简介

一个类似MATLAB的第三方库，将MATLAB的功能制作成python第三方库，主要用来实现数据可视化图表，可以绘制散点图，直方图等高线等各种能想到的图，具体在matplotlib官方文档中有demo。

一篇总结

一篇不错的matplotllib库总结——来自简书

基本要点

模块之pyplot

负责绘图的模块

```
# 导入绘图模块
from matplotlib import pyplot as plt
```

plt.subplot 划分区域绘制子图

参数说明：

```
plt.subplot(
numbRow ,  # 划分为几行
numbCol ,  # 划分为几列
plotNum   # 当前子图的序号(从左到右，从下到上来排列，从1开始计数)
)
```

代码示例：

```
# 这里两行两列显示4张图片x
for i in [1, 2]:   # 控制行号
    for j in [1, 2]:   # 控制列号
        plt.subplot(2, 2, j + (i-1) * 2)   # 在第j + (i-1) * 2个位置绘制子图
        '''
        划分为(2,2)后序号的排列应是这个样子：
        1   2
        3   4
        '''
        plt.imshow(x)   # 在当前子区域显示图片x
        plt.axis('off')   # 关闭坐标轴
plt.show()   # 显示figure
```

plt.plot 绘制折线图

应用场景：

- 观察数据变化趋势
- 寻找极值

```
# 散点x值和y值，数量要相等，能一一对应
x = [1, 6, 5, 1, 10, 25, 4]
y = [1, 2, 5, 3, 20, 6, 15]
# 传入x和y，绘制出折线图，还有三个可选参数，linestyle(线条样式)、marker(点样式)、color(线条颜色)
plt.plot(x, y, color="b",lable = "温度折线")
plt.plot(x, y2 ,lable = "湿度折线")  # 在同一图中绘制第二条线
# 展示图形
plt.show()
```

绘图细节

1. 设置图片大小
2. 保存图片到本地
3. 添加描述信息（轴标题，图标题）
4. 调整x或y的刻度间距
5. 线条样式（颜色，透明度等）
6. 特殊点标记（极值等）
7. 图片加水印

   ……

调整图片大小

```
# fig对象，通过创建fig对象调整图片大小以及其他属性
def figure(num=None,  # 图片对象的ID，不指定则自增
figsize=None,  #图片框的尺寸，输入为tuple类型，(x,y)单位为英寸
dpi=None,  #图片分辨率，每英寸像素点的个数，接受一个整数
facecolor=None,  # 背景颜色
edgecolor=None,  # 边框颜色
frameon=True,  # 是否绘制边框
FigureClass=Figure,
clear=False,  # 如果figure已存在，是否清除该figure上已绘制的图形
**kwargs
):
# 附注：如果创建了多个figure实例，必须调用 plt.close() 来释放你已经不再使用的 figure 实例。因为只有这样 pylab 才能正确的释放内存。
```

保存图片到本地

```
plt.savefig(path)  # 保存图片到本地，可以保存为.svg矢量图格式，文件小，放大不失真
```

设置轴刻度

```
# 设置x、y轴刻度，接受两个可迭代对象，第一个作为刻度点，第二个作为刻度点对应的刻度标签，如只有单个参数，则标签就是刻度值
plt.xticks([-np.pi, -np.pi / 2, 0, np.pi / 2, np.pi * 3 / 4,
            np.pi],
           [r'$-\pi$', r'$-\frac{\pi}{2}$', r'$0$',
            r'$\frac{\pi}{2}$', r'$\frac{3\pi}{4}$',
            r'$\pi$'],
            rotation=90)  # rotation 标签旋转的角度
plt.yticks(range(50))
```

设置显示字体

matplotlib默认字体==中文不显示==
第一种：通过matplotlib.rc修改，对系统有要求，不通用

```
matplotlib.rc("font",family = 'MicroSoft YaHei',weight = "bold")  # 包含很多关于字体设置的参数可以修改，如字体family，大小size，线条等等,源码有demo
```

第二种：设置字体的方法，通用(推荐使用)

```
# 导入字体管理模块
from matplotlib import font_manager
# 实例化一个字体管理对象，参数fname为字体文件的路径
my_font = font_manager.FontProperties(fname = fontParh)
# 在需要使用时将该对象传入参数fontproperties，如设置刻度等
plt.xticks(xlist,lable_list,rotation = 45,fontproperties = my_font)
```

设置正常显示负号

```
matplotlib.rcParams['axes.unicode_minus']=False      # 正常显示负号
```

添加描述信息(轴、图标题)

```
plt.xlabel("时间",fontproperties = my_font)  # 同样要设置中文字体，否则中文无法显示
plt.ylabel("温度 单位(℃)",fontproperties =  my_font)
plt.title("10点至12点 气温/分钟",fontproperties =  my_font)  # 设置图标题
```

绘制网格

```
plt.grid(
alpha = 0.4 ,  # 不透明度
linestyle = '-.'  # 线型
)  # 以x刻度和y刻度来绘制网格线，还有一些其他参数
```

绘制多条线并添加图例

```
#  在同一图中绘出折线“张三”和“李四”
plt.plot(x,y1,lable="张三")
plt.plot(x,y2,lable="李四")

# 添加图例，前提是plot调用时传入了lable标签
plt.legend(
prop = my_font,  # 同样的问题，不加prop参数无法显示中文(线条的lable)，但要注意，只有这里参数名为prop，其他都为fontproperties
loc = 0  # loc参数 指定位置，默认为best即0，具体可选值查看源码
)
```

---

plt.scatter 绘制散点图

==与折线图唯一区别就是绘图方法不同== 应用场景：

- 不同条件(维度)之间的内在关联关系
- 观察数据的离散聚合情况

```
x1 = [2,5,2,45,23,4,2,1,5,45]
y1 = [21,41,56,48,2,4,6,95,45,12]
x2 = range(1,20,2)
y2 = range(1,10)
# 绘制散点图
plt.scatter(x1,y1)
plt.scatter(x2,y2)
# 展示图形
plt.show()
```

---

plt.bar 绘制条形图

应用场景：

- 数量统计
- 频率统计(市场饱和度)

```
# x轴标签(直方图中的类别)
x1 = ["张三","李四","王五"]
y1 = [21,52,13]
# 以x元素的各下标为刻度，以y1为各类别的值绘制条形图
plt.bar(range(len(x1)),y1,
width = 0.2  # 设置条形宽度
)
# 将x作为刻度标签添加到图中
plt.xticks(range(len(x1)),x1,fontpropreties = my_font,rotation = 45)
```

---

plt.barh 绘制横条形图

这里与竖条形图只有细微差别，但要注意，==否则容易报错==

```
plt.barh(range(len(x1)),y1,
height = 0.2  # 这里width变成了height,默认为0.8
)
# 将x作为刻度标签添加到图中
plt.yticks(range(len(x1)),x1,fontpropreties = my_font)  # 这应将刻度画在y轴，所以调用yticks
```

---

plt.hist 绘制直方图

直方图与条形图的区别

	直方图	条形图
横轴	数值型数据，根据范围来连续划分	是一个个类别，非连续
矩形分布	无空隙，连续	有空隙，不连续
纵轴	一般纵轴为 频数/组距	一般表示频数

```
"""
绘制直方图 plt.hist
```

```
data:必选参数，绘图数据
bins:直方图的长条形数目，可选项，默认为10
normed:是否将得到的直方图向量归一化，可选项，默认为0，代表不归一化，显示频数。normed=1，表示归一化，显示频率。
facecolor:长条形的颜色
edgecolor:长条形边框的颜色
alpha:透明度
"""
# 需要统计的数据
population_ages =
[22,55,62,45,21,22,34,42,42,4,99,102,110,120,121,122,130,111,115,112,80,75,65,54,44,43,42,48]
# 各组的分隔点,这种方式传参相当于制定好每一组的分组范围
bins = [0,10,20,30,40,50,60,70,80,90,100,110,120,130]   # 也可以直接传数字 bins = 13即从数据最小值到最大值平均分为13组
plt.hist(population_ages, bins, histtype='bar', rwidth=0.8)  #
```

总结

使用时应先明确问题，分析需要可视化的数据，据此选择出最合适的图表形式，最后再绘制图表。

一些其他的可视化工具：

- ECharts 在线可视化绘图工具，图形绘制使用JS交互的形式。
- plotly 非常好的一个绘图工具库，相比于matplotlib更加简单，图形更漂亮，同时兼容matplotlib和pandas，使用方法也非常简单，官方文档十分详细。
  plotly官方文档