



matplotlib



matplotlib

- Matplotlib是非常强大的python画图工具
- 是Python最常用的绘图库，提供了一整套十分适合交互式绘图的命令
- API，比较方便的就可以将其嵌入到GUI应用程序中。
- Matplotlib可以画图线图、散点图、等高线图、条形图、柱形图、3D图形、图形动画等。
- 官网：<https://matplotlib.org/>

matplotlib安装

安装方式：

- Python的Anaconda发行版，已经安装好matplotlib库，不需要另外安装
- 使用Anaconda界面安装，选择对应的matplotlibs进行勾选安装即可
- 使用Anaconda命令安装：`conda install matplotlib`
- 使用PyPi安装命令安装：`pip install matplotlib`

matplotlib基本使用

1、使用import导入模块matplotlib.pyplot，并简写成plt 使用import导入模块numpy，

并简写成np

2、接下来，我们调用plot的.plot方法绘制一些坐标。 这个.plot需要许多参数，但前两个是'x'和'y'坐标，我们放入列表。 这意味着，根据这些列表我们拥有 3 个坐标：1,5 2,7和3,4。

3、plt.plot在后台『绘制』这个绘图，但绘制了我们想要的一切之后，当我们准备好的时候，我们需要把它带到屏幕上。


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

```
import numpy as np
```

```
plt.plot([1,2,3],[5,7,4])
```

```
plt.show()
```

设置在jupyter中matplotlib的显示情况

1.%matplotlib tk 在GUI中显示  弹出一个窗口

2.%matplotlib inline 在行内显示

默认行

matplotlib基本使用-figure对象

没有创建会默认

想要使用matplotlib绘图，必须先要创建一个figure(画布)对象，然后还要有axes(坐标系)。但是观察上述代码，我们并没有创建figure对象。

对于上述疑问，接下来我们就要讲述创建figure(画布)的两种方式。

① 隐式创建figure对象

当第一次执行plt.xxx()画图代码时，系统会去判断是否已经有了figure对象，如果没有，系统会自动创建一个figure对象，并且在这个figure之上，自动创建一个axes坐标系(注意：默认创建一个figure对象，一个axes坐标系)。也就是说，如果我们不设置figure对象，那么一个figure对象上，只能有一个axes坐标系，即我们只能绘制一个图形。

② 显示创建figure对象：以后的文章会详细讲述布局设置

```
figure = plt.figure()
```

```
axes1 = figure.add_subplot(2,1,1)
```

```
axes2 = figure.add_subplot(2,1,1)
```

```
axes1.plot([1,3,5,7],[4,9,6,8])
```

```
axes2.plot([1,2,4,5],[8,4,6,2])
```

```
figure.show()
```

1. 为横着的几部分 → 若改为 (2,2,1) 则分成了 2 行
① → 改为几列
② → 改为 (2,1,2) 则是上下
折线图 → 为几行 (以为为井头) 相互独立的
第几行画了一半

matplotlib基本使用-figure对象

函数:

```
def figure(num=None, figsize=None, dpi=None, facecolor=None, edgecolor=None,
frameon=True, FigureClass=Figure, clear=False, **kwargs)
```

```
import numpy as np
```

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

```
%matplotlib tk
```

```
fig1 = plt.figure(num='fig1',figsize=(6,9))
```

```
plt.xlim(0,20)
```

```
plt.ylim(0,10)
```

```
plt.show()
```

名字

宽度
高度

x轴范围

y轴范围

matplotlib基本使用

函数: plot

```
plot(x,y,color='red', linestyle='dashed', marker='o'.....)
```

绘图中用到的直线属性包括:

(1) LineStyle: 线形

(2) LineWidth: 线宽

(3) Color: 颜色

(4) Marker: 标记点的形状

(5) label: 用于图例的标签

matplotlib基本使用

`legend ()`: 生成默认图例, matplotlib 中的 `legend` 图例就是为了帮我们展示出每个数据对应的图像名称. 更好的让读者认识到你的数据结构.

`xlabel`、`ylabel`:设置X轴Y轴标签

`title`: 设置标题

`xlim`、`ylim`: 控制图标的范围

`xticks`、`yticks`: 控制图标的刻度

`gca`获取当前坐标轴信息。使用`spines`设置边框,使用`set_color`设置边框颜色:

默认白色

解决中文显示问题

```
mpl.rcParams['font.sans-serif'] = ['SimHei']
```

```
mpl.rcParams['axes.unicode_minus'] = False
```


matplotlib基本使用

```
import matplotlib.pyplot as plt

import numpy as np

# 调用figure函数创建figure(1)对象,可以省略

# 这样那plot时,它就自动建一个
plt.figure('正弦曲线') 图像名字
x=np.linspace(-1,1,50)#定义x数据范围
y1=2*x+1#定义y数据范围
y2=x**2

plt.figure()#定义一个图像窗口

plt.plot(x,y1)#plot()画出曲线

plt.plot(x,y2)#plot()画出曲线

plt.show()#显示图像
```

```
import matplotlib.pyplot as plt

import numpy as np
x=np.linspace(-3,3,50)#50为生成的样本数
y1=2*x+1
y2=x**2
plt.figure(num=1,figsize=(8,5))#定义编号为1 大小
为(8,5)
plt.plot(x,y1,color='red',linewidth=2,linestyle='--')#
颜色为红色, 线宽度为2, 线风格为--
plt.plot(x,y2,color='green', marker='o', 标识形状,
linestyle='dashed', linewidth=1, markersize=6),
#dashed虚线 也可以linestyle=':' 标识节点大小
plt.show()#显示图
```

matplotlib基本使用-线条属性设置

mark 参数设置

'.'	point marker
','	pixel marker
'o'	circle marker
'v'	triangle_down marker
'^'	triangle_up marker
'<'	triangle_left marker
'>'	triangle_right marker
'1'	tri_down marker
'2'	tri_up marker
'3'	tri_left marker
'4'	tri_right marker
's'	square marker
'*'	star marker
'+'	plus marker
'x'	x marker
'D'	diamond marker
'd'	thin_diamond marker

线条颜色设置

别名	颜色名
b	蓝色
g	绿色
r	红色
y	黄色
c	青色
k	黑色
m	洋红色
w	白色

线条风格linestyle或ls 描述

'-'	solid line style 实线
'--'	dashed line style 虚线
'-.'	dash-dot line style 点画线
'.'	dotted line style 点线

matplotlib基本使用-设置坐标轴

```
x=np.linspace(-3,3,50)

y1=2*x+1

y2=x**2

plt.figure(num=2,figsize=(8,5))

plt.plot(x,y1,color='red',linewidth=2,linestyle='-')

plt.plot(x,y2)#进行画图

plt.xlim(-1,2)    #设置x轴显示范围

plt.ylim(-2,3)    #设置y轴显示范围

plt.xlabel("I'm x")    #设置x轴名称

plt.ylabel("I'm y")    #设置y轴名称

plt.show()
```

matplotlib基本使用-自定义坐标轴

```
x=np.linspace(-3,3,50)
```

```
y1=2*x+1
```

```
y2=x**2
```

```
plt.figure(num=2,figsize=(8,5))
```

```
plt.plot(x,y1,color='red',linewidth=2,linestyle='-')
```

```
plt.plot(x,y2)#进行画图
```

```
plt.xlim(-1,2)
```

```
plt.ylim(-2,3)
```

```
plt.xlabel("I'm x")
```

```
plt.ylabel("I'm y")
```

```
new_ticks=np.linspace(-1,2,4)#小标从-1到2分为5个单位
```

```
plt.xticks(new_ticks)#进行替换新下标
```

```
plt.yticks([-2,-1,1,2,], [r'$really\ bad$', '$bad$', '$well$', '$really\ well$'])
```

对单单词没影响，没有美元符号，
直接有空格

matplotlib基本使用-设置边框属性

```
x=np.linspace(-3,3,50)

y1=2*x+1

y2=x**2

plt.figure(num=2,figsize=(8,5))

plt.plot(x,y1,color='red',linewidth=2,linestyle='--')

plt.plot(x,y2)#进行画图

plt.xlim(-1,2)

plt.ylim(-2,3)

new_ticks=np.linspace(-1,2,5)#小标从-1到2分为5个单位

plt.xticks(new_ticks)#进行替换新下标

plt.yticks([-2,-1,1,2,], [r'$really\ bad$', '$bad$', '$well$', '$really\ well$'])

ax=plt.gca()#gca=get current axis

ax.spines['right'].set_color('none')#边框属性设置为none 不显示

ax.spines['top'].set_color('none')

plt.show()
```

有只是设为透明了.

matplotlib基本使用-添加图例

有
loc可改. 1 → 边缘

matplotlib中legend图例帮助我们展示数据对应的图像名称

```
x=np.linspace(-3,3,50)
```

```
y1=2*x+1
```

```
y2=x**2
```

```
plt.figure(num=2,figsize=(8,5))
```

```
plt.xlim(-1,2)
```

```
plt.ylim(-2,3)
```

```
new_ticks=np.linspace(-1,2,5)#小标从-1到2分为5个单位
```

```
plt.xticks(new_ticks)#进行替换新下标
```

```
plt.yticks([-2,-1,1,2],
```

```
[r'$really\ bad$', '$bad$', '$well$', '$really\ well$'])
```

```
l1=plt.plot(x,y1,color='red',linewidth=2,linestyle='--',label='linear line')
```

```
l2=plt.plot(x,y2,label='square line')#进行画图
```

```
plt.legend(loc='best')#显示在最好的位置
```

```
plt.show()#显示图
```

要说明

图例

属性

matplotlib基本使用-添加图例

调整位置和名称，单独修改label信息，我们可以在plt.legend输入更多参数

```
plt.legend(handles=[l1, l2], labels=['up', 'down'], loc='best')
```

```
#或plt.legend(handles=[l1, l2], labels=['up', 'down'], loc=0)
```

#loc有很多参数 其中best自分配最佳位置

```
'best' : 0,
```

```
'upper right' : 1,
```

```
'upper left' : 2,
```

```
'lower left' : 3,
```

```
'lower right' : 4,
```

```
'right' : 5,
```

```
'center left' : 6,
```

```
'center right' : 7,
```

```
'lower center' : 8,
```

```
'upper center' : 9,
```

```
'left' : 10,
```

画图种类-Scatter散点图

$n=1024$

$X = \text{np.random.normal}(0, 1, n)$ # 每一个点的X值

$Y = \text{np.random.normal}(0, 1, n)$ # 每一个点的Y值

$T = \text{np.arctan2}(Y, X)$ # arctan2 返回给定的X和Y值的反正切值

scatter画散点图 size=75 颜色为T 透明度为50% 利用xticks函数来隐藏x坐标轴

$\text{plt.scatter}(X, Y, s=75, c=T, \text{alpha}=0.5)$

$\text{plt.xlim}(-1.5, 1.5)$

$\text{plt.xticks}()$ # 忽略xticks

$\text{plt.ylim}(-1.5, 1.5)$

$\text{plt.yticks}()$ # 忽略yticks

$\text{plt.show}()$

正态分布

个数

范围 $(-20, 20)$

和位置有关, 改变颜色

也可 $s = Y = \text{np.random.normal}(0, 1, n)$
 $c = s$

画图种类-条形图

关于X轴对称

n=12

X=np.arange(n)

均匀分布

Y1=(1-X/float(n))*np.random.uniform(0.5,1,n)

Y2=(1-X/float(n))*np.random.uniform(0.5,1,n)

plt.bar(X,+Y1,facecolor='#9999ff',edgecolor='white')

plt.bar(X,-Y2,facecolor='#ff9999',edgecolor='white')

for x,y in zip(X,Y1):#zip表示可以传递两个值

plt.text(x+0.4,y+0.05,'%.2f'%y,ha='center',va='bottom')#ha表示横向对齐 bottom表示向下对齐

for x,y in zip(X,Y2):

plt.text(x+0.4,-y-0.05,'%.2f'%y,ha='center',va='top')

plt.xlim(-0.5,n)

plt.xticks(())#忽略xticks

plt.ylim(-1.25,1.25)

plt.yticks(())#忽略yticks

plt.show()

位置

X轴方向

Y轴方向

相对于 -y-0.0 与这条线; 线在上面

画图种类-条形图

两者都在X轴上方

```
plt.figure(3)
```

```
x_index = np.arange(5) #柱的索引
```

```
x_data = ('A', 'B', 'C', 'D', 'E')
```

```
y1_data = (20, 35, 30, 35, 27)
```

```
y2_data = (25, 32, 34, 20, 25)
```

```
bar_width = 0.35 #定义一个数字代表每个独立柱的宽度
```

```
rects1 = plt.bar(x_index, y1_data, width=bar_width,alpha=0.4, color='b',label='legend1')  
#参数: 左偏移、高度、柱宽、透明度、颜色、图例
```

#参数: 左偏移、高度、柱

```
rects2 = plt.bar(x_index + bar_width, y2_data, width=bar_width,alpha=0.5,color='r',label='legend2') #参数: 左偏移、高度、柱宽、透明度、颜色、图例
```

#关于左偏移, 不用关心每根柱的中心不中心, 因为只要把刻度线设置在柱的中间就可以了

```
plt.xticks(x_index + bar_width/2, x_data) #x轴刻度线
```

```
plt.legend() #显示图例
```

```
plt.tight_layout() #自动控制图像外部边缘, 此方法不能够很好的控制图像间的间隔
```

```
plt.show()
```

画图种类-等高线图

```
n=256
```

```
x=np.linspace(-3,3,n)
```

```
y=np.linspace(-3,3,n)
```

```
X,Y=np.meshgrid(x,y)#meshgrid从坐标向量返回坐标矩阵
```

```
#f函数用来计算高度值 利用contour函数把颜色加进去 位置参数依次为x,y,f(x,y), 透明度为0.75, 并将f(x,y)的值对应到cmap之中
```

```
def f(x,y):
```

```
    return (1 - x / 2 + x ** 5 + y ** 3) * np.exp(-x ** 2 - y ** 2)
```

```
plt.contourf(X,Y,f(X,Y),8,alpha=0.75,cmap=plt.cm.hot)#8表示等高线分成多少份 alpha表示透明度 cmap表示color map
```

```
#使用plt.contour函数进行等高线绘制 参数依次为x,y,f(x,y), 颜色选择黑色, 线条宽度为0.5
```

```
C=plt.contour(X,Y,f(X,Y),8,colors='black',linewidth=0.5)
```

```
#使用plt.clabel添加高度数值 inline控制是否将label画在线里面, 字体大小为10
```

```
plt.clabel(C,inline=True,fontsize=10)
```

```
plt.xticks(())#隐藏坐标轴
```

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

```
plt.show()
```

画图种类-子图

```
plt.figure()
```

```
plt.subplot(2,1,1)#表示整个图像分割成2行2列，当前位置为1
```

```
plt.plot([0,1],[0,1])#横坐标变化为[0,1] 竖坐标变化为[0,2]
```

```
plt.subplot(2,3,4)
```

```
plt.plot([0,1],[0,2])
```

```
plt.subplot(2,3,5)
```

```
plt.plot([0,1],[0,3])
```

```
plt.subplot(2,3,6)
```

```
plt.plot([0,1],[0,4])
```

```
plt.show()
```

画图种类-图中图

```
fig=plt.figure()

x=[1,2,3,4,5,6,7]

y=[1,3,4,2,5,8,6]

#绘制大图：假设大图的大小为10，那么大图被包含在由(1,1)开始，宽8高8的坐标系之中。

left, bottom, width, height = 0.1, 0.1, 0.8, 0.8

ax1 = fig.add_axes([left, bottom, width, height]) # main axes

ax1.plot(x, y, 'r')#绘制大图，颜色为red

ax1.set_xlabel('x')#横坐标名称为x

ax1.set_ylabel('y')

ax1.set_title('title')#图名称为title

#绘制小图，注意坐标系位置和大小改变

ax2 = fig.add_axes([0.2, 0.6, 0.25, 0.25])

ax2.plot(y, x, 'b')#颜色为blue

ax2.set_xlabel('x')

ax2.set_ylabel('y')

ax2.set_title('title inside 1')

#绘制第二个小兔

plt.axes([0.6, 0.2, 0.25, 0.25])

plt.plot(y[::-1], x, 'g')#将y进行逆序
```

次坐标轴

```
x=np.arange(0,10,0.1)
```

```
y1=0.5*x**2
```

```
y2=-1*y1
```

```
fig, ax1 = plt.subplots()
```

```
ax2 = ax1.twinx()#镜像显示
```

```
ax1.plot(x, y1, 'g-')
```

```
ax2.plot(x, y2, 'b-')
```

```
ax1.set_xlabel('X data')
```

```
ax1.set_ylabel('Y1 data', color='g')#第一个y坐标轴
```

```
ax2.set_ylabel('Y2 data', color='b')#第二个y坐标轴
```

```
plt.show()
```

THANKS!

