第5章

数据可视化分析

【本章导读】

数据可视化是数据分析流程中最具表达力的环节。再精确的统计指标、再复杂的算法模型，最终都需要通过直观的图表与业务决策者进行沟通。数据可视化能够将数据背后隐藏的模式、趋势与异常转换为视觉信号，帮助分析结论跨越技术门槛，直接服务于商业判断与行动决策。

本章从可视化设计的基本逻辑出发，首先介绍 Matplotlib 的绘图底层机制与 rc 参数配置方法，建立对图形对象的整体认知，引导绘制既专业又符合个性化需求的图表；随后系统讲解折线图、柱状图、散点图等六类常用图表的适用场景与实现代码，每类图表均紧密围绕真实的商业分析需求展开绘制。内容不仅停留于代码实现，更关注“为何选用这张图”“如何读懂这张图”。

本章以“人力资源数据可视化分析”作为综合实践项目，完整呈现从数据读入、图表选择、多图组合到业务解读的全过程。通过该项目，将零散的绘图指令整合为完整的数据叙事，在图表布局、色彩搭配、信息层级等方面形成规范化的可视化表达习惯。

5.1 数据可视化分析简介

5.1.1 数据可视化分析的经典案例

数据可视化与数据可视化分析并非同一概念。数据可视化是将数据映射为视觉元素的技术过程，关注如何绘制图表、调用什么函数以及设置哪些参数；而数据可视化分析是以可视化作为认知工具，从数据中发现问题、验证假设并提炼结论的完整思维活动。前者回答“如何画”，后者回答“为什么画”以及“画完之后看到了什么”。

这一区别在经典案例中体现得尤为分明。以下两个跨越百余年的可视化实践，其技术手段在今天看来均非难事——一张点地图、一个动态仪表盘，任何具备基础绘图能力的人皆可复现。然而真正让它们载入史册的，从来不是“如何画出这些图表”，而是“从图表中看到了什么，并因此改变了什么”。

在互联网时代，数据可视化分析的力量体现在对复杂系统的实时洞察与全球协同上。

一个典范是基于自动识别系统(AIS)数据构建的全球船舶实时航行地图(图 5-1)。海事数据本身是公开的,但最初分散在成千上万个端口。当开发者将这些数据汇聚,并映射到一张可交互的全球地图上时,隐藏的世界经济脉搏便清晰浮现。从这张图中,任何人都能直观看到全球贸易的实时流动:主要航运枢纽的拥堵程度、重要海峡的通行密度、特定区域内的船只数量与类型,甚至能回溯单艘船的完整航程。大宗商品交易员据此预判供应链瓶颈与到货时间;环保组织通过监控船只航迹,分析排放并保护敏感海域;新闻机构则用它来揭示地缘政治事件对全球物流的即时影响,例如运河堵塞或主要港口作业变化如何像涟漪般扩散至全球。这张图本身并未创造新的数据,但其价值在于将浩如烟海的离散信息整合进一个统一的视觉框架,让全球物流这一庞大、动态且复杂的系统变得透明、可理解。它让决策者能够基于实时态势而非滞后报告作出响应,是数据可视化分析在全球化时代提升系统效率、优化资源配置与增强风险应对能力的杰出例证。

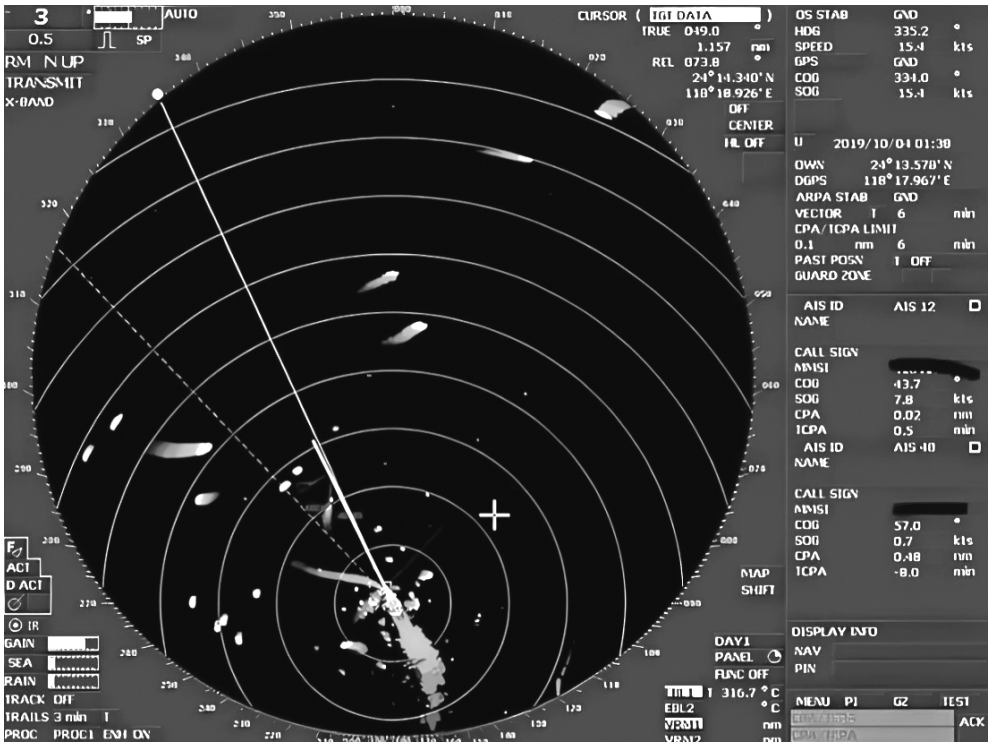

图 5-1　带 AIS(自动识别系统)的船舶雷达界面

5.1.2　数据可视化的分析用途

数据可视化并非图表的简单罗列,而是贯穿数据分析全流程的关键工具。从初步接触数据集,到最终交付分析结论,可视化在每个阶段承担着不同职能,服务于不同分析目标。按照数据分析的典型流程,可视化的主要分析作用可归纳为以下 4 类。

1. 数据理解

任何分析工作的起点都是理解数据本身。可视化是最直接的分析手段。例如,通过直方图观察字段分布是否偏态,通过柱状图对比各分类数值,通过散点图矩阵探查变量间的相

关关系。这些视觉线索能够帮助分析人员快速建立对数据集的整体认知，在识别数据质量问题时发现潜在规律，为后续的特征工程、变量筛选与模型选型提供客观依据。

2. 异常检测

在完成初步数据理解后，需要从数据中识别出偏离常规模式的样本或行为。可视化可以将少数异常值从海量正常数据中凸显出来。例如，箱线图中偏离上下限的数值、时序曲线上的突发峰值、多维空间中的孤立簇群、地理分布上的离群点位，都能通过恰当的图表快速定位。这一过程不仅服务于数据清洗，也为后续的精细化分析划定重点关注区域。

3. 模型诊断

当进入建模阶段，可视化将模型内部状态与预测结果外显化。例如，混淆矩阵能够直观呈现分类模型在各类别上的表现差异，残差图可以揭示回归模型的误差分布模式，特征重要性排序则帮助判断哪些变量真正贡献了预测能力。通过将模型的成功与失败转换为视觉信号，算法工程师得以定位系统性的偏差来源，进而优化特征、调整阈值或补充训练数据。

4. 结果展示

数据分析的最终产出往往需要交付给非技术背景的管理者或业务方。此时，可视化是将专业分析结论转换为业务语言的核心桥梁。例如，用一张清晰的柱状图或折线图，直观呈现客户流失风险、销售趋势变化或营销活动效果，帮助决策者快速理解分析结论、评估商业价值并制定后续策略。可视化让分析工作形成闭环，也让数据价值真正落地。

5.1.3 数据可视化图表类型

面对同样的数据，选用不同的图表类型，传递的信息与视角截然不同。可视化图表的核心任务是将数据映射为视觉元素——位置、长度、面积、角度、颜色、形状等，每种映射关系对应着特定的分析意图。根据数据关系与表达目标的差异，常用图表可归纳为五大类，如表 5-1 所示。

表 5-1 数据可视化常用图表的五大类型

图表类别	核心任务	典型图表	视觉映射要素
比较类	展示不同类别、组别或时间段的数值差异	条形图、柱状图、雷达图	长度、位置、面积
分布类	揭示数据整体形态、集中趋势与离散程度	直方图、箱线图、散点图、核密度曲线	位置、密度、间距
趋势类	呈现变量随时间或其他有序维度的变化轨迹	折线图、面积图	位置、方向、斜率
构成类	展示整体与部分的比例关系	饼图、环形图、堆叠柱状图	角度、长度、面积
关系类	揭示变量或个体之间的关联结构	散点图、气泡图、热力图、网络图	位置、颜色、连线

5.1.4 Python 数据可视化方法

Python 生态中拥有丰富的数据可视化工具库，不同库在设计理念、功能边界及适用场

景上各有侧重。根据任务目标选择合适的工具，能够让分析工作更加高效顺畅。Python 主流可视化工具及说明如表 5-2 所示。

表 5-2　Python 主流可视化工具库及说明

库名称	设计定位	核心优势	主要局限	典型适用场景
Matplotlib	底层绘图引擎	高度可定制，出版级输出，生态基石	API 复杂，默认样式朴素	学术论文、科研图表、高度定制化静态图表
Pandas.plot	数据处理集成	极简调用，无缝嵌入数据处理流程	功能基础，定制能力弱	快速数据探查、即时可视化
Seaborn	统计可视化	API 简洁，默认美观，擅长多变量关系	定制能力受限，静态图表	数据探索、统计建模、快速生成汇报用图
Pyecharts	国内交互可视化	图表华丽，交互丰富，中国地图支持	与 NumPy/Pandas 集成度稍弱	网页报告、商业展示、中国地图可视化
Plotly	国际交互可视化	强大交互性，Dash 仪表盘生态，3D 与地图	文件体积大，静态导出受限	交互式 Web 应用、可探索数据仪表盘

上述工具各有所长，但若论及可视化生态的根基，则非 Matplotlib 莫属。无论是 Pandas.plot 的便捷调用、Seaborn 的统计图表，还是 Pyecharts 与 Plotly 的交互渲染，底层均不同程度地依赖或借鉴 Matplotlib 的图形架构。Matplotlib 提供从画布、坐标系到每个视觉元素的完整控制接口，这种设计虽然增加了初期学习成本，但也换来了几乎不受限制的图表定制能力。本章将从 Matplotlib 的基本绘图流程入手，逐步展开常用图表类型的实践应用，最终通过完整项目串联从数据到图表的全流程绘制路径。

5.2　Matplotlib 基础绘图

5.2.1　Matplotlib 的安装

Matplotlib 是 Python 生态中最经典的数据可视化库。在使用之前，需要先将其安装到当前的 Python 环境中，常用的安装方式有以下两种。

1. 使用 pip 安装

pip 是 Python 的包管理工具。在命令行中输入以下命令即可完成安装。

```
pip install matplotlib
```

执行后，pip 会自动从 Python Package Index(PyPI)下载 Matplotlib 及其依赖项。安装过程中，pip 会提示安装进度信息。若命令执行完毕无报错，则表明安装成功。

2. 使用 Conda 安装

若使用 Anaconda 或 Miniconda 环境，可通过 Conda 进行安装。在 Anaconda Prompt(Windows)或终端(macOS/Linux)中输入以下命令。

```
conda install matplotlib
```

Conda 会自动处理依赖关系，并将 Matplotlib 安装到当前激活的 Conda 环境中。这种

方式便于在多项目环境中隔离不同版本的包依赖。

安装完成后，可通过以下脚本验证 Matplotlib 是否可正常使用。

```
import matplotlib.pyplot as plt
import numpy as np
x = np.linspace(0, 10, 100)              #创建并存储 100 个数据点
y = np.sin(x)                            #计算对应的正弦值
plt.plot(x, y)                           #绘制正弦曲线
plt.show()                               #显示图形
```

运行程序后，若能显示正弦曲线（如图 5-2 所示的效果图），说明 Matplotlib 已经成功安装并可供调用。

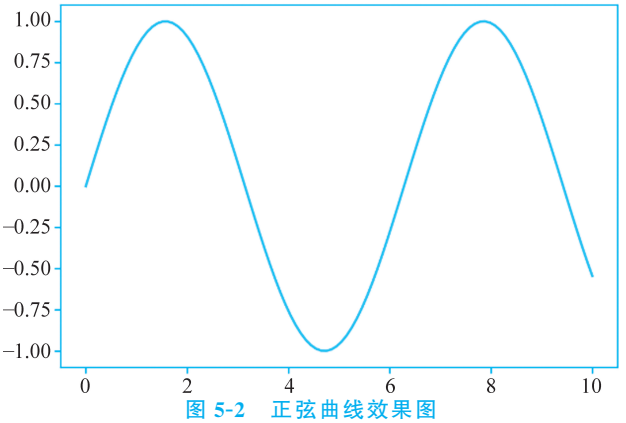

图 5-2　正弦曲线效果图

5.2.2　Matplotlib 绘图主要流程

Matplotlib 的 pyplot 模块提供了一套简洁的绘图接口，通过函数调用即可完成从创建画布到输出图形的全过程。Matplotlib 绘图流程如图 5-3 所示，有 3 个主要阶段：初始化画布与子图、配置图形元素与绘图、输出与展示图形。

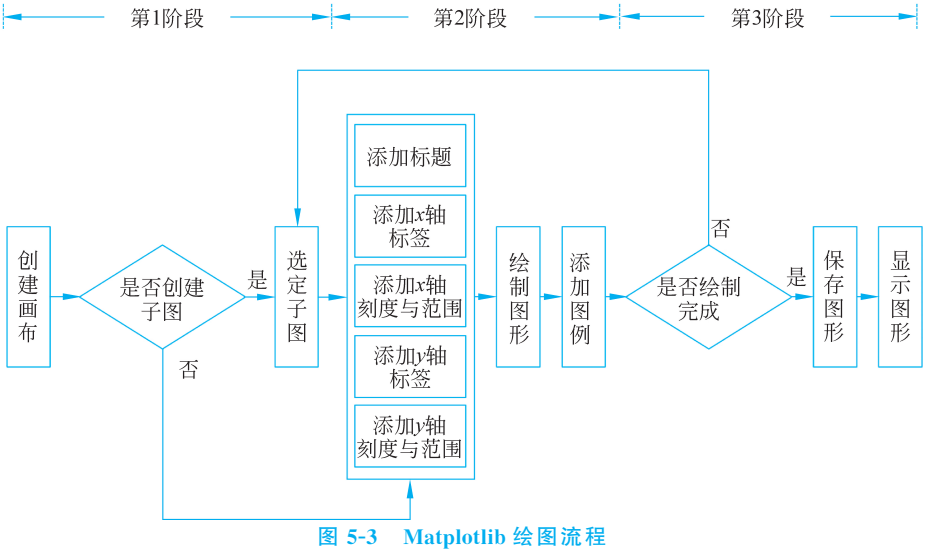

图 5-3　Matplotlib 绘图流程

1. 初始化画布与子图

创建画布与子图是绘图的起点。plt.figure()用于创建一个空白画布，可指定画布大小、分辨率等属性。若需要在画布上划分多个子图，可通过 Figure 对象的 add_subplot()方法添加子图，并指定子图的行数、列数及当前选中子图的位置，基本语法如下。

```
fig = plt.figure(figsize=(8, 6))                    #创建宽 8 英寸、高 6 英寸的画布
ax = fig.add_subplot(1, 2, 1)                       #划分为 1 行 2 列，选中第 1 个子图
```

更常用的方式是直接使用 plt.subplots()，该函数一次性完成画布创建与子图划分，并返回 Figure 对象和 Axes 对象(或 Axes 数组)，代码更为简洁，示例代码如下。

```
fig, ax = plt.subplots(1, 2, figsize=(8, 6))    #创建 1 行 2 列的子图布局
```

【例 5-1】　创建 1 行 2 列的图表。

```
#创建 1 行 2 列的图表
import matplotlib.pyplot as plt                     #导入库
#创建空白画布
fig = plt.figure(figsize=(12, 8))

#逐个添加子图
ax1 = fig.add_subplot(1, 2, 1)                      #1 行 2 列第 1 个位置
ax2 = fig.add_subplot(1, 2, 2)                      #1 行 2 列第 2 个位置

#在各个子图中绘图
ax1.plot([1,2,3], [1,2,3])                          #绘制折线图
ax1.set_title('fig 1')
ax2.scatter([1,2,3], [3,2,1])                       #绘制散点图
ax2.set_title('fig 2')
```

程序创建 1 行 2 列的图表，可视化效果如图 5-4 所示。

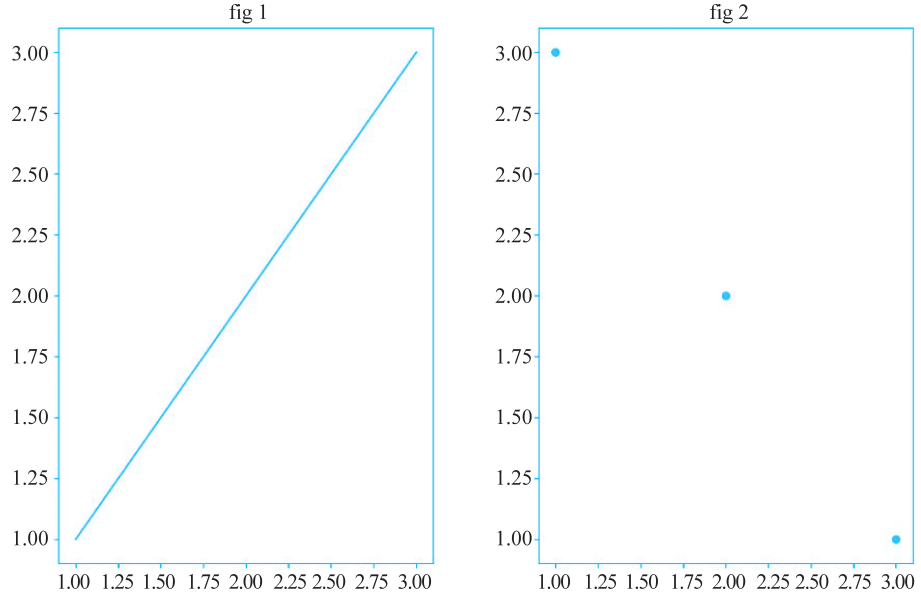

图 5-4　1 行 2 列的图表可视化效果图

2. 配置图形元素与绘图

图形元素配置与数据绘制并无严格的先后顺序，可在绘图前后灵活调整。常用配置包括标题、坐标轴标签、刻度范围和图例等。pyplot 常用图形配置函数及说明如表 5-3 所示。

表 5-3 pyplot 常用图形配置函数及说明

函　数	说　明
plt.title()	在当前图形中添加标题，可以指定标题的名称、位置、颜色及字体大小等参数
plt.xlabel()	在当前图形中添加 x 轴标签，可以指定位置、颜色及字体大小等参数
plt.ylabel()	在当前图形中添加 y 轴标签，可以指定位置、颜色及字体大小等参数
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()	获取或设置 x 轴的当前刻度位置和标签
plt.yticks()	获取或设置 y 轴的当前刻度位置和标签
plt.legend()	指定当前图形的图例，可以指定图例的大小、位置及标签

3. 输出与展示图形

图形绘制完成后，可通过以下两种方式输出。基本语法如下。

```
pylot.savefig()    #将图形保存为文件，支持 PNG、PDF、SVG 等格式，适用于报告或论文出版
pylot.show()       #将图形渲染至屏幕，适用于交互式探索场景
```

【例 5-2】 绘制 3 个函数的曲线图，并显示图例。

```
#基础绘图，显示图例
import numpy as np
import matplotlib.pyplot as plt

#准备数据
data = np.arange(0,1.5,0.01)          #产生[0,1.5]区间的数据，按固定步长 0.01 生成数组

#绘制图形
plt.plot(data,data**2)                                    #绘制曲线 y=x²
plt.plot(data,data)                                       #添加曲线 y=x
plt.plot(data,np.log(data+1))                             #添加曲线 y=log(x+1)

plt.legend(['y=x^2','y=x','y=log(x+1)'])                  #添加图例
pylot.savefig()                                           #保存图片
plt.show()                                                #显示图形
```

程序创建 3 个函数的曲线图，可视化效果如图 5-5 所示。

5.2.3 图形属性与 rc 参数配置

Matplotlib 允许对图形中的各类视觉元素进行精细调整，包括线条样式、标记形状、颜色映射及字体显示等。这些属性既可在绘图函数中作为参数临时指定，也可通过 rc 参数机制全局统一配置，避免重复代码。前者适用于单次个性化调整，后者适合建立统一的绘图风格规范。

1. 线条样式

线条样式包括线条的颜色、线型和线宽等，可通过 plot()函数的参数进行设置。常用线

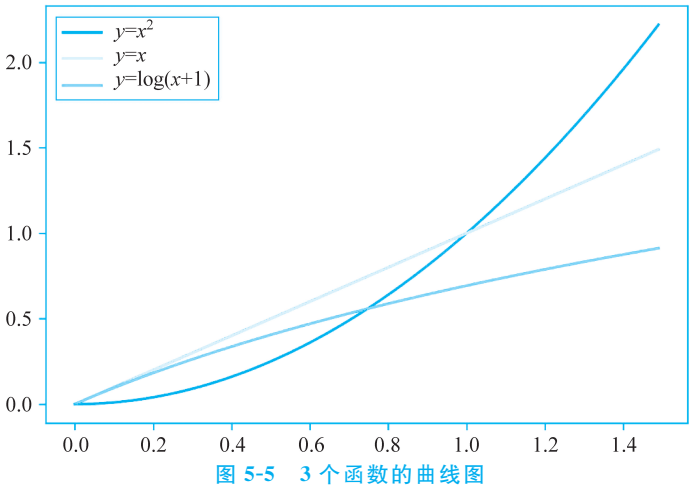

图 5-5　3 个函数的曲线图

条参数说明如表 5-4 所示。

表 5-4　常用线条参数说明

参　数	说　明
color	设置线条颜色。可选用'b'(蓝色)、'r'(红色)、'g'(绿色)、'k'(黑色)、'#FF0000'(十六进制表示的红色)等
linestyle	'-'(实线)、'--'(虚线)、'-.'(点画线)、':'(点线)
linewidth	设置线宽。可选用 1、2、3 等(单位为磅,数值越大线越粗)

配置线条样式的基本语法如下。

```
plt.plot(x, y1, color='r', linestyle='--', linewidth=2)     #红色虚线,线宽 2
plt.plot(x, y2, color='b', linestyle='-', linewidth=1.5)    #蓝色实线,线宽 1.5
```

2. 标记样式

为了更清晰地区分数据点,可给图形添加标记。可通过 plot()函数的参数进行标记样式设置,包括标记的形状、大小及颜色等。标记样式的常用参数说明如表 5-5 所示。

表 5-5　标记样式的常用参数说明

参　数	说　明
marker	设置标记形状。可选用'o'(圆形)、's'(正方形)、'^'(三角形)、'*'(星形)、'+'(加号)等
markersize	设置标记大小。可选用 5、10、15 等(数值越大标记越大)
markerfacecolor	设置填充颜色。可选用'b'(蓝色)、'r'(红色)、'g'(绿色)等
markeredgecolor	设置标记边缘颜色。可选用'k'(黑色)、'w'(白色)等

配置标记样式的基本语法如下。

```
#绘制数据点,设置标记为圆形,大小为 10,面颜色为绿色,边缘颜色为黑色
plt.plot(x, y, marker='o', markersize=10, markerfacecolor='g', markeredgecolor
='k')
```

3. 颜色设置

在 Matplotlib 中，线条颜色通过 color 参数进行设置，该库支持多种颜色指定方式。

（1）预定义的颜色名称（如'red'表示红色、'blue'表示蓝色）。

（2）缩写字母，如'r'（红）、'b'（绿）、'g'（蓝）、'k'（黑）、'w'（白）等。

（3）十六进制颜色码（格式为'#RRGGBB'，例如'#FF5733'表示橙色）。

（4）RGB 或 RGBA 元组数值定义颜色。其中 RGBA 格式的第 4 个参数表示透明度（Alpha 通道），采用 0～1 范围值，0 为完全透明，1 为完全不透明。

线条颜色配置方式如下。

```
plt.plot(x, y, color='#FF00FF')          #使用十六进制颜色码设置线条颜色
plt.plot(x, -y, color=(0, 1, 0, 0.5))    #使用 RGB 值设置线条颜色，透明度为 0.5
```

4. 字体与中文显示

图形中的标题、标签、刻度等包含文字元素。字体设置主要包括字体样式、大小及颜色等。在 Matplotlib 中，可通过 rcParams 字典来全局设置字体属性，也可以在单个函数中通过参数进行局部设置。字体设置常用参数说明如表 5-6 所示。

表 5-6　字体设置常用参数说明

参　数	说　明
font.family	设置字体家族。可选用'serif'（衬线字体）、'sans-serif'（无衬线字体）、'monospace'（等宽字体）等
font.size	设置字体大小。可选用 10、12、14 等
font.weight	设置字体粗细。可选用'normal'（正常）、'bold'（加粗）、'light'（细体）等
text.color	设置文字颜色。可选用'r'（红色）、'k'（黑色）等

Matplotlib 默认字体不支持中文，需要手动指定中文字体。同时，设置中文字体后，坐标轴上的负号（如－10、－5）可能显示为方块，需通过 axes.unicode_minus 参数调整负号显示方式。示例代码如下。

```
plt.rcParams['font.sans-serif'] = 'SimHei'   #指定中文字体（黑体），防止中文乱码
plt.rcParams['axes.unicode_minus'] = False   #使用普通连字符显示负号，防止负号异常
```

【例 5-3】　绘制某电商平台一周销售额趋势分析折线图，并设置字体参数。

```
import matplotlib.pyplot as plt
import numpy as np
#设置中文字体
plt.rcParams['font.sans-serif'] = ['SimHei']  #使用黑体
plt.rcParams['axes.unicode_minus'] = False    #正常显示负号

#一周的销售额（万元）
days = ['周一', '周二', '周三', '周四', '周五', '周六', '周日']
sales = [25.3, 28.7, 32.5, 30.1, 45.8, 62.3, 55.6]
plt.figure(dpi=100)                           #创建画布
#绘制折线图
plt.plot(days,                                #x 轴数据，表示一周的日期
         sales,                               #y 轴数据，表示对应的日销售额/万元
         marker='o',                          #设置数据点的标记形状为圆形
```

```
        linewidth=2,                              #设置折线宽度为 2 磅
        markersize=8,                             #设置数据点标记的大小为 8 磅
        markerfacecolor='red',                    #设置数据点标记的填充颜色为红色
        color='#2E86C1',          #设置折线颜色为十六进制颜色码#2E86C1(一种蓝色)
        label='日销售额'          #设置图例标签为"日销售额",用于在图表图例中显示
       )

#设置图表标题和标签
plt.title('某电商平台一周销售额趋势分析')
plt.xlabel('日期')
plt.ylabel('销售额 (万元)')
plt.legend()                              #添加图例
plt.show()                                #显示图表
```

某电商平台一周销售额趋势分析折线图显示效果如图 5-6 所示。

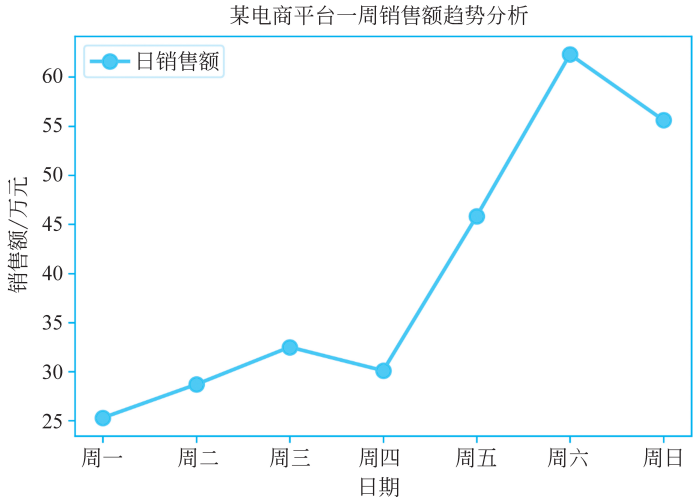

图 5-6　某电商平台一周销售额趋势分析折线图显示效果

5.3　Matplotlib 常用图表绘制

图表是数据可视化最直接的呈现形式。Matplotlib 支持绘制数十种图表类型，其中折线图、柱状图、散点图、饼图、箱线图和热力图是数据分析中非常常用、覆盖场景广泛的 6 类基础图表。它们分别对应着趋势比较、类别比较、变量关系、占比构成、分布概括和矩阵密度等核心分析任务。然而，同一组数据可以绘制出多种不同的图表，真正决定选用哪种的，并非图表本身的功能多寡，而是待回答的业务问题与分析目的。图表类型的选择，本质上是数据思维与业务理解共同作用的结果。

本章围绕某百货商场近 4 年约 189 万条真实会员消费记录展开，部分数据样例如图 5-7 所示。通过该数据集，逐一演示每类图表的具体实现代码、参数配置逻辑，并结合业务背景对输出图形进行解读，帮助建立从原始数据到业务洞察的完整可视化分析链条。

5.3.1　折线图

折线图通过将数据点按顺序连接，展现数值随某一连续维度（通常是时间）变化的趋势。