

# Python 数据可视化

Introduction to Data Visualization with Python

刘子涵

大三 电院 信息安全

zihanliu2000@gmail.com

2021/1/18

# Contents

I 数学建模中的可视化

II matplotlib 库常用函数介绍

III pyecharts 库简介

IV seaborn 库简介

# I 数学建模中的可视化

在数学建模竞赛中，可视化主要有三大功能：

- ① 直观表达模型的思路 —— 思维导图、流程图、框架图
- ② 直观展现模型的亮点 —— 几何物理模型图、模型示意图等
- ③ 直观呈现建模的结果 —— 各种数据图表



Python 数据可视化

# 数学建模中可视化的功能及分类

## ① 直观表达模型的思路 —— 思维导图、流程图、框架图

绘图工具：PPT、Visio、Edraw、ProcessOn

- PPT：简单易用，方便排版
- Visio/Edraw：模板多，功能强（此处推荐Edraw，容易上手，可视化美观，但是要收费(╥\_╥)）
- ProcessOn：在线流程图网站 (<https://www.processon.com/>)

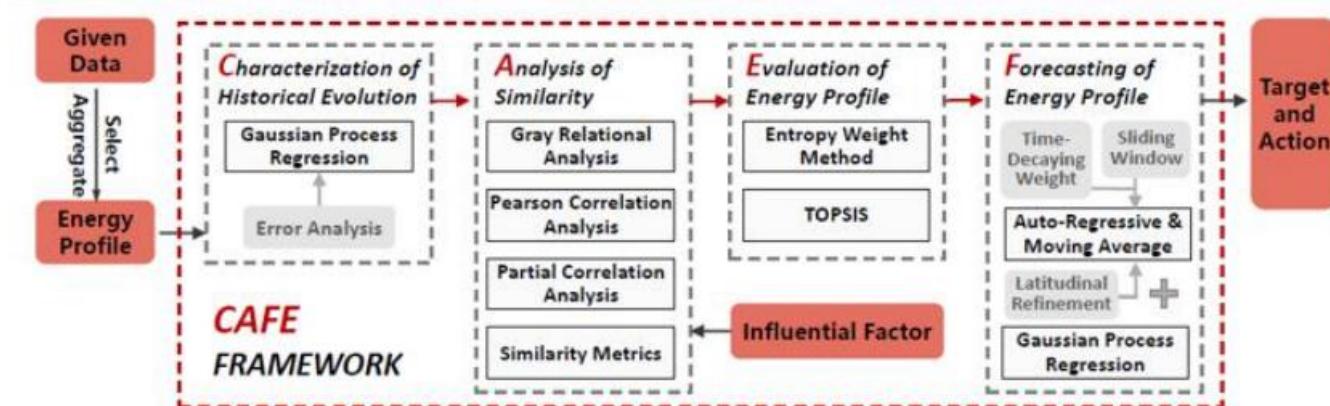
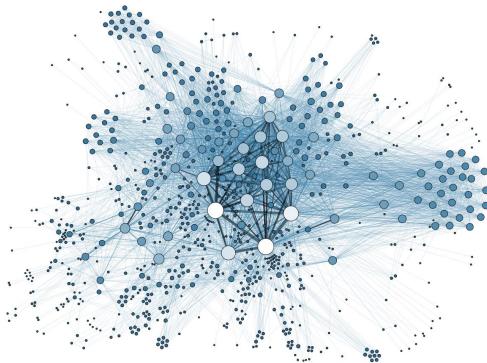


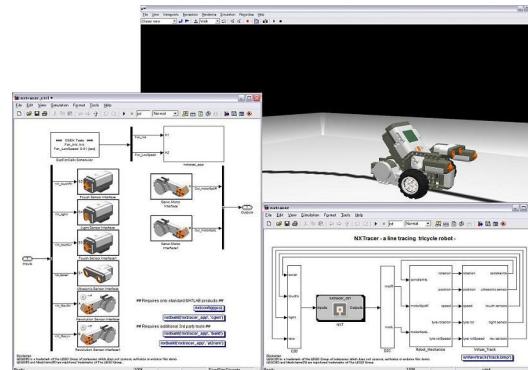
Figure 1: Framework of CAFE (Characterization, Analysis, Forecasting, and Evaluation of Energy Profile)

## ② 直观展现模型的亮点 —— 几何物理模型图、模型示意图等

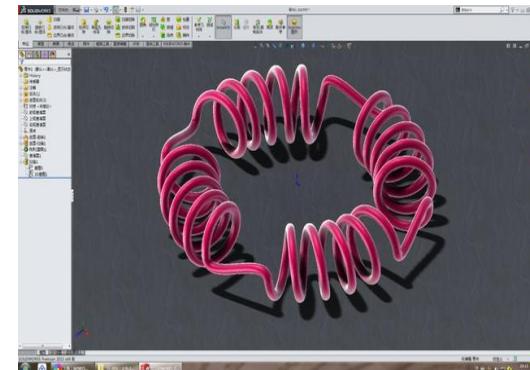
- 侧重特定专业模型图（比如动力学建模、机械系统建模、复杂网络建模、地理建模、生物学建模等）
- 比如 2019 国赛 B 题“同心鼓”策略，2020 美赛 D 题足球运动中的传球网络建模
- 特定专业领域的作图往往难以快速上手，需要平时积累！



Gephi：  
复杂网络数据可视化



MATLAB-Simulink：  
动态系统建模、仿真



SolidWorks：  
三维建模、机械设计

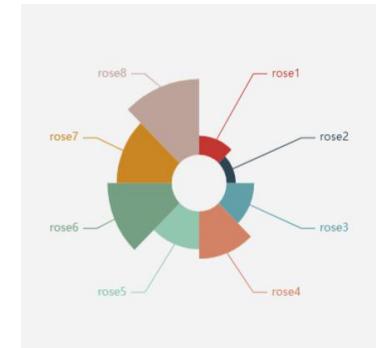
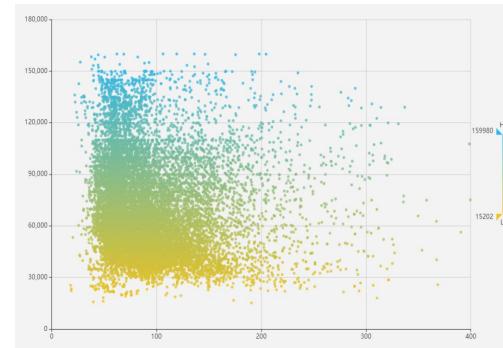
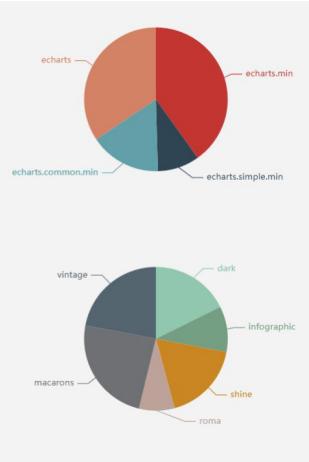
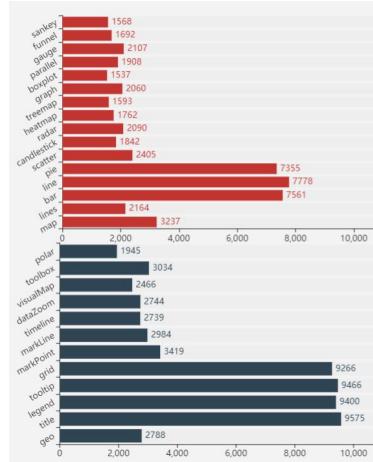
# 数学建模中可视化的功能及分类

## ③ 直观呈现建模的结果 —— 各种数据图表

- 借助编程工具
  - Python (Matplotlib, Seaborn 等)
  - MATLAB
  - R
  - Origin, Excel
- .....

- 借助在线工具
  - Highcharts (<https://www.highcharts.com/>)
  - Echarts (<https://www.echartsjs.com/en/index.html>)

- 数据可视化是数据分析的关键:
  - 不仅仅是用于呈现建模结果，还可以反过来帮助建模



Echarts 示例：高端、大气、上档次

# 有了 Highcharts，Echarts，为什么还要学 Python 可视化？

## 1、Python 功能齐全而强大，集成多种数据分析和数据挖掘工具

- 数学建模不仅仅是可视化，而是多种技术的交叉融合，熟练掌握 Python 能为数学建模竞赛、科研、工作等打下良好基础。

- 除了 matplotlib, seaborn, pyecharts, plotly 等可视化库，还有
  - 比如 numpy, pandas 提供类似 MATLAB 的数据分析包，
  - 比如 sklearn, keras, tensorflow, pytorch 等机器学习、深度学习库，
  - 比如 urllib, BeautifulSoup 提供网络爬虫技术，一种数据收集的重要途径。

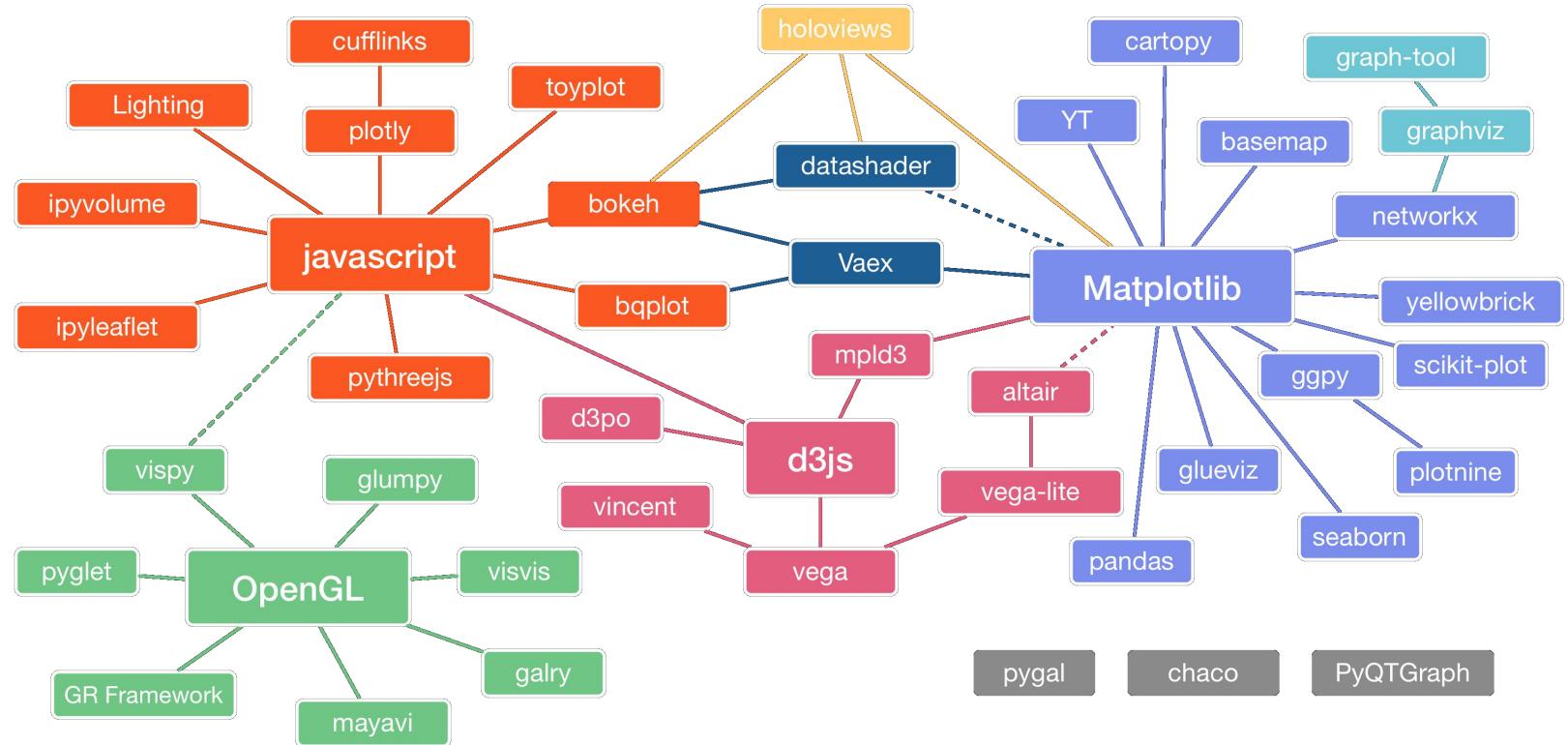
## 2、Python 可视化工具更为灵活，更易排版生成的图片

- 比如一个简单的 seaborn 绘制散点图的函数，参数达20个以上，便于灵活作图。

```
seaborn.scatterplot(x=None, y=None, hue=None, style=None, size=None, data=None,
palette=None, hue_order=None, hue_norm=None, sizes=None, size_order=None, size_norm=None,
markers=True, style_order=None, x_bins=None, y_bins=None, units=None, estimator=None,
ci=95, n_boot=1000, alpha='auto', x_jitter=None, y_jitter=None, legend='brief', ax=None,
**kwargs)
```

- 自由设置、排版子图

# Python 可视化工具总览



在数学建模竞赛中，Matplotlib 是最为核心、最为基础的 Python 可视化工具。

## III matplotlib 库常用函数介绍

# plot() 函数——展现变量的变化趋势

```
# matplotlib/plot.py

import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(0.5, 3.5, 300) # 在 0.5 到 3.5 之间均匀地取 300 个数
y = np.sin(x)

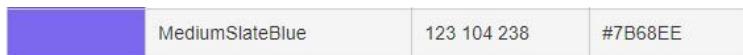
# 绘制曲线
plt.plot(x, y, color='blue', linestyle='-', linewidth=1, marker='o',
          markersize=4, label='Line')
```

## ① color:

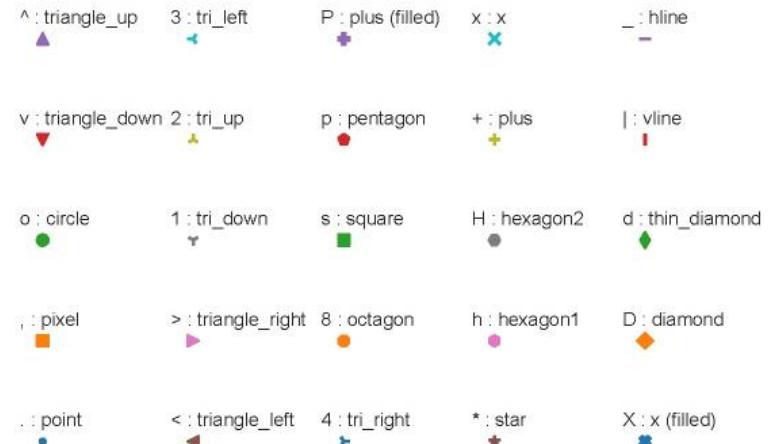
简单颜色可简写，比如

b: blue	g: green
r: red	c: cyan
m: magenta	y: yellow
k: black	w: white

其它颜色，可使用16进制表示法：



## ④ marker:



## ② linestyle:

'-' 直线    '--' 虚线  
'-.' 点划线    ':' 点线

## ③ linewidth: 调整线的粗细程度

## ⑤ markersize: 调整点的粗细程度

## ⑥ label: 图例上显示的标签

# Matplotlib 图表基本组成 (1) —— 关于 plt

## ① 坐标轴标签属性

```
plt.xlabel()  
plt.ylabel()
```

## ② 坐标轴上坐标数标签属性

```
plt.xticks()  
plt.yticks()
```

## ③ 标题显示

```
plt.title()
```

## ④ 坐标显示范围属性

```
plt.xlim()  
plt.ylim()
```

## ⑤ 图例显示 (显示label中所写内容)

```
plt.legend()
```

## ⑥ 网格线显示

```
plt.grid()
```

```
# matplotlib/plot.py  
  
# 坐标轴范围  
plt.xlim(1, 3) # x 范围  
plt.ylim(min(y), max(y)) # y 范围  
  
# 坐标轴坐标数标签  
# 第一个参数为欲替换的 x 坐标, 第二个参数为对应替换坐标的标签  
plt.xticks([0, 2, 4, 6, 8], [1, 2, 'a', 4, 5])  
# 也可以直接给出坐标数组  
plt.yticks(np.linspace(-1, 1, 20))  
  
# 坐标轴标签  
plt.xlabel('time', fontname='Times New Roman', fontsize=10)  
plt.ylabel('value')  
  
# 图片标题  
plt.title('plot')  
  
# 图例显示  
# loc--位置 (1:右上, 2: 左上, 3: 左下, 4: 右下)  
plt.legend(loc=1)  
  
# 网格线开关  
plt.grid(axis="x")  
plt.grid(axis="y")  
  
# 保存图片  
plt.savefig('plot.png', dpi=800)  
  
# 显示图片  
plt.show()
```

# Matplotlib 图表基本组成 (2) —— 基本元素

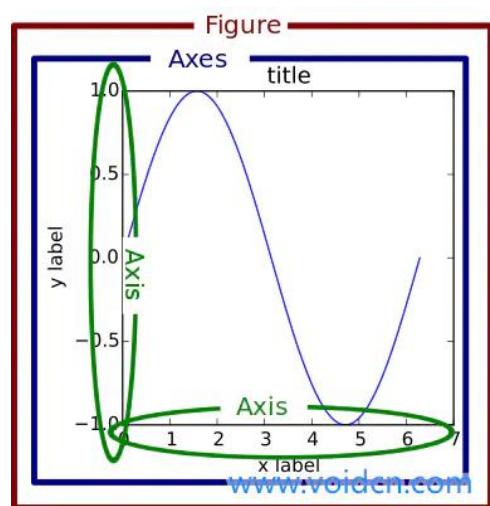
## ① Figure

图像窗口，可以理解为一张画布

## ② Axes

子图，带有数据的图像区域

子图 ax 类似 plt，但函数不同，  
调整图表参数更为灵活



图表结构

```
# matplotlib/figure.py

x = np.linspace(0.5, 3.5, 300) # 在 0.5 到 3.5 之间均匀地取 300 个数
y = np.sin(x)
z = np.cos(x)

# fig, ax = plt.subplots(figsize=(14,7)) 不设置子图
fig, ax = plt.subplots(1,2,figsize=(14,7))
ax[0].plot(x,y)
ax[1].plot(x,z)

ax[0].set_title('sin(x)', fontsize=18)
ax[1].set_title('cos(x)', fontsize=18)

ax[0].set_xlabel('x', fontsize=18, fontfamily = 'sans-serif',
fontstyle='italic')
ax[1].set_xlabel('x', fontsize=18, fontfamily = 'sans-serif',
fontstyle='italic')

ax[0].set_ylabel('y', fontsize='x-large',fontstyle='oblique')
ax[1].set_ylabel('y', fontsize='x-large',fontstyle='oblique')

ax[0].set_xlim(0,16)
ax[0].grid(axis='both')

ax[1].xaxis.set_tick_params(rotation=45, labelsize=12)
start, end = ax[1].get_xlim()
ax[1].xaxis.set_ticks(np.arange(start, end, 1)) # np.arange(start,
end, step)
ax[1].yaxis.tick_right()

plt.savefig('figure.png', dpi=800)
plt.show()
```

# scatter( ) 函数——寻找变量之间的关系

scatter(x, y, s=10, c='b', marker='+', cmap=None, label='scatter')

x: x轴数值

y: y轴数值

s: 点的大小

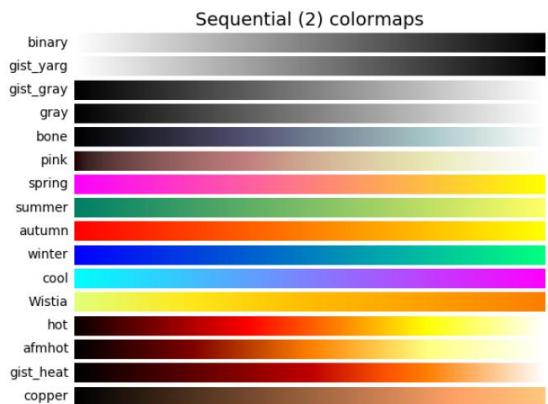
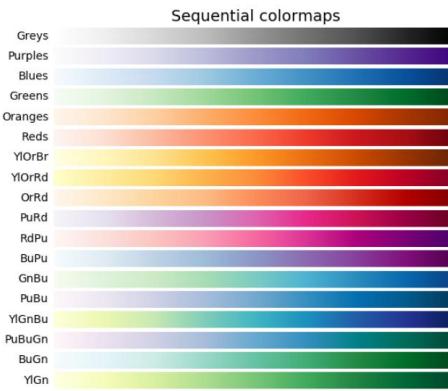
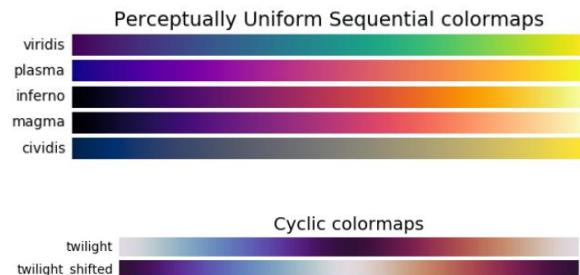
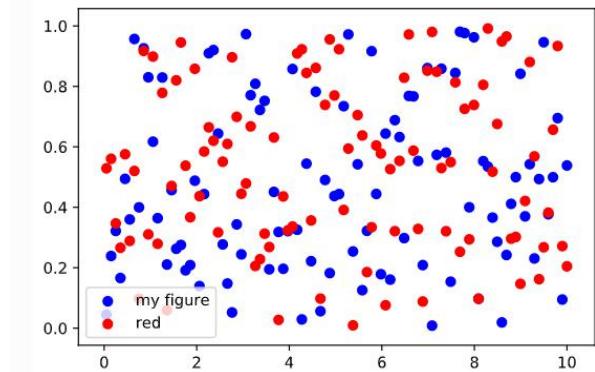
c: 点的颜色

marker: 点的样式

cmap: 颜色图

(具体可参考 <https://matplotlib.org/3.1.0/tutorials/colors/colormaps.html>)

label: 标签



matplotlib 中预定义了大量颜色图，可以直接使用

## scatter( ) 函数实例演示 (1)

给出以下数据，如何研究 heightIn 与 ageYear 和 sex 之间的关系？  
怎么可视化这三个变量？

sex	ageYear	ageMonth	heightIn	weightLb	age
f	11.92	143	56.3	85	11
f	12.92	155	62.3	105	12
f	12.75	153	63.3	108	12
f	13.42	161	59	92	13
f	15.92	191	62.5	112.5	15
f	14.25	171	62.5	112	14
f	15.42	185	59	104	15
f	11.83	142	56.5	69	11
f	13.33	160	62	94.5	13

## scatter( ) 函数实例演示 (2)

(ageYear, heightIn) 为二维平面上的离散点，可以用二维散点图呈现；

sex 在有限集合中取值，可以采用非坐标轴的方式（如点的颜色、点的大小等）来描述。

```
# matplotlib/scatter.py

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# 导入数据集
df = pd.read_csv("scatter_data.csv")
# 拆分不同性别对应数据
mdf = df.query("sex=='m'")
fdf = df.query("sex=='f'")
# 绘制男性数据
ax = mdf.plot.scatter(x='ageYear', y='heightIn', c='b', s=10,
label='male')
# 绘制女性数据
fdf.plot.scatter(x='ageYear', y='heightIn', c='r', s=30,
label='female', ax=ax)
# 取题目
ax.set_title('scatter')
# 保存图片
plt.savefig('scatter.png', dpi=800)
# 显示绘图结果
plt.show()
```

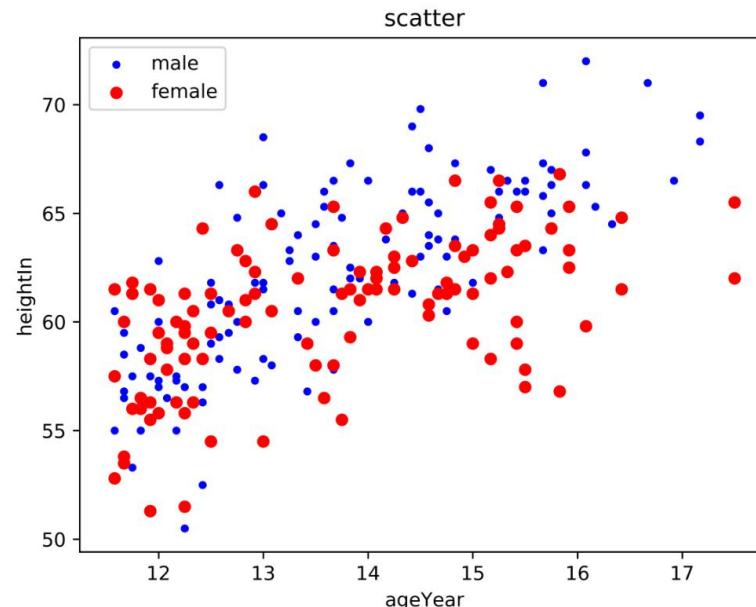
## scatter( ) 函数实例演示 (3)

从图中我们可以观察得到：

- heightIn 与 ageYear 呈正相关关系；
- 在 ageYear 较小的时候，两性之间 heightIn 没有较大区别，而在 ageYear 较大的时候，女性比男性偏低。

得到散点图后，如何进行下一步数据分析？

- 根据模型假设和实际需求的不同，可以回归分析、聚类分析、降维处理.....



# bar( ) 函数 & barh( ) 函数 —— 垂直 & 水平柱状图

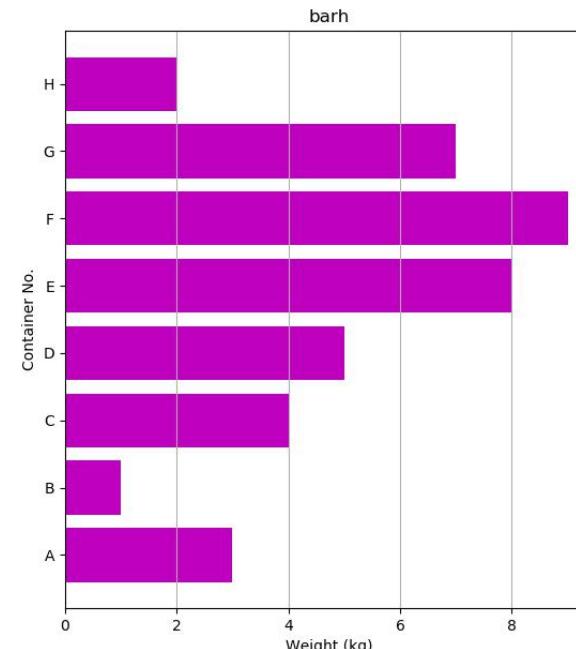
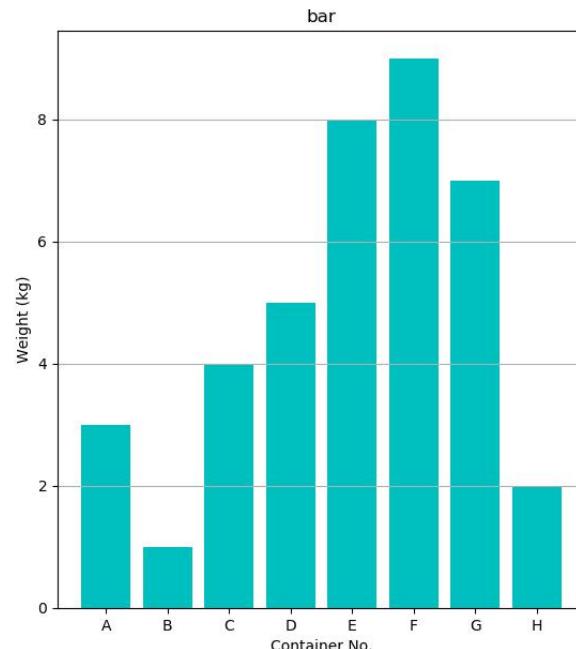
bar(x, y, align='center', color='c', tick\_label = ['A','B','C','D','E','F','G','H'])

绘制垂直柱状图

x: 标示在 x 轴上的定性数据的分布特征

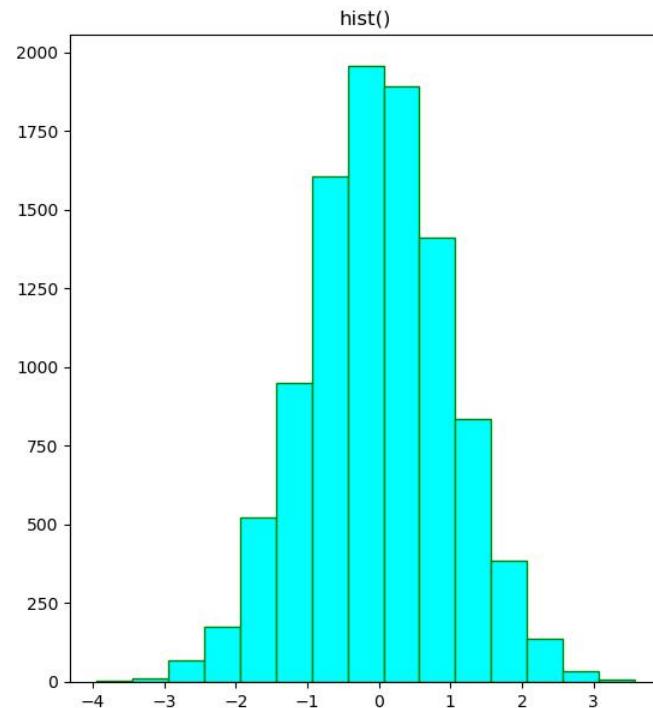
y: 每种定性数据类别的数量

barh( )类似 (水平柱状图)



# hist() 函数——直方图

hist(x): 绘制直方图，在x轴上绘制定量数据的分布特征。



```
# matplotlib/hist.py  
  
ax[0].hist(X, bins=15,  
            edgecolor='green',  
            facecolor='cyan',  
            linewidth=1,  
            histtype='bar')
```

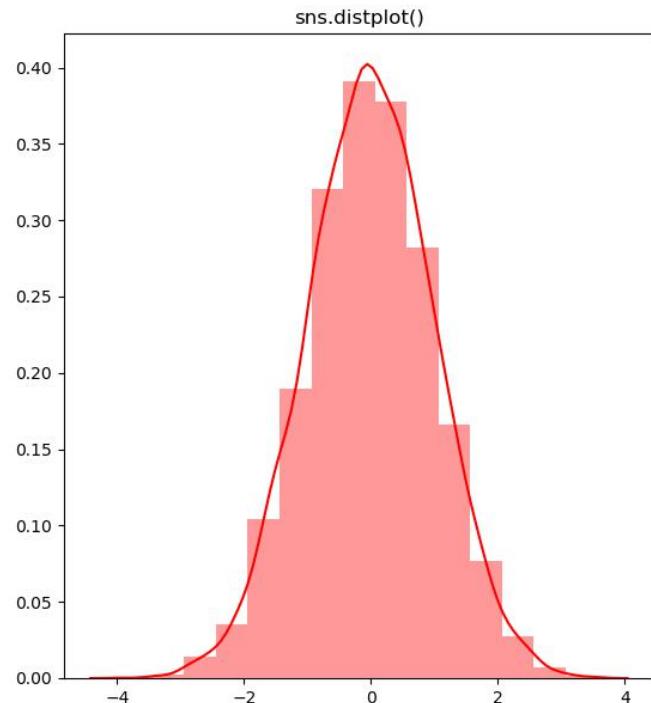
重要参数：

X：一维数组

bins：将数据划分的组数

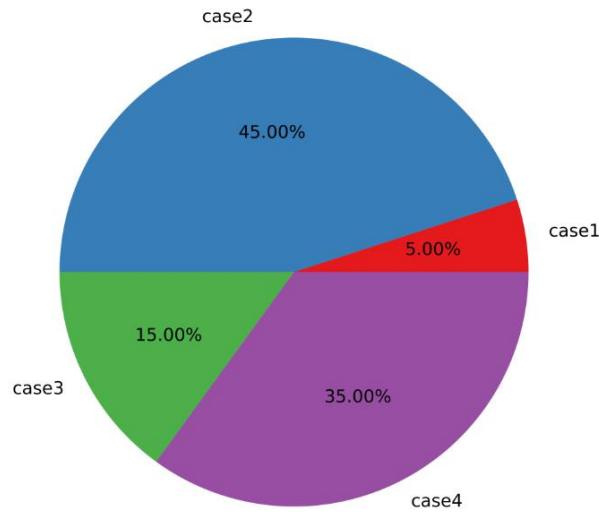
如何画出概率密度曲线来拟合直方图？

# seaborn.distplot() 函数——直方图 + 拟合曲线



```
# matplotlib/hist.py  
  
import seaborn as sns  
  
sns.set_palette("hls") # 设置所有图的颜色，使用 hls 色彩空间  
sns.distplot(X, color="r", bins=15, kde=True, ax=ax[1])  
ax[1].set_title('sns.distplot()')
```

# pie() 函数——饼状图



```
# matplotlib/pie.py

colors = ["#e41alc", "#377eb8", "#4daf4a", "#984ea3"]
soldNums = [0.05, 0.45, 0.15, 0.35]
kinds = ['case1', 'case2', 'case3', 'case4']

plt.pie(soldNums,
        labels=kinds,
        autopct="% .2f%%", # 保留两位小数
        startangle=0, # 起始角度
        colors=colors);
```

# stackplot() 函数——堆叠折线图

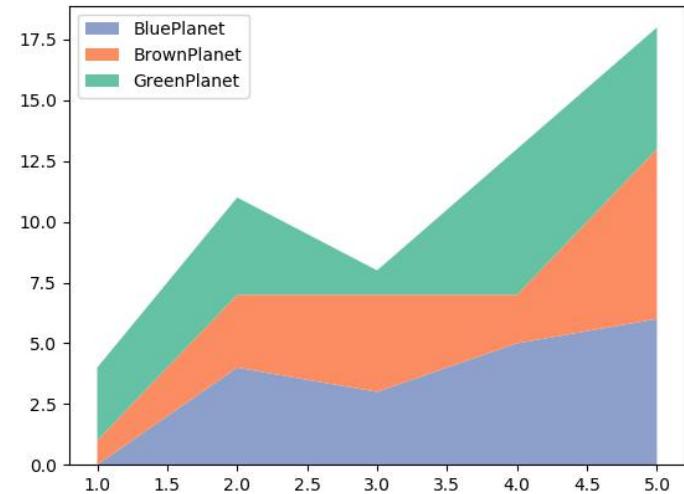
**堆叠图**（堆叠折线图、堆叠条形图）的优势：

- 横向比较：

将一个大类目拆分成多个子类目，能够显示各子类目的占比情况；

- 纵向比较：

- 大类目整体随某一变量变化情况；
- 各子类目随某一变量变化情况。



```
# matplotlib/stackplot.py

x = np.arange(1,6) # [1 2 3 4 5]
y1 = [0,4,3,5,6]
y2 = [1,3,4,2,7]
y3 = [3,4,1,6,5]

labels = ["BluePlanet","BrownPlanet","GreenPlanet"]
colors = ["#8da0cb","#fc8d62","#66c2a5"]

plt.stackplot(x, y1, y2, y3, labels=labels, colors=colors)
```

# Summary

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

折线图: plt.plot()

散点图: plt.scatter()

柱状图: plt.bar() / plt.barh()

直方图: plt.hist() / seaborn.distplot()

饼状图: plt.pie()

堆叠折线图: plt.stackplot()

.....

# Matplotlib Cheat Sheet

## Python For Data Science Cheat Sheet

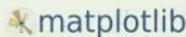
### Matplotlib

Learn Python [Interactively](#) at [www.DataCamp.com](http://www.DataCamp.com)



### Matplotlib

Matplotlib is a Python 2D plotting library which produces publication-quality figures in a variety of hardcopy formats and interactive environments across platforms.



### 1 Prepare The Data

Also see [Lists & NumPy](#)

#### 1D Data

```
>>> import numpy as np  
>>> x = np.linspace(0, 10, 100)  
>>> y = np.cos(x)  
>>> z = np.sin(x)
```

#### 2D Data or Images

```
>>> data = 2 * np.random.random((10, 10))  
>>> data2 = 3 * np.random.random((10, 10))  
>>> Y, X = np.mgrid[-3:3:100j, -3:3:100j]  
>>> U = -1 - X**2 + Y  
>>> V = 1 + X - Y**2  
>>> from matplotlib.cbook import get_sample_data  
>>> img = np.load(get_sample_data('axes_grid/bivariate_normal.npy'))
```

### 2 Create Plot

```
>>> import matplotlib.pyplot as plt
```

#### Figure

```
>>> fig = plt.figure()  
>>> fig2 = plt.figure(figsize=plt.figaspect(2.0))
```

#### Axes

All plotting is done with respect to an Axes. In most cases, a subplot will fit your needs. A subplot is an axes on a grid system.

```
>>> fig.add_axes()  
>>> ax1 = fig.add_subplot(221) # row-col-num  
>>> ax3 = fig.add_subplot(212)  
>>> fig3, axes3 = plt.subplots(nrows=2, ncols=2)  
>>> fig4, axes2 = plt.subplots(ncols=3)
```

### 3 Plotting Routines

#### 1D Data

```
>>> fig, ax = plt.subplots()  
>>> lines = ax.plot(x,y)  
>>> ax.scatter(x,y)  
>>> axes[0,0].bar([1,2,3],[3,4,5])  
>>> axes[0,0].barh([0.5,1.2,2.5],[0,1,2])  
>>> axes[1,1].axhline(0.45)  
>>> axes[0,1].axvline(0.65)  
>>> ax.fill(x,y,color='blue')  
>>> ax.fill_between(x,y,color='yellow')
```

Draw points with lines or markers connecting them  
Draw unconnected points, scaled or colored  
Plot vertical rectangles (constant width)  
Plot horizontal rectangles (constant height)  
Draw a horizontal line across axes  
Draw a vertical line across axes  
Draw filled polygons  
Fill between y-values and 0

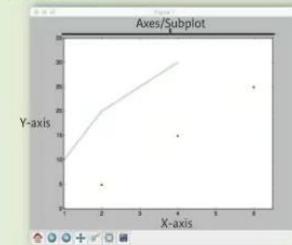
#### 2D Data or Images

```
>>> fig, ax = plt.subplots()  
>>> im = ax.imshow(img,  
                  cmap='gist_earth',  
                  interpolation='nearest',  
                  vmin=-2,  
                  vmax=2)
```

Colormapped or RGB arrays

## Plot Anatomy & Workflow

### Plot Anatomy



### Workflow

The basic steps to creating plots with matplotlib are:

- 1 Prepare data
- 2 Create plot
- 3 Plot
- 4 Customize plot
- 5 Save plot
- 6 Show plot

```
>>> import matplotlib.pyplot as plt  
>>> x = [1,2,3,4] Step 1  
>>> y = [10,20,25,30]  
>>> fig = plt.figure() Step 2  
>>> ax = fig.add_subplot(111) Step 3  
>>> ax.plot(x, y, color='lightblue', linewidth=3) Step 4, 4  
>>> ax.scatter([2,4,6],  
             [5,15,25],  
             color='darkgreen',  
             marker='^')  
>>> ax.set_xlim(1, 6.5)  
>>> plt.savefig('foo.png')  
>>> plt.show() Step 6
```

### 4 Customize Plot

#### Colors, Color Bars & Color Maps

```
>>> plt.plot(x, x, x, x**2, x, x**3)  
>>> ax.plot(x, y, alpha = 0.4)  
>>> ax.plot(x, y, c='k')  
>>> fig.colorbar(im, orientation='horizontal')  
>>> im = ax.imshow(img,  
                  cmap='seismic')
```

#### Markers

```
>>> fig, ax = plt.subplots()  
>>> ax.scatter(x,y,marker=".")  
>>> ax.plot(x,y,marker="o")
```

#### Linestyles

```
>>> plt.plot(x,y,linewidth=4.0)  
>>> plt.plot(x,y,ls='solid')  
>>> plt.plot(x,y,ls='--')  
>>> plt.plot(x,y,'--',x**2,y**2,'-.')  
>>> plt.setp(lines,color='r',linewidth=4.0)
```

#### Text & Annotations

```
>>> ax.text(1,  
           -2.1,  
           'Example Graph',  
           style='italic')  
>>> ax.annotate("Sine",  
               xy=(8, 0),  
               xycoords='data',  
               xytext=(10.5, 0),  
               textcoords='data',  
               arrowprops=dict(arrowstyle=">",  
                               connectionstyle="arc3"),),
```

#### Vector Fields

```
>>> axes[0,1].arrow(0,0,0.5,0.5)  
>>> axes[1,1].quiver(y,z)  
>>> axes[0,1].streamplot(X,Y,U,V)
```

#### Data Distributions

```
>>> ax1.hist(y)  
>>> ax3.boxplot(y)  
>>> ax3.violinplot(z)
```

#### Mathtext

```
>>> plt.title(r'$\sigma_i=15$', fontsize=20)
```

#### Limits, Legends & Layouts

**Limits & Autoscaling**  
>>> ax.margins(x=0.0,y=0.1)  
>>> ax.axis('equal')  
>>> ax.set(xlim=[0,10.5],ylim=[-1.5,1.5])  
>>> ax.set\_xlim(0,10.5)  
**Legends**  
>>> ax.set(title='An Example Axes',  
 ylabel='Y-Axis',  
 xlabel='X-Axis')  
>>> ax.legend(loc='best')

**Ticks**  
>>> ax.xaxis.set(ticks=range(1,5),  
 ticklabels=[3,100,-12,"foo"])  
>>> ax.tick\_params(axis='y',  
 direction='inout',  
 length=10)

**Subplot Spacing**  
>>> fig3.subplots\_adjust(wspace=0.5,  
 hspace=0.3,  
 left=0.125,  
 right=0.9,  
 top=0.9,  
 bottom=0.1)

>>> fig.tight\_layout()

**Axis Spines**  
>>> ax1.spines['top'].set\_visible(False)  
>>> ax1.spines['bottom'].set\_position(('outward',10))

Add padding to a plot  
Set the aspect ratio of the plot to 1  
Set limits for x and y-axis  
Set limits for x-axis

Set a title and x and y axis labels

No overlapping plot elements

Manually set x-ticks

Make y-ticks longer and go in and out

Adjust the spacing between subplots

Fit subplot(s) in to the figure area

Make the top axis line for a plot invisible  
Move the bottom axis line outward

### 5 Save Plot

#### Save figures

```
>>> plt.savefig('foo.png')  
Save transparent figures  
>>> plt.savefig('foo.png', transparent=True)
```

### 6 Show Plot

```
>>> plt.show()
```

### Close & Clear

```
>>> plt.clf()  
>>> plt.cla()  
>>> plt.close()
```

Clear an axis  
Clear the entire figure  
Close a window



### III pyecharts 库简介

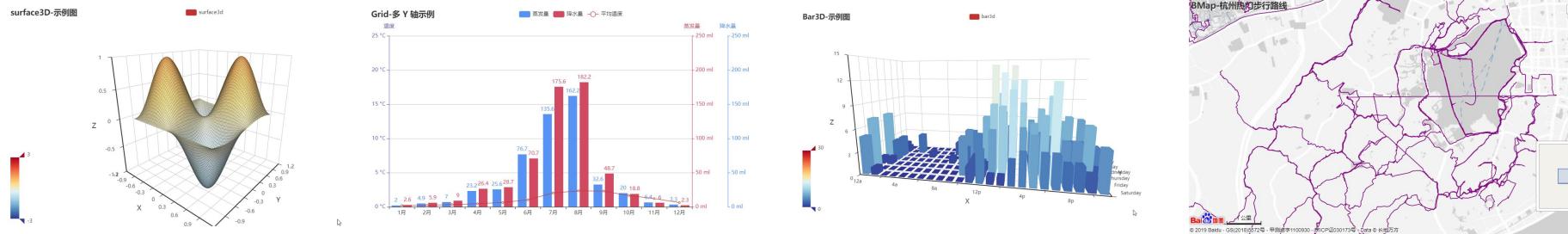
# 什么是 pyecharts?

*pyecharts* 是一个用于生成 Echarts 图表的 Python 库，便于用户在 Python 中处理数据之后直接生成类似 Echarts 的可视化效果。

- Echarts 是由百度开源的数据可视化 JS 库，图表交互性强，设计精巧，*pyecharts* 继承了 Echart 的图表风格。
- 借助百度地图 API 和大量内置地图文件，为地理数据可视化提供支持。



Echarts 能画的，  
*pyecharts* 基本都能画！



# pyecharts 能画什么？

## 基本图表

- Calendar: 日历图
- Funnel: 漏斗图
- Gauge: 仪表盘
- Graph: 关系图
- Liquid: 水球图
- Parallel: 平行坐标系
- Pie: 饼图
- Polar: 极坐标系
- Radar: 雷达图
- Sankey: 桑基图
- Sunburst: 旭日图
- ThemeRiver: 主题河流图
- WordCloud: 词云图

## 直角坐标系图表

- Bar: 柱状图/条形图
- Boxplot: 箱形图
- EffectScatter: 涟漪特效散点图
- HeatMap: 热力图
- Kline/Candlestick: K线图
- Line: 折线/面积图
- PictorialBar: 象形柱状图
- Scatter: 散点图
- Overlap: 层叠多图

## 地理图表

- Geo: 地理坐标系
- Map: 地图
- BMap: 百度地图

## 3D 图表

- Bar3D: 3D柱状图
- Line3D: 3D折线图
- Scatter3D: 3D散点图
- Surface3D: 3D曲面图
- Map3D - 三维地图

## 树型图表

- Tree: 树图
- TreeMap: 矩形树图

具体画法可参考：

① pyecharts 官方文档: <https://pyecharts.org/>

② pyecharts gallery —— 大量可参考的实例和代码: <https://gallery.pyecharts.org/>

# 一个简单例子——柱状图

```
# pyecharts/bar.py

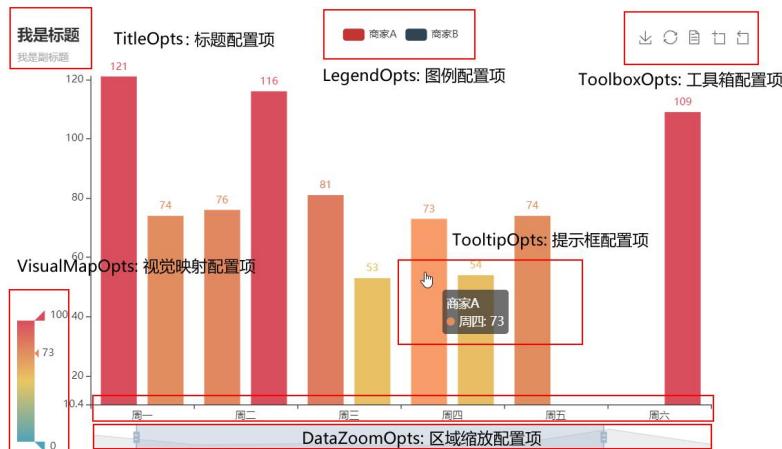
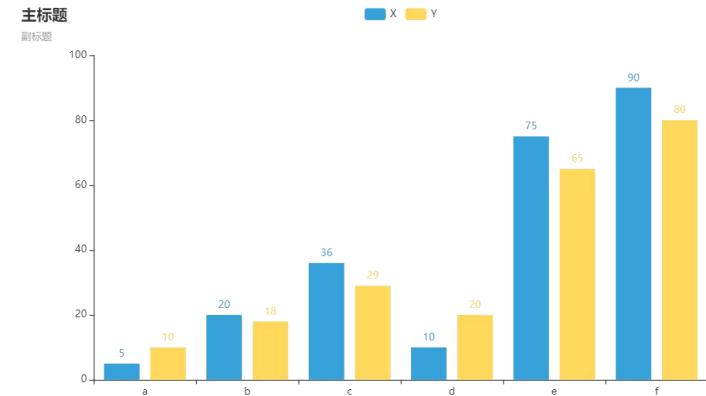
from pyecharts.charts import Bar
from pyecharts import options as opts

# 图片主题
from pyecharts.globals import ThemeType

#bar = Bar()
bar = Bar(init_opts=opts.InitOpts(theme=ThemeType.LIGHT)) # 在初始化选项中设置图片主题

bar.add_xaxis(["a", "b", "c", "d", "e", "f"])
bar.add_yaxis("X", [5, 20, 36, 10, 75, 90])
bar.add_yaxis("Y", [10, 18, 29, 20, 65, 80])
bar.set_global_opts(title_opts=opts.TitleOpts(title="主标题",
subtitle="副标题"))

bar.render("bar.html")
```



在 pyecharts 中，图表的一切皆由 Options 来修饰调整。

- global\_options: 初始化、工具箱、标题、图例、视觉映射、提示框、坐标轴等（如左图）
- series\_options: 图元、文字、标签、线、标记点与标记线、区域填充等

# 利用 pyecharts 实现地理空间可视化（1）

涉及地图的三大模块：

- ① Geo —— 地理坐标系
- ② Map —— 地图
- ③ BMap —— 百度地图

## Geo 模块举例 1：

```
# pyecharts/geol.py

from pyecharts.charts import Geo
from pyecharts import options as opts

data = [[['广东', 24],
         ['北京', 56],
         ['上海', 56],
         ['江西', 119],
         ['湖南', 79],
         ['浙江', 23],
         ['江苏', 97]]]

# 采用链式调用
chart = (
    Geo()
    # 控制地图类型、视角中心点等
    .add_schema(maptype="china")
    # 添加图表名称、传入数据集、选择 geo 图类型、调整图例等
    .add("geo", data)
    # 系列配置项 series_options
    .set_series_opts(label_opts=opts.LabelOpts(is_show=False))
    # 全局配置项 global_options
    .set_global_opts(
        visualmap_opts=opts.VisualMapOpts(),
        title_opts=opts.TitleOpts(title="Geo_China"),
    )
)

chart.render('geol.html')
```



参考《使用pyecharts绘制交互式动态地图》：  
<https://zhuanlan.zhihu.com/p/83231415>

# 利用 pyecharts 实现地理空间可视化 (2)

## Geo 模块举例 2：

```
# pyecharts/geo2.py

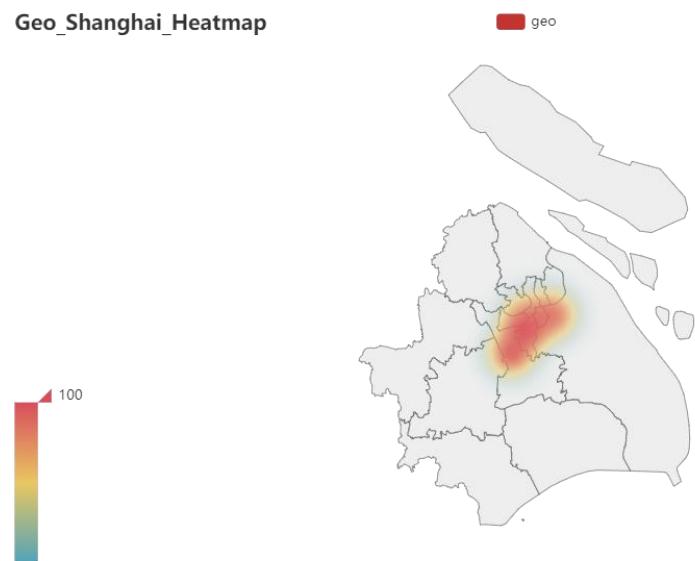
from pyecharts.charts import Geo
from pyecharts import options as opts
from pyecharts.globals import ChartType, SymbolType

data = [['黄浦区', 24],
        ['徐汇区', 105],
        ['长宁区', 56],
        ['闵行区', 258],
        ['虹口区', 8],
        ['浦东新区', 88],
        ['普陀区', 77]]

chart = (
    Geo()
    .add_schema(maptype="上海")
    .add("geo", data, type_=ChartType.HEATMAP) # 热力图形式
    .set_series_opts(label_opts=opts.LabelOpts(is_show=False))
    .set_global_opts(
        visualmap_opts=opts.VisualMapOpts(),
        title_opts=opts.TitleOpts(title="Geo_Shanghai_Heatmap"),
    )
)

chart.render('geo2.html')
```

Geo\_Shanghai\_Heatmap



# 利用 pyecharts 实现地理空间可视化（3）

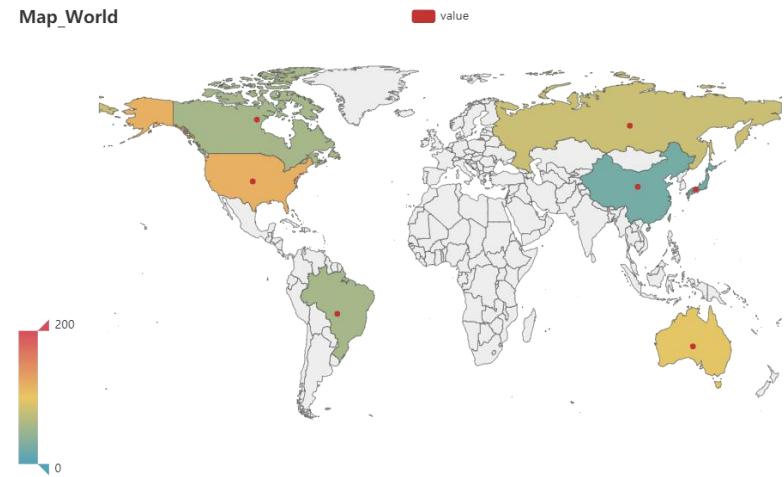
## Map 模块举例：

```
# pyecharts/map.py

from pyecharts.charts import Map
from pyecharts import options as opts

data = [['China', 24],
        ['Canada', 56],
        ['Brazil', 56],
        ['United States', 119],
        ['Russia', 79],
        ['Japan', 23],
        ['Australia', 97]]

chart = (
    Map()
    .add("value", data, "world")
    .set_series_opts(label_opts=opts.LabelOpts(is_show=False))
    .set_global_opts(
        title_opts=opts.TitleOpts(title="Map_World"),
        visualmap_opts=opts.VisualMapOpts(max_=200),
    )
)
chart.render('map.html')
```

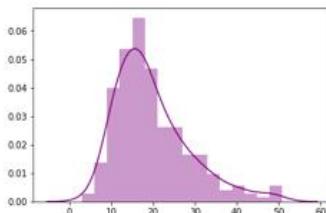


## IV seaborn 库简介

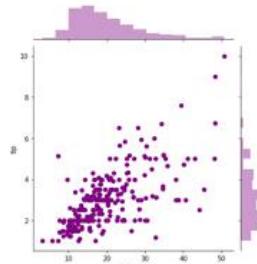
# 什么是 seaborn?

**seaborn** 是基于 matplotlib 的统计数据可视化 Python 库，进行了相比 matplotlib 更高级的 API 封装。

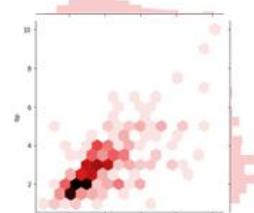
- seaborn 提供了高度交互式界面，能作出 more attractive、more informative 的图
- 应该把 seaborn 视为 matplotlib 的补充，而不是替代物。
- seaborn 能高度兼容 numpy 与 pandas 数据结构、scipy 与 statsmodels 等统计模式。掌握 seaborn 能很大程度帮助我们更高效的观察数据与图表，并且更加深入了解它们。



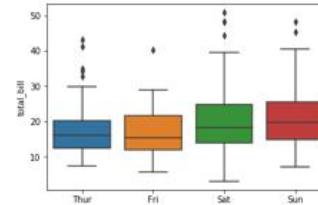
distplot



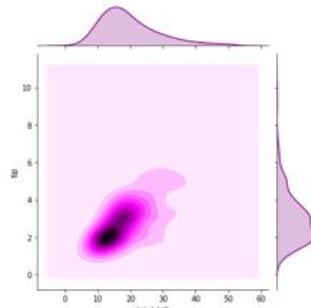
Jointplot



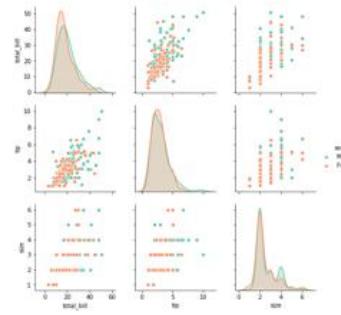
Hexplots



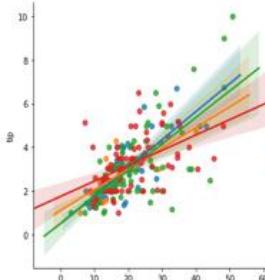
Boxplots



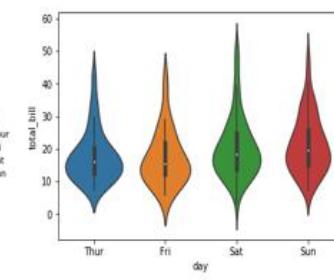
KDE Plot



Pair Plots



LM Plots



Violin Plots

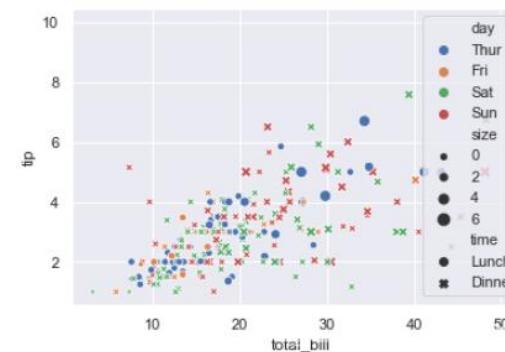
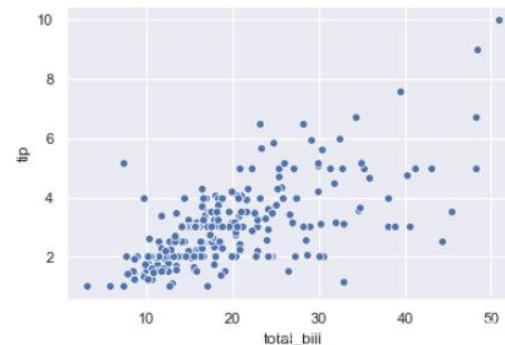
# seaborn 能画什么？

此处仅作简要介绍，具体可参考：

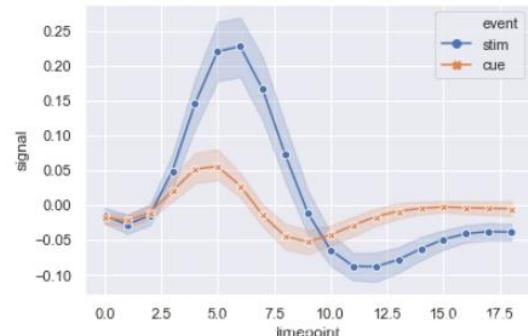
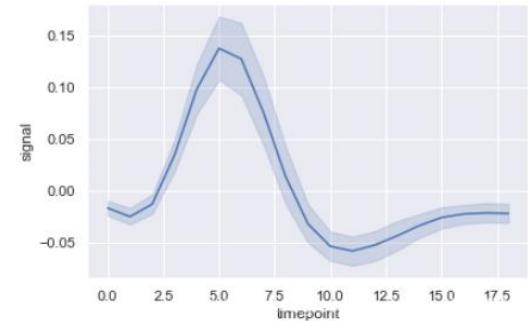
- 官方文档：<https://seaborn.pydata.org/>
- 一篇写的很好的博客：[https://blog.csdn.net/qq\\_40195360/article/details/86605860](https://blog.csdn.net/qq_40195360/article/details/86605860)

## ① 关系图 (Relational plots)

当数据被正确地可视化时，人类视觉系统可以看到指示某种关系的趋势和模式。



散点图：sns.scatterplot()



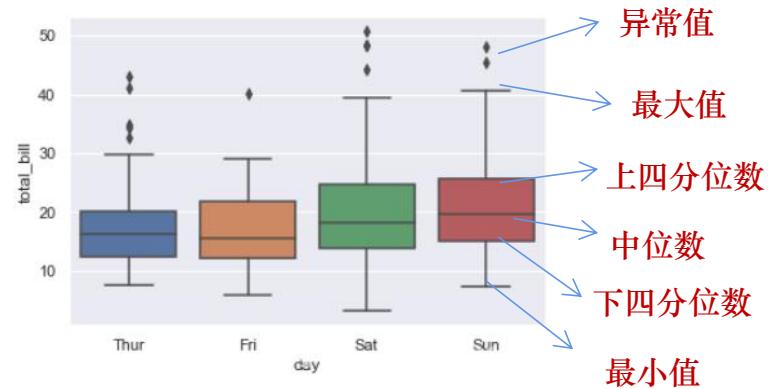
线图：sns.lineplot()

# seaborn 能画什么？

## ② 分类分布图 (Categorical distribution plots)

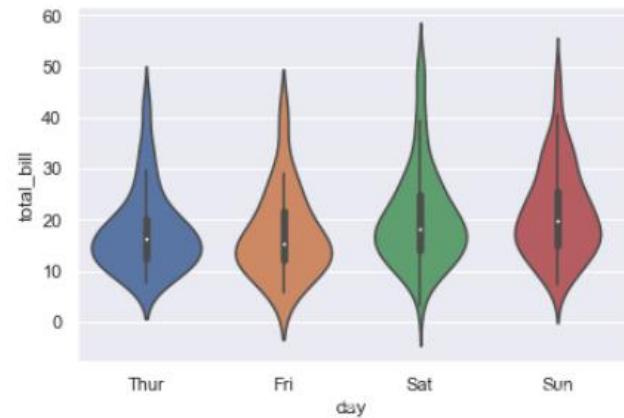
### 箱线图：sns.boxplot()

- boxplot 便于在变量之间或跨类别变量级别比较的方式，显示定量数据的分布情况。
- 框显示数据集的四分位数，线显示分布的其余部分，它能显示出一组数据的最大值、最小值、中位数及上下四分位数，使用四分位数范围函数的方法可以确定“离群值”的点。



### 小提琴图：sns.violinplot()

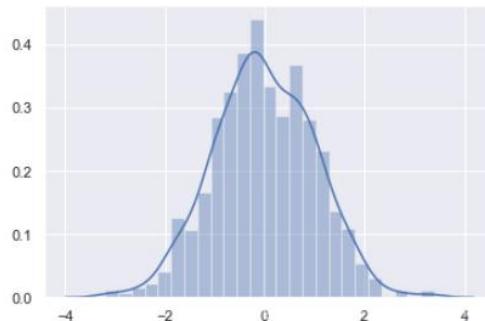
- violinplot 以基础分布的核密度估计为特征，通过小提琴图可以知道哪些位置的密度较高。
- 在图中，白点是中位数，黑色盒型的范围是下四分位点到上四分位点，细黑线表示 95% 置信区间。外部形状即核密度估计。



# seaborn 能画什么？

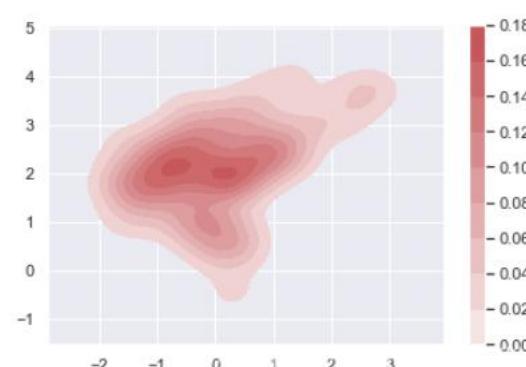
## ③ 分布图 (Distribution plots)

直方图 + 核密度: `sns.distplot()`



核密度图: `sns.kdeplot()`

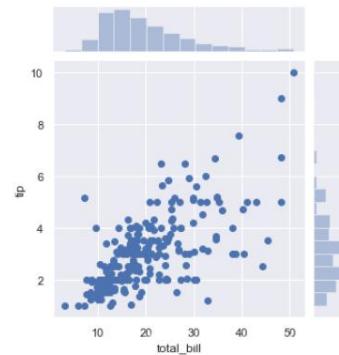
以下为双变量核密度图:



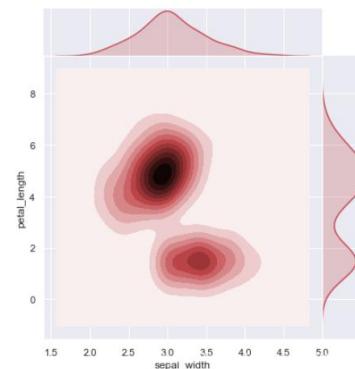
联合分布图:

(1) `sns.jointplot(kind='scatter')`:

用边缘直方图绘制散点图



(2) `sns.jointplot(kind='kde')`: 用核密度估计代替

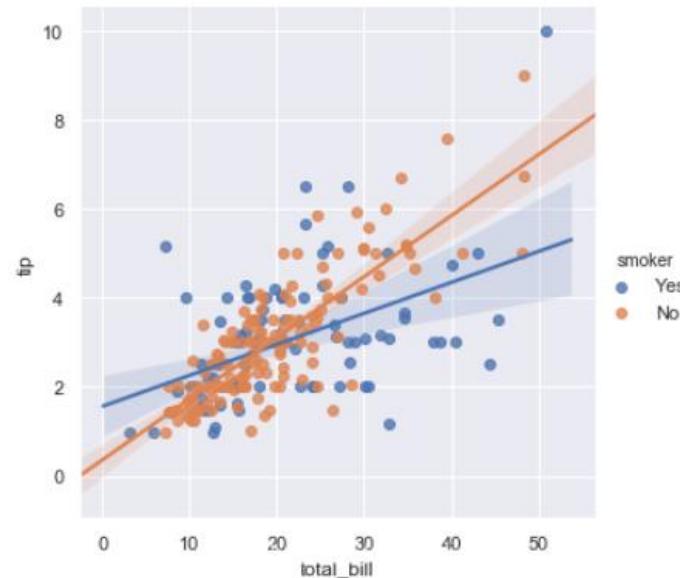


seaborn 能画什么？

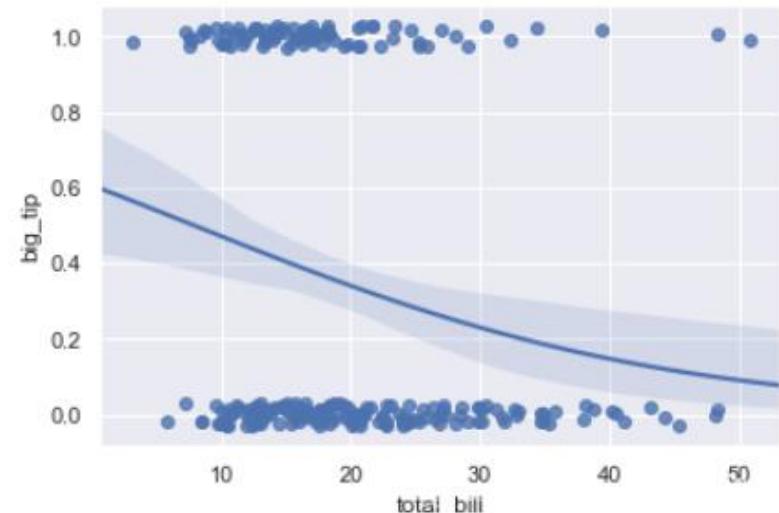
## ④ 回归图 (Regression plots)

回归图: sns.lmplot()

- 支持线性回归、多项式回归和逻辑回归
- 自动生成95%置信带



线性回归

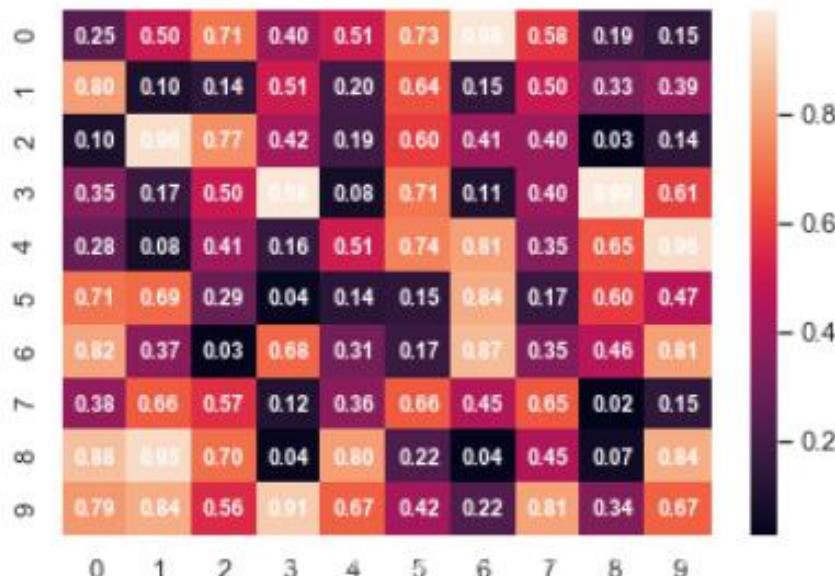


Logistic回归实现分类

# seaborn 能画什么？

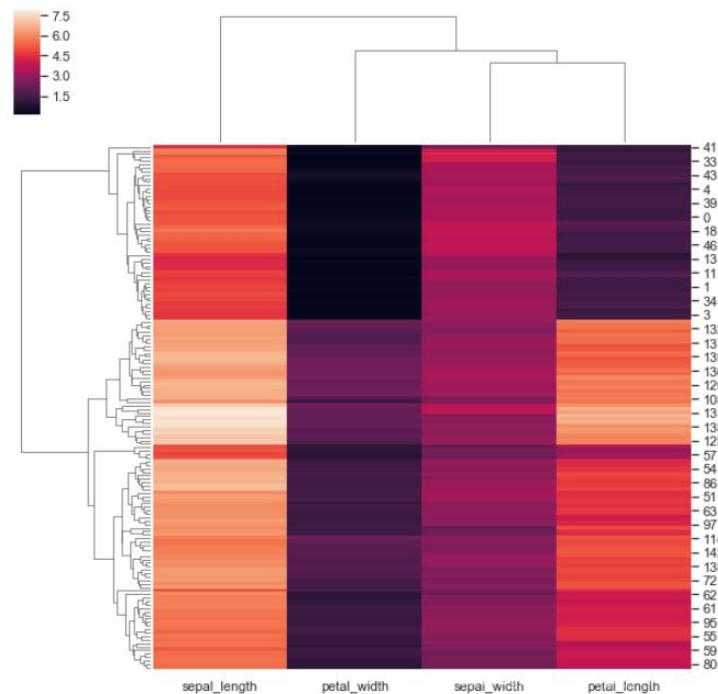
## ⑤ 矩阵图 (Matrix plots)

热力图: sns.heatmap( )



利用热力图可以看数据表里  
多个特征两两的相似度

聚类图: sns.clustermap( )



利用iris数据集生成层次聚类热图

Python 提供了丰富的数据可视化方法，  
不必记住每一个方法的具体使用，  
不懂的学会查官方文档，  
需要大家多多实践！

*Thanks for Listening!*