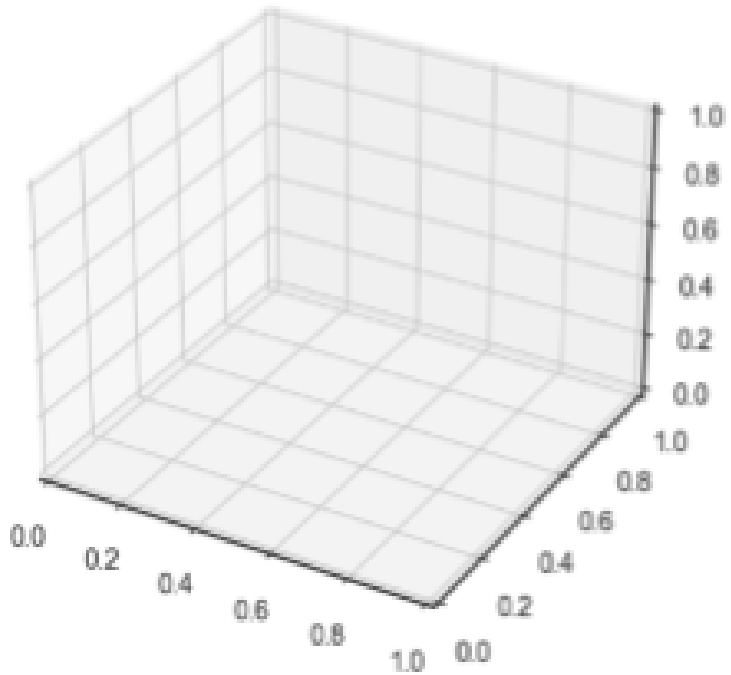
```
plt.hist(data, bins=30, alpha=0.5, histtype='stepfilled',
         color='steelblue', edgecolor='none');
```



Đồ thị ba chiều

Matplotlib có một số gói vẽ sơ đồ ba chiều đã được xây dựng trên màn hình hai chiều . Để sử dụng biểu đồ ba chiều bằng cách nhập bộ công cụ *mplot3d*.

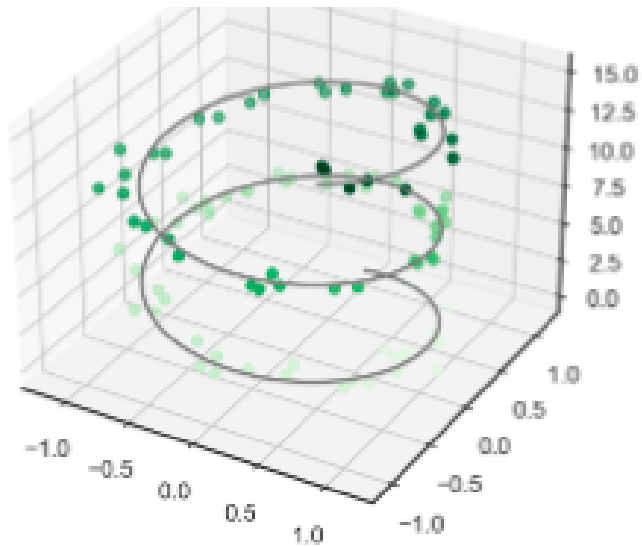
```
from mpl_toolkits import mplot3d
```



Điểm và đường ba chiều

Đồ thị ba chiều đơn giản nhất là một biểu đồ đường hoặc phân tán các điểm dữ liệu được tạo từ các bộ ba *(x, y, z)* bằng cách sử dụng các hàm *ax.plot3D* và *ax.scatter3D*.

```
%matplotlib notebook
ax = plt.axes(projection='3d')
# Du lieu cho duong 3 chieu
zline = np.linspace(0, 15, 1000)
xline = np.sin(zline)
yline = np.cos(zline)
ax.plot3D(xline, yline, zline, 'gray')
# Du lieu cho cac diem roi rac 3 chieu
zdata = 15 * np.random.random(100)
xdata = np.sin(zdata) + 0.1 * np.random.randn(100)
ydata = np.cos(zdata) + 0.1 * np.random.randn(100)
ax.scatter3D(xdata, ydata, zdata, c=zdata, cmap='Greens');
```



Dữ liệu địa lý

- Một loại trực quan hóa phổ biến trong khoa học dữ liệu là dữ liệu địa lý.
- Công cụ chính của *Matplotlib* cho kiểu trực quan hóa này là bộ công cụ Bản đồ cơ sở - *Basemap*, đây là một trong một số bộ công cụ của *Matplotlib* nằm trong miền *mpl_toolkits*.
- Bên cạnh đó hiện nay API Google Maps được sử dụng phổ biết và là lựa chọn tốt hơn để trực quan hóa bản đồ chuyên sâu.

Bài tập Chương 3

- Tạo hoặc tìm dữ liệu phù hợp để trực quan hoá dữ liệu dbằng các loại đồ thi sau:
  - Đồ thị dạng đường thẳng
  - Đồ thị điểm rời rạc
  - Trực quan hóa lỗi
  - Đồ thị đường viền
  - Histograms và mật độ
  - Đồ thị ba chiều
  - Dữ liệu địa lý

Mô tả và nhận xét về đồ thị vẽ được.