# 探索性分析 Matplotlib基础

# 目录


数据可视化与Matplotlib

pyplot基础语法

rc参数

绘图注释

小结

# 数据可视化

对数据的分析离不开数据的可视化。传统的数据可视化起源于统计图形学，与信息图形、视觉设计等现代技术密切相关，其表现形式通常在二维空间。与之相比，大数据可视化往往更关注抽象高维的数据，空间属性较弱，与所针对的数据类型密切相关。

# Matplotlib数据可视化基础

- **Matplotlib 是一个在 python 下实现的类似 matlab 的纯 python 的第三方库,旨在用 python实现 matlab 的功能,是python下最出色的绘图库。其风格跟 matlab 相似，同时也继承了 python 的简单明了。**
- **matplotlib 对于图像美化方面比较完善，可以自定义线条的颜色和样式，可以在一张绘图纸上绘制多张小图，也可以在一张图上绘制多条线， 可以很方便地将数据可视化并对比分析。**

# Matplotlib数据可视化基础

- Matplotlib模块依赖于NumPy和tkinter模块，可以绘制多种形式的图形，包括线图、直方图、饼图、散点图等，图形质量满足出版要求，是数据可视化的重要工具。
- **Matplotlib中应用最广的是matplotlib.pyplot模块**。Pyplot提供了一套和Matlab类似的绘图API，使得Matplotlib的机制更像Matlab。我们只需要调用Pyplot模块所提供的函数就可以实现快速绘图并设置图表的各个细节。
- 在Jupyter notebook中进行交互式绘图，需要执行一下语句：

```
% matplotlib notebook
```

- 使用matplotlib时，使用的导入惯例为：

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

# Matplotlib绘图基础


Matplotlib中的常用绘图及分组

# 掌握pyplot基础语法

## 1.创建画布与创建子图

| 函数名称 | 函数作用 |
| --- | --- |
| plt.figure | 创建一个空白画布，可以指定画布大小，像素。 |
| figure.add_subplot | 创建并选中子图，可以指定子图的行数，列数，与选中图片编号。 |

第一部分主要作用是构建出一张空白的画布，并可以选择是否将整个画布划分为多个部分，方便在同一幅图上绘制多个图形的情况。最简单的绘图可以省略第一部分，而后直接在默认的画布上进行图形绘制。

# 掌握pyplot基础语法

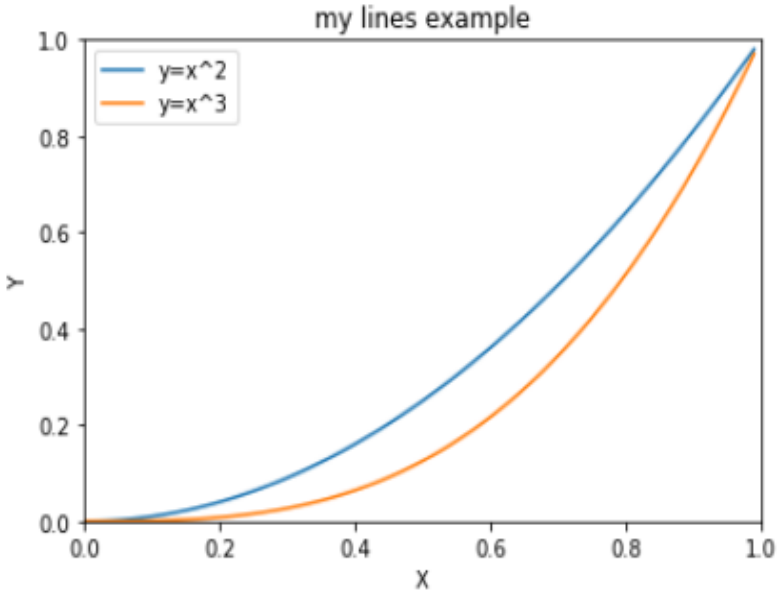
## 2.添加画布内容

| 函数名称 | 函数作用 |
| --- | --- |
| plt.title | 在当前图形中添加标题，可以指定标题的名称、位置、颜色、字体大小等参数。 |
| plt.xlabel | 在当前图形中添加x轴名称，可以指定位置、颜色、字体大小等参数。 |
| plt.ylabel | 在当前图形中添加y轴名称，可以指定位置、颜色、字体大小等参数。 |
| plt.xlim | 指定当前图形x轴的范围，只能确定一个数值区间，而无法使用字符串标识。 |
| plt.ylim | 指定当前图形y轴的范围，只能确定一个数值区间，而无法使用字符串标识。 |
| plt.xticks | 指定x轴刻度的数目与取值。 |
| plt.yticks | 指定y轴刻度的数目与取值。 |
| plt.legend | 指定当前图形的图例，可以指定图例的大小、位置、标签。 |

第二部分是绘图的主体部分。其中添加标题，坐标轴名称，绘制图形等步骤是并列的，没有先后顺序，可以先绘制图形，也可以先添加各类标签。但是添加图例一定要在绘制图形之后。

# 掌握pyplot基础语法

```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
data=np.arange(0,1,0.01)
plt.title('my lines example')
plt.xlabel('X')
plt.ylabel('Y')
plt.xlim(0,1)
plt.ylim(0,1)
plt.xticks([0,0.2,0.4,0.6,0.8,1])
plt.yticks([0,0.2,0.4,0.6,0.8,1])
plt.plot(data,data**2)
plt.plot(data,data**3)
plt.legend(['y=x^2','y=x^3'])
plt.show()
```



# 掌握pyplot基础语法

```
data=np.arange(0,np.pi*2,0.01)
fig1=plt.figure(figsize=(9,7),dpi=90)
#确定画布大小
#绘制第一幅子图
ax1=fig1.add_subplot(1,2,1)
plt.title('lines example')
plt.xlabel('X')
plt.ylabel('Y')
plt.xlim(0,1)
plt.ylim(0,1)
plt.xticks([0,0.2,0.4,0.6,0.8,1])
plt.yticks([0,0.2,0.4,0.6,0.8,1])
plt.plot(data,data**2)
plt.plot(data,data**3)
plt.legend(['y=x^2','y=x^3'])
```

```
#绘制第二幅子图
ax1=fig1.add_subplot(1,2,2)
plt.title('sin-cos')
plt.xlabel('X')
plt.ylabel('Y')
plt.xlim(0,np.pi*2)
plt.ylim(-1,1)
plt.xticks([0,np.pi/2,np.pi,np.pi*3/2,np.pi*2])
plt.yticks([-1,-0.5,0,0.5,1])
plt.plot(data,np.sin(data))
plt.plot(data,np.cos(data))
plt.legend(['sin','cos'])
plt.show()
```