第七章 Matplotlib 数据可视化

7.1 Matplotlib 核心概念

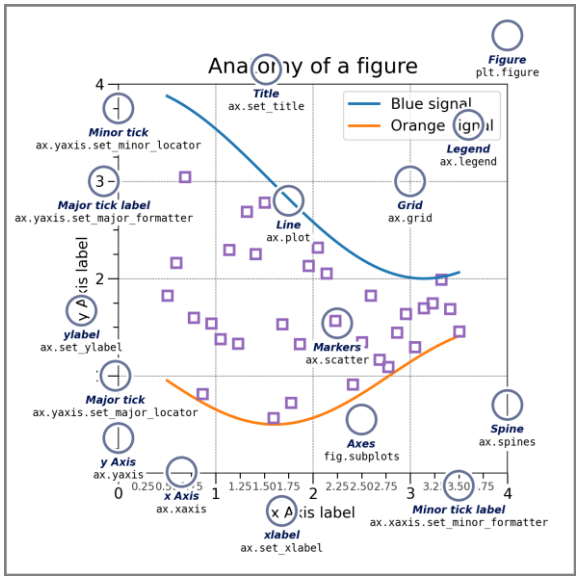
基本原则

① 颜色映射不能使用渐变色、颜色需要对比清晰
② 图注字号需要与正文一致或略小一号
③ 使用矢量图或高分辨率图片

7.1.1 核心概念

后端层 Backend Layer 处理绘图设备(屏幕、文件格式)的细节，用户不直接接触。
美工层 Artist Layer 包含构成图表的各种元素（图形、坐标系、线条等），大部分操作在这里。
脚本层 Scripting Layer 提供一个便捷的、状态驱动的接口（pyplot）。

7.1.2 基本组成元素



官方文档介绍

Figure 画布 整个图表的**顶层容器**，所有绘图元素都位于画布之上。 使用 `plt.figure()`创建。一个画布可以包含多个子图。
常用参数有：长宽(英寸)、dpi 分辨率、背景色

Axes 坐标系/子图 **实际进行数据绘制的区域**，通常包含两个（或三个，用于 3D 图）`axes` 对象。
大多数绘图方法都是 `axes` 对象的方法。
使用 `plt.subplots()`同时创建 `figure` 和一个或多个 `axes`。返回 `Figure` 对象和一个包含 `axes` 对象的 NumPy 数组（即使只有一个 `axes`）。

Axis 坐标轴 **数值线**（x轴，y轴，z轴），负责定义数据范围、生成刻度 (Ticks) 和刻度标签 (Tick Labels)。
通常通过 `axes` 对象访问和控制，例如 `ax.set_xlim()`, `ax.set_xlabel()`, `ax.xaxis`

Tick 刻度 坐标轴上的标记点，指示特定数值。包含**主要刻度、次要刻度**

Label 刻度标签 依附于刻度的文字，显示该刻度代表的数值。

Label 轴标签 描述坐标轴含义的文字，使用 `ax.set_xlabel()`，`ax.set_ylabel()`设置

Title 标题 `axes` 的标题，概括该子图的内容。使用 `ax.set_title()` 设置。
`Figure` 也可以有自己的大标题：`fig.suptitle()`。

Legend 图例 解释图中不同线条、标记或颜色的含义。在绘图函数中指定 label 参数，然后调用 `ax.legend()`

Grid 网格线 背景中的辅助线，有助于对齐和读取数值。 开启/关闭/定制：`ax.grid()`

Spines 边框线 包围 `axes` 区域的线条。可以控制其显示和位置。

颜色映射

7.1.3 两套接口

pyplot 接口 状态机。在处理多个图表或复杂逻辑时容易混淆当前状态，不够灵活，**不推荐使用**。
隐式地跟踪当前的 Figure 和 Axes，直接调用 `plt.plot()`, `plt.title()` 等函数。

面向对象 OO Object-Oriented 需要显式地创建对象。显式地创建和操作 Figure 和 Axes 对象。
调用这些**对象的方法**来绘图和定制，如 `ax.grid()`，`ax.set_title()`。
该接口使得代码更清晰、结构化，对图表元素有更强的控制力，特别适合复杂的图表、函数封装

和嵌入式应用。推荐在编写脚本和复杂应用时使用此接口。

```
x = np.linspace(0, 5, 10)
y = x  2

plt.figure()      # 隐式创建 Figure 和 Axes
plt.plot(x, y, 'r-') # 在当前 Axes 上绘图
plt.xlabel('X axis')
plt.ylabel('Y axis')
plt.title('Pyplot Interface Example')
plt.show()
```

```
# 1. 显式创建 Figure 和 Axes 对象
fig, ax = plt.subplots() # ★ 常用方法
# 2. 在指定的 Axes 对象上调用方法
ax.plot(x, y, 'b--')
ax.set_xlabel('X axis')
ax.set_ylabel('Y axis')
ax.set_title('Object-Oriented Interface Example')
plt.show() # pyplot 仍然用于显示图表
```

7.2 常见绘制图形

7.2.1 线图 ax.plot

适用 展示连续数据随某个变量（通常是时间或顺序）的变化趋势。

常用参数 color, linestyle (-, --, :, -.), linewidth, marker (o, s, ^, *, .), markersize, label

示例
```
ax.plot(x, y1, color='blue', linestyle='-', linewidth=2, marker='o', markersize=4, label='sin(x)')
```

7.2.2 散点图 ax.scatter

适用 用于观察两个变量之间的关系或数据的分布。可以将第三、第四个变量映射到点的大小和颜色上。

常用参数 x, y, s (大小), c (颜色), cmap (颜色映射), alpha (透明度), marker

cmap: Colormap (颜色映射方案) 'viridis', 'plasma', 'inferno', 'magma', 'cividis', 'coolwarm', 'jet' 等

示例
```
scatter = ax.scatter(x, y, c=colors, s=sizes, alpha=0.7, cmap='viridis', label='P')
```

7.2.3 柱状图 ax.bar, ax.barh

适用 比较不同类别之间的数值大小。

示例
```
rects1 = ax.bar(x_pos - bar_width/2, values1, bar_width, label='Dataset 1', color='steelblue')
rects2 = ax.bar(x_pos + bar_width/2, values2, bar_width, label='Dataset 2', color='lightcoral')
```

注意 ax.bar: 垂直柱状图。 ax.barh: 水平柱状图。

分组柱状图: 通过调整 x 轴位置 (x_pos - width/2, x_pos + width/2) 实现。

堆叠柱状图: 通过设置 bottom 参数实现 (ax.bar(x_pos, values2, bottom=values1))。

7.2.4 直方图 ax.hist

适用 展示数值型数据（连续或离散）的分布频率。

关键参数 bins (决定了分组的粒度), density (频率 vs. 频数密度), histtype.

示例
```
# bins: 区间数量或边界；density: 是否归一化(面积为 1)；alpha: 透明度；histtype: 'bar', 'step', 'stepfilled'
ax.hist(data1, bins=30, density=True, alpha=0.7, label='Distribution 1 (N(0,1))', color='skyblue', edgecolor='black')
```

7.2.5 饼图 ax.pie

适用	展示各部分占整体的比例。注意：饼图对于比较细微差异或展示过多类别时效果不佳，此时柱状图通常是更好的选择。
关键参数	x (数据), labels, explode (突出块), autopct (百分比显示), colors, startangle, shadow.
示例	`wedges, texts, autotexts = ax.pie(sizes, explode=explode, labels=labels, autopct='%1.1f%%', shadow=True, startangle=90, colors=['yellowgreen', 'gold', 'lightskyblue', 'lightcoral'])` `# autopct: 显示百分比格式; startangle: 起始角度; shadow: 阴影`

7.3 图表定制与多子图布局

7.3.1 图表定制

7.3.1.1 颜色 Colors

名称	`'blue', 'red', 'green', 'black', 'white', 'cyan', 'magenta', 'yellow'` `('b', 'r', 'g', 'k', 'w', 'c', 'm', 'y')`
十六进制	`'#FF5733', '#33FFA8'`
RGB	RGB/RGBA 元组：`(0.1, 0.2, 0.5)` 或 `(0.1, 0.2, 0.5, 0.7)` (含透明度)。
灰度	`'0.7'` (0=黑, 1=白)
cmap	不实用。用于将数值映射到颜色序列，常用于 scatter, imshow, contourf 等

7.3.1.2 线条样式 (linestyle) 与标记 (marker)

线条样式	`'-'` (实线), `'--'` (虚线), `':'` (点线), `'-.'` (点划线)。
标记样式	`'o'` (圆), `'s'` (方块), `'^'` (上三角), `'v'` (下三角), `'+', '*', '.'` (点), `'x'` 等
控制参数	markersize: 控制大小。markeredgecolor, markerfacecolor: 控制边框和填充色。

7.3.1.3 文本与注释 (ax.text, ax.annotate)

添加文本	`ax.text(x, y, "Some text")` 在指定坐标 (x, y - 数据坐标系) 处添加文本。
箭头注释	`ax.annotate("Annotation text", xy=(arrow_tip_x, arrow_tip_y), xytext=(text_x, text_y), arrowprops=dict(facecolor='black', shrink=0.05))` 添加带箭头的注释。xy 是箭头指向的点，xytext 是文本位置。

7.3.1.4 坐标轴范围与刻度

范围	`ax.set_xlim(min_val, max_val), ax.set_ylim(...)`
刻度位置	`ax.set_xticks([pos1, pos2, ...]), ax.set_yticks(...)`
刻度标签	`ax.set_xticklabels(['label1', 'label2', ...]), ax.set_yticklabels(...)`
对数刻度	`ax.set_xscale('log'), ax.set_yscale('log')`
刻度参数	`ax.tick_params(axis='both', direction='in', length=6, width=2, colors='r', labelsize=10)` 控制刻度线方向、长度、宽度、颜色，标签大小等

7.3.1.5 风格表 (plt.style)

风格表	Matplotlib 提供了一些预设的风格，可以快速改变图表的整体外观。 `plt.style.available` 可以查看可用的风格
示例	`plt.style.use('seaborn-v0_8-darkgrid')` `# 尝试 'ggplot', 'fivethirtyeight', 'seaborn-v0_8-whitegrid'`

7.3.2 多子图布局

7.3.2.1 plt.subplots

用途 创建规则的网格子图。

子图网格 创建 2 行 2 列的网格:

```
fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(10, 8), sharex=True, sharey=True)
```

单独设置

```
axes[0, 1].plot(x, x**2, 'g--')        axes[0, 1].set_title('Quadratic')
axes[1, 0].plot(x, np.log(x), 'b:')    axes[1, 0].set_title('Logarithmic')
axes[0, 0].set_ylabel('Y value')       axes[1, 0].set_ylabel('Y value')
```

调整布局

```
plt.tight_layout(rect=[0, 0, 1, 0.95]) # rect 调整总标题与子图间距
```

7.3.2.2 GridSpec

用途 允许创建非均匀的、跨越多行或多列的子图。

子图网格 2x3 的网格 `gs = gridspec.GridSpec(2, 3, figure=fig)`

子图设置

```
ax1 = fig.add_subplot(gs[0, :])   第一个子图占据第一行所有列
ax2 = fig.add_subplot(gs[1, 0])   第二行第一列
ax3 = fig.add_subplot(gs[1, 1:])  第二行，从第二列到最后一列
```

7.3.3 图表保存

用途 将绘制好的图表保存到文件中。

常见格式 **位图**：**PNG, JPG**，由像素组成，放大后会失真。适合网页展示、普通文档。

矢量图：**PDF, SVG, EPS**，由数学公式描述，无限放大不失真。适合出版物、需要编辑的场合。

注意

① 保存图表（应在 plt.show() 之前调用，或者在非交互式环境中使用）

② `format`：自动从文件名后缀推断，也可显式指定

③ `dpi`：分辨率（dots per inch），影响位图清晰度

④ `transparent`：背景是否透明（对 PNG 有效）

⑤ `bbox_inches='tight'`：尝试裁剪掉图表周围多余的空白

示例

```
fig.savefig('my_plot.png', dpi=300, transparent=False, bbox_inches='tight')
fig.savefig('my_plot.pdf', bbox_inches='tight') # 保存为矢量图 PDF
fig.savefig('my_plot.svg', bbox_inches='tight') # 保存为矢量图 SVG
```