Matplotlib

点击一键运行代码→ | 获取更多速查表→ | 使用说明书 [视频 & 图文]→ bilibili ShowMeAI

Matplotlib 是使用 Python 进行数据可视化最重要的基础工具库，Matplotlib 可以很好地和 Numpy 和 Pandas 进行兼容，方便在数据计算和探索分析的同时，以图示的方式呈现信息。

作者 | 韩信子 @ShowMeAI
设计 | 南 乔 @ShowMeAI
参考 | datacamp cheatsheet

使用下列别名导入该库

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

Matplotlib 绘图基本步骤与示例

Step 1 准备数据 Step 2 创建图形 Step 3 绘图 Step 4 自定义设置 Step 5 保存图形 Step 6 显示图形

```
> 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 3, 4
  ax.scatter([2, 4, 6], [5, 15, 25], color='darkgreen', marker='^')
  ax.set_xlim(1, 6.5)
  plt.savefig('foo.png')                              #Step 5
  plt.show()                                          #Step 6
```

1. 准备数据

一维数据

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

二维数据或图片

```
> 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. 绘制图形

```
> import matplotlib.pyplot as plt  # 导入库
```

画布

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

坐标轴

图形是以坐标轴为核心绘制的，大多数情况下子图就可以满足需求。

子图是栅格系统的坐标轴。

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

3. 绘图详解

一维数据

```
> fig, ax = plt.subplots()
> lines = ax.plot(x, y) # 用线或标记连接点
> ax.scatter(x, y) # 缩放或着色未连接的点
> axes[0, 0].bar([1, 2, 3], [3, 4, 5]) # 绘制柱状图
> axes[1, 0].barh([0.5, 1, 2.5], [0, 1, 2]) # 绘制水平柱状图
> axes[1, 1].axhline(0.45) # 绘制与轴平行的横线
> axes[0, 1].axvline(0.65) # 绘制与轴垂直的竖线
> ax.fill(x, y, color='blue') # 绘制填充多边形
> ax.fill_between(x, y, color='yellow') # 填充 y 值和 0 之间
```

Vector Fields - 向量场

```
> axes[0, 1].arrow(0, 0, 0.5, 0.5) # 为坐标轴添加箭头
> axes[1, 1].quiver(y, z) # 二维箭头
> axes[0, 1].streamplot(X, Y, U, V) # 二维箭头
```

Data Distributions - 数据分布

```
> ax1.hist(y) # 直方图
> ax3.boxplot(y) # 箱形图
> ax3.violinplot(z) # 小提琴图
```

二维数据或图片

```
> fig, ax = plt.subplots()
> im = ax.imshow(img, cmap='gist_earth', interpolation='nearest', vmin=-2, vmax=2)
> axes2[0].pcolor(data2) # 二维数组伪彩色图
> axes2[0].pcolormesh(data) # 二维数组等高线伪彩色图
> CS = plt.contour(Y, X, U) # 等高线图
> axes2[2].contourf(data1) # 等高线轮廓图
> axes2[2]= ax.clabel(CS) # 等高线图标签
```

5. 自定义图形

颜色、色条与色彩表

```
> 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')
```

标记

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

线型

```
> plt.plot(x, y, linewidth=4.0)
> plt.plot(x, y, ls='solid')
> plt.plot(x, y, ls='--')
> plt.plot(x, y, '--', x2, y2, '-.')
> plt.setp(lines, color='r', linewidth=4.0)
```

数学符号

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

尺寸限制、图例和布局

尺寸限制与自动调整

```
> ax.margins(x=0.0, y=0.1) # 添加内边距
> ax.axis('equal') # 将图形纵横比设置为 1
> ax.set(xlim=[0, 10.5], ylim=[-1.5, 1.5]) # 设置 x 轴与 y 轴的限制
> ax.set_xlim(0, 10.5) # 设置 x 轴的限制
```

图例

```
> ax.set(title='An Example Axes', ylabel='Y-Axis',  xlabel='X-Axis') # 设置标题与 x、y 轴的标签
> ax.legend(loc='best') # 自动选择最佳的图例位置
```

标记

```
> ax.xaxis.set(ticks=range(1, 5), ticklabels=[3, 100, -12, "foo"]) # 手动设置 X 轴刻度
> ax.tick_params(axis='y', direction='inout', length=10) # 设置 Y 轴长度与方向
```

子图间距

```
> fig3.subplots_adjust(wspace=0.5, hspace=0.3, left=0.125, right=0.9, top=0.9, bottom=0.1) # 调整子图间距
> fig.tight_layout() # 设置画布的子图布局
```

坐标轴边线

```
> ax1.spines['top'].set_visible(False) # 隐藏顶部坐标轴线
> ax1.spines['bottom'].set_position(('outward', 10)) # 设置底部边线的位置为 outward
```

文本与标注

```
> 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"), )
```


5. 保存

savefig 函数

```
> plt.savefig('foo.png') # 保存画布
> plt.savefig('foo.png', transparent=True) # 透明画布
```

6. 显示图形

show 函数

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

7. 关闭与清除

绘图清除与关闭

```
> plt.cla() # 清除坐标轴
> plt.clf() # 清除画布
> plt.close() # 关闭窗口
```