第3章

数据可视化

CHAPTER 3

为了更好地满足金融领域中对数据可视化的需求，本章介绍了如何绘制多种常见数据可视化图形，如折线图、散点图、属性两两相关图、小提琴图等，并对每种图形的功能及适用情形进行了总结。

本章旨在通过简单易懂的语言阐述各种图的功能，并介绍图形的绘制。除此之外，本章还以多种经济金融市场数据为切入点，以实例进行代码的讲解，力求提高代码的适用性。在表述中，定义一张存放数据的表格为一个信息系统，在表格中，一般而言，列标签为特征，也可以称为属性。列标签又可以分为条件属性和决策属性。例如，在表格“上市公司风险预警样本”的 Sheet3（注意，不是工作表 Sheet1）工作表中，第一行是列标签，其中，财务指标是条件属性，最后一列标识上市公司是正常还是 ST 的'正常'，是决策属性。本章主要内容结构如图 3.1 所示。

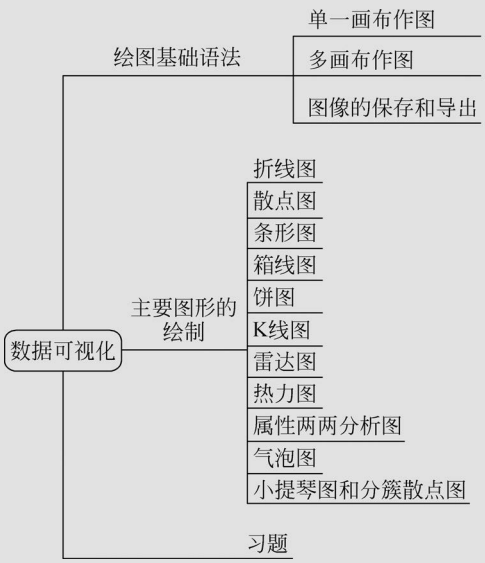

图 3.1　本章主要内容结构

3.1 绘图基础语法

数据可视化是将数据转换为图像形式,用一种直观的方式呈现出数据的分布特征。这不仅有助于人们观察数据的分布和变化,还可以发现隐藏在数据中的规律和问题。然而,海量的数据使得手动绘制数据图表不切实际。而编程手段的发展为数据可视化提供了多种多样的工具。

Python 是数据可视化工具之一,它提供了多种库来进行数据可视化的操作。其中最常用的是 Matplotlib 库和 Seaborn 库。这两个库拥有丰富的绘图函数,能够帮助用户创建各种类型的图形。在正式绘图之前,先来了解一下其中的一些基础函数。

3.1.1 单一画布作图

```
#coding = utf8                                          #支持中文编码
import matplotlib.pyplot as plt
import pandas as pd
plt.rcParams['font.sans - serif'] = ['SimHei']          #设置中文显示的字体是 SimHei
plt.rcParams['axes.unicode_minus'] = False              #坐标轴正常显示负号
#设置 x 轴变量将 2012 - 12 - 31 日设置为起点,按年份产生 10 个数据
x = pd.date_range(start = '2012 - 12 - 31',freq = 'YE',periods = 10)
data = pd.DataFrame(data = [16,13,10,12,17,14,12,8,10,6],index = x ,columns = ['成本'])
#设置画布大小、分辨率、背景色
plt.figure(figsize = (5,4),dpi = 300,facecolor = 'lightgrey')
plt.xlabel('日期')                                      #设置 x 轴标题
plt.ylabel('成本')                                      #设置 y 轴标题
plt.grid()                                              #设置网格
y = data['成本']                                        #设置 y 轴变量
plt.plot(x,y,color = 'r',linestyle = '- .')             #颜色是'r'(红色),线性是'- .'(点画线)
plt.title('历年成本',fontdict = {'fontsize': 16, 'color': 'blue'},loc = 'left')
plt.show()                                              #绘制图像(见图 3.2)
```

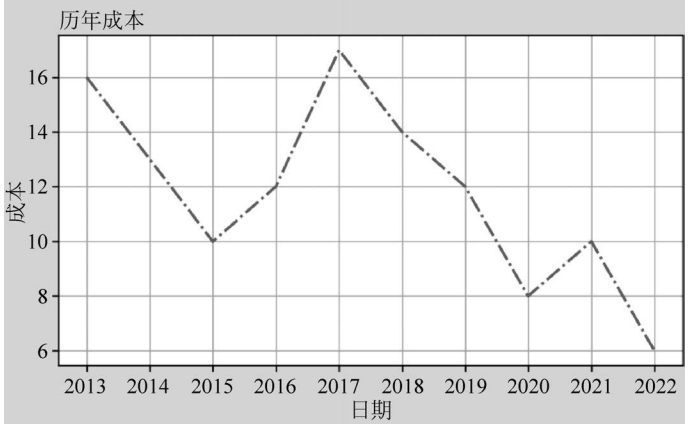

图 3.2 单一画布作图

注意:在利用 matplotlib 绘图时,PyCharm 社区版的部分版本会报错,错误提示是 This probably means that Tcl wasn't installed properly 或 This probably means that Tk wasn't installed properly。处理方式是打开 Python 根目录下的文件夹 tcl,将其中的 tk 包和 tcl

包的文件夹复制到根目录下的文件夹 Lib 中。

上述程序中所涉及的重要函数介绍如下。

1. pandas.date_range(start = None,end = None,periods = None,freq = 'D',tz = None)

该函数用来生成时间数据,参数设置如下。

- start：日期的起点,要求为字符串形式或时间格式,默认值是 None。
- end：日期的终点,也要求为字符串形式或时间格式,默认值是 None。
- periods：要生成的日期索引值的个数。当 periods 未指定时,start 和 end 不能为空。
- freq：计时单位,默认值是'D',表示以自然日为单位。
- tz：时区,如'Asia/Hong_Kong'。

2. figure(num = None,figsize = None,dpi = None,facecolor = None,edgecolor = None,frameon = True)

该函数用于创建新的图形,参数设置如下。

- num：图像编号或名称。
- figsize：figure 的宽和高。
- dpi：分辨率。
- facecolor：背景颜色。
- edgecolor：边框颜色。
- frameon：是否显示边框。

3. plt.plot(x,y,color = None,linestyle = None,marker = None)

该函数用于绘制线图,参数设置如下。

- x：x 轴数据的列表或数组。
- y：y 轴数据的列表或数组。
- color：线条的颜色,简写为 c,可选参数包括'b'(蓝色)、'g'(绿色)、'r'(红色)、'c'(蓝绿色)、'm'(洋红色)、'y'(黄色)、'k'(黑色)、'w'(白色)等。
- linestyle：线型,简写为 ls,取 None,即默认取实线。可选参数包括'-'(实线)、'--'(虚线)、'-. '(点画线)、':'(点线)等。
- marker：点型,取 None,即默认不显示数据点的标记。可选参数包括'.'(点标记)、','(像素标记)、'o'(实心圆标记)、'>'(右三角标记)、'<'(左三角标记)、'*'(星形标记)、's'(实心正方形标记)、'p'(实心五角形标记)等。

4. plt.title(label,fontdict = None,loc = None,pad = None,y = None, ** kwargs)

该函数用于在绘制的图表上添加标题,参数设置如下。

- label：标题,可以为字符串类型或者数学表达式。
- fontdict：参数是一个字典,包括'family'、'size'、'color'和'weight'4 个键。'family'键用于设置字体族；'size'键用于设置字体大小；'color'键用于设置字体颜色；'weight'键用于设置字体粗细。

- loc：标题的位置，可选参数包括'center'(居中)、'left'(靠左)、'right'(靠右)。
- pad：标题与图表的距离。
- ** kwargs：表示允许传入任意数量的关键字参数(包括 0 个参数，即无参数)。

注意：plt.show()用于展示图形，若没有该代码，则图形不会直接显示。

3.1.2 多画布作图

1. subplot(nrows,ncols,index, ** kwargs)

subplot()函数的参数设置如表 3.1 所示。

表 3.1 subplot()函数的参数及功能

参 数	功 能
nrows	在画布纵轴上分隔出几行
ncols	在画布横轴上分隔出几列
index	子图索引
** kwargs	一些涉及子图属性的关键字参数

多画布作图示例如下。

```
import numpy as np
import matplotlib.pyplot as plt
x = np.linspace(1, 10, 100) #产生 100 个数,这些数在 1～10 中均匀分布
plt.subplot(2, 2, 1)        #将画布分为 2 行 2 列,在位置 1 作图
plt.plot(x, np.sin(x), color = 'y',linestyle = ':')   #x 轴的数据是 x,y 轴的数据是 np.sin
                                                      #(x),颜色是'y'(黄色),线型是':'(虚线)
plt.subplot(2, 2, 2)        #将画布分为 2 行 2 列,在位置 2 作图
plt.plot(x,3 * x)
plt.subplot(2, 2, 3)        #将画布分为 2 行 2 列,在位置 3 作图
plt.plot(x, np.cos(x))
plt.subplot(2, 2, 4)        #将画布分为 2 行 2 列,在位置 4 作图
plt.plot(x, np.tan(x))
plt.show()                  #绘制图像(见图 3.3)
```

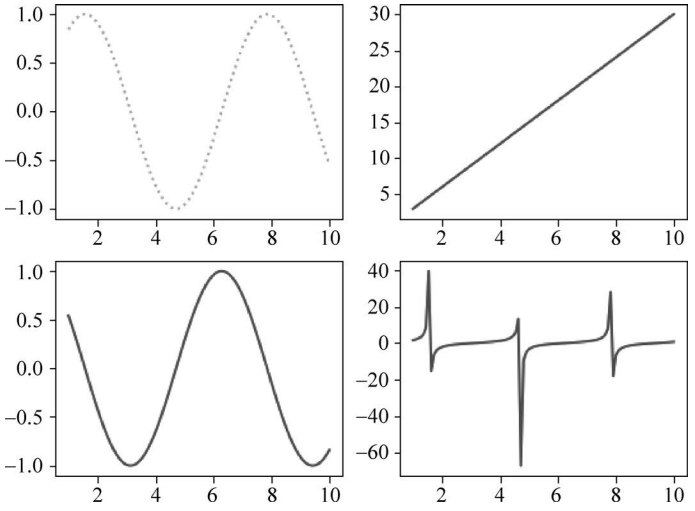

图 3.3 多画布

2. subplots(nrows = 1, ncols = 1, sharex = False, sharey = False)

subplots()函数的参数及功能如表 3.2 所示。

表 3.2　subplots()函数的参数及功能

参　数	功　能
nrows, ncols	将画布分割后的行数和列数，输入的是整数类型，默认是 1
sharex, sharey	是否共轴，可选布尔值、'none'、'all'、'row'、'col'。选择'True'或者'all'时，所有子图共享 x 轴或者 y 轴；选择'False'或者'none'时，所有子图的 x、y 轴各自独立；选择'row'时，每一行的子图会共享 x 或者 y 轴；选择'col'时，每一列的子图会共享 x 或者 y 轴

```
fig, ax = plt.subplots(nrows = 2, ncols = 2,sharey = True)
          #将画布分为 2 行 2 列，创建子图对象，子图共享 y 轴。fig 代表整个图形，ax 代表子图
ax[0][0].plot(x, np.sin(x))
ax[0][1].plot(x, 3 * x)
ax[1][0].plot(x, np.cos(x))
ax[1][1].plot(x, np.tan(x))
plt.show()                                          #绘制图像(见图 3.4)
```

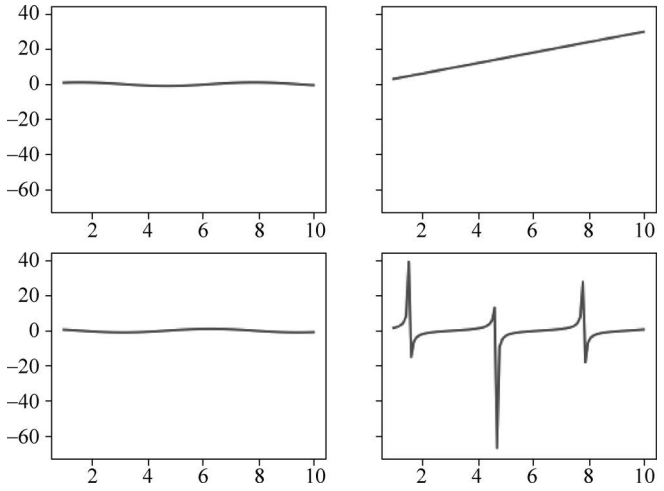

图 3.4　共享轴

对比图 3.3 和图 3.4 可以发现，正弦和余弦的 y 值在[−1,1]上分布，因为图 3.4 的子图共享 y 轴，公共的 y 轴的范围是[−60,40]，显示出的正弦图和余弦图类似直线。

3. subplot2grid(shape, loc, rowspan = 1, colspan = 1, fig = None)

subplot2grid()函数的参数及功能如表 3.3 所示。

表 3.3　subplot2grid()函数的参数及功能

参　数	功　能
shape	表示画布的网格形状，若为(2,2)，则将画布分为 2 行 2 列
loc	表示当前选择的绘图区，若为(0,0)，则图片从第 1 行第 1 列开始展示
rowspan	表示向下跨越的行数，默认为 1
colspan	表示向右跨越的列数，默认为 1
fig	表示放置子图的画布，默认为当前画布

```
#将图片分为3行3列,从第1行第1列开始显示,图像跨3列1行。结合图3.5可以直观理解参数
#数值选择的含义
plt.subplot2grid((3, 3), (0, 0), colspan = 3, rowspan = 1)
plt.plot(x, np.sin(x))
#将图像分为3行3列,从第2行第1列开始显示,图像跨2列1行
plt.subplot2grid((3, 3), (1, 0), colspan = 2, rowspan = 1)
plt.plot(x, np.cos(x))
#将图像分为3行3列,从第2行第3列开始显示,函数跨1列2行
plt.subplot2grid((3, 3), (1, 2), colspan = 1, rowspan = 2)
plt.plot(x, 3 * x)
#将图像分为3行3列,从第3行第1列开始显示,图像跨1列1行
plt.subplot2grid((3, 3), (2, 0))
plt.plot(x, np.tan(x))
#将图像分为3行3列,从第3行第2列开始显示,图像跨1列1行
plt.subplot2grid((3, 3), (2, 1))
plt.plot(x, 5 * x)
plt.show() #绘制图像(见图3.5)
```

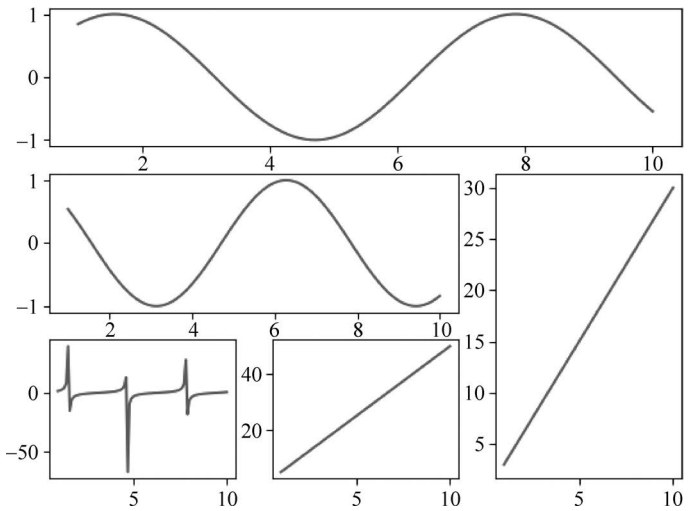

图 3.5 不规则画布

subplot()函数、subplots()函数以及 subplot2grid()函数均可用于绘制多图,但它们在使用方式和功能上有一些区别。subplot()与 subplots()这两个函数都是先划分网格然后作图。subplot()采用逐渐分割的方式逐个网格地作图; subplots()也是逐个网格地作图,即使某个子图上没有作图内容,出图时该子图网格也会存在并显示出来。而 sublot2grid()既可以规则划分网格作图,也可以通过设置参数达到不规则划分画布作图,在图像展出格式上更加自由,能够适应各种复杂的布局需求。

3.1.3 图像的保存和导出

Matplotlib 允许将图形保存为多种格式,其中包括 PNG、PDF 和 SVG。例如,通过 plt.savefig(r'C:\\量化金融\\picture',format='png',dpi=300)就可以将绘制的图形保存为分辨率为 300 的名为 picture 的 PNG 格式文件。

注意:保存这一步骤(savefig())要写在图片展示(show())前面,否则保存的就是空白

图像，若没有设定绝对路径，则可直接通过 plt.savefig('picture',format='png',dpi=300)保存图像，图像与当前正在编写代码的 Python 脚本处在同一文件夹中。

3.2　主要图形的绘制

财务报表是投资决策的考量因素，也是投资收益的反映。作为财务报表之一的利润表是反映公司在某一特定日期经营成果的报表，包括营业收入、营业成本、费用和利润等方面。这些数据直接反映了公司的财富增长规模和经营能力。通过分析财务报表，可以评估公司的经营状况，衡量企业绩效，预测公司前景，从而为公司制定更好的战略和决策提供参考，帮助投资者做出更明智的投资决策。作为数据分析的重要手段之一，财务数据可视化可以直观展示公司的营利能力、偿债能力、营运能力和成长能力。下面以 2012—2022 年格力电器利润表数据(数据来源：Wind)为例进行演示。

3.2.1　折线图

折线图以折线的形式显示数据随时间或其他有序变量变化的趋势，适用于分析数据的趋势。

1. 折线图的功能

(1) 展示数据趋势：可用于呈现数据随时间或其他有序变量变动而变化的趋势。通过连接数据点形成的线条，可以直观地呈现数据的递增或递减趋势以及增减的速率和规律。

(2) 对比分析：可用于比较多组数据随时间或其他有序变量变动而变化规律。通过在同一张图中绘制多条折线，可以直观地比较不同组数据之间的差异。

(3) 识别极值：可用于识别数据集中的峰值和谷值，这对于分析数据的极值、转折点(局部极值)等趋势发生反转的点非常有帮助。

2. 绘制折线图

选取格力电器的净利润与营业利润绘制折线图。

1) 读取表格

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
plt.rcParams['font.sans-serif'] = ['SimHei']    #设置中文显示
plt.rcParams['axes.unicode_minus'] = False      #坐标轴显示负号
path = 'C:\\量化金融\\第3章\\第3章数据'
data = pd.read_excel(path + '\\' + '格力电器年利润表.xlsx', sheet_name=0,index_col=0,
usecols='A:L')
#因为这种数据表格可能隐含着格式，如果没有添加参数 usecols='A:L'来限定读取表格的范围，则
#读取 data 后，可能会出现很多 NaN 和 Unnamed。虽然可以通过 filtered_yyprofit = [x for x in yyprofit
#if not math.isnan(x)]来清洗 NaN，但是为了简便，所以限定读表范围
```

2) 数据转换

```
profit = data.loc['利润总额']          #选取利润总额
yyprofit = data.loc['营业利润']        #选取营业利润
year = data.columns.values.tolist()    #将 data 的列标签(年份)的值转换为列表
#data.columns 是 data 的列标签;data.columns.values 是列标签的值
```

3) 绘图

```
plt.plot(year, profit,label = '利润总额',marker = 'o',markersize = 3,linestyle = ':')
                #以 year 为 x 轴,profit 为 y 轴绘图。点型 marker 为实心圆,线型 linestyle 为点线
plt.plot(year, yyprofit,label = '营业利润',marker = 'o',markersize = 3)
plt.title('折线图') #设置图名
```

4) 设置数据标签位置及大小

```
#为 year 所对应的 profit 设置数据标签,标签采用水平方向中心对齐、竖直方向底部对齐方式
for a, b in zip(year,profit ):
    plt.text(a, b, b, ha = 'center', va = 'bottom', fontsize = 10)
#通过 zip 将 year 和 profit 捆绑,在注释函数 text(x, y, string,ha,va)中,(x,y)是数据标签的坐
#标,在此处 x 是 a,y 是 b,其中,a = year,b = profit,即数据标签的坐标实际上是(year, profit)。
#显示的注释内容 string 是 b,b = profit 为 year 对应的 yyprofit 设置数据标签
for a, b in zip(year,yyprofit ):
    plt.text(a, b, b, ha = 'center', va = 'bottom', fontsize = 10)
plt.show( ) #运行结果显示见图 3.6
```

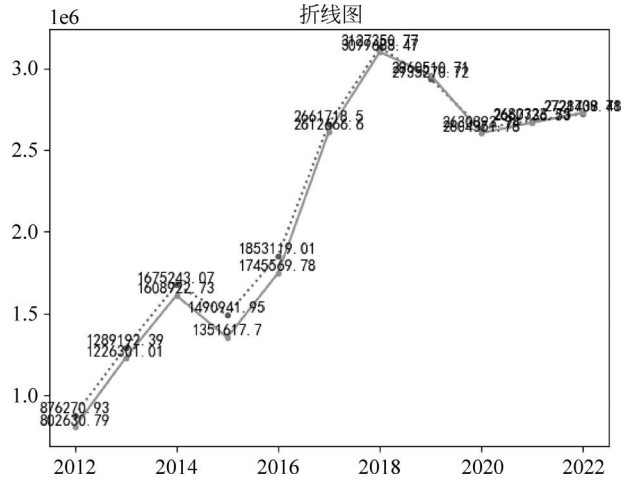

图 3.6 双折线图

根据图 3.6 可知,格力电器的利润总额和营业利润总额在 2018 年达到了顶峰,在 2015 年、2019 年和 2020 年均出现了短暂的下降。这可能是市场竞争加剧、成本上升或宏观经济环境的不利影响等因素导致的。尽管出现了个别年份的下滑,但总体来看,格力电器在 2012—2022 年间实现了利润的增长,这说明公司具有较强的市场适应能力和业务恢复能力。此外,值得注意的是,折线图显示格力电器每年的利润总额均大于营业利润总额,这表明除了主营业务带来的利润外,格力电器还通过其他业务如投资等获取了额外的利润。

3. 重要函数介绍

上述程序中所涉及的重要函数介绍如下。

1) zip(* iterables)

* iterables 表示可变数量的可迭代对象参数,可以是列表、元组等。zip()函数是从参数中依次取一个元素,返回一个元组。

```
list1 = [1,2,3]
list2 = [4,5,6]
for a, b in zip(list1,list2):
    print('a',a)
    print('b',b)
    print([x for x in zip(list1, list2)])
```

运行结果如下。

```
a 1
b 4
[(1, 4), (2, 5), (3, 6)]
a 2
b 5
[(1, 4), (2, 5), (3, 6)]
a 3
b 6
[(1, 4), (2, 5), (3, 6)]
```

从上述结果可以发现,zip()函数依次读取 list1 和 list2 中的元素,每次各读取一个,然后把这两个元素捆绑成一个元组并返回。

2) plt.text(x,y,string,ha=None,va=None,color=None)

函数 plt.text()的作用在于注释,该函数的参数设置如下。

- x: 注释内容的横坐标。
- y: 注释内容的纵坐标。
- string: 注释的内容。
- ha: 绘图的点在注释内容的水平位置,可选参数包括'right'、'center'、'left'。
- va: 绘图的点在注释内容的竖直位置,可选参数包括'top'、'bottom'、'center'、'baseline'。
- color: 注释内容的颜色。

3.2.2　散点图

散点图是数据点在直角坐标系上的分布图,它表示因变量随自变量而变化的趋势,也可以用来分析一组数据的变化趋势。

1. 散点图的功能

回归分析中往往用散点图来检验数据是否具有相关性,为选择合适的拟合模型提供依据。

(1) 展示数据关系: 可以展示两个或多个变量之间的关系。通过将数据点绘制在直角

坐标系中,可以直观地看出变量之间是否存在某种关系,如正相关、负相关或无关关系。

(2) 识别数据趋势:可以揭示数据的大致趋势。通过观察散点图数据点的分布和趋势线,可以发现变量之间的潜在关系,并据此进行数据分析和预测。

(3) 探测异常值:散点图有助于发现数据集中的异常值。

2. 绘制散点图

以 2012—2022 年格力电器的净利润为例,绘制散点图。

```
plt.scatter(year,profit)        #绘制散点图
plt.xlabel('年份')              #设置 x 轴标签
plt.ylabel('净利润')            #设置 y 轴标签
plt.title('散点图')             #设置图名
plt.show()                      #运行结果见图 3.7
```

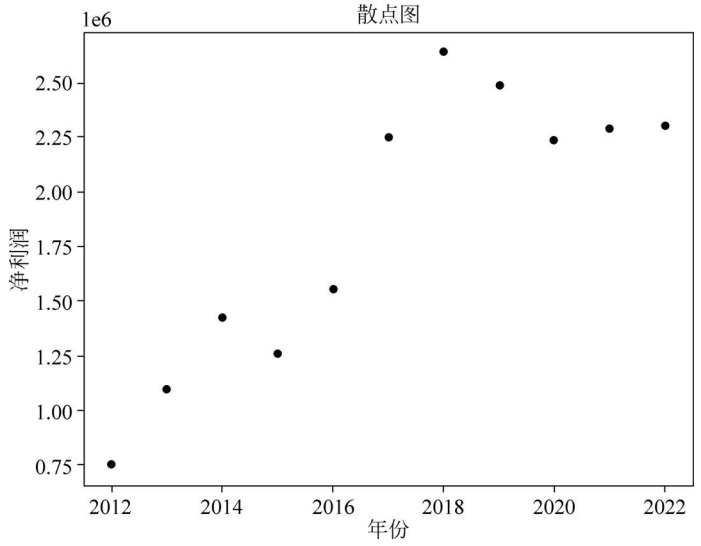

图 3.7 散点图

通过散点图,也能得出上述格力电器净利润的变化趋势。

3. 重要函数介绍

1) plt.scatter(x,y,s=None,c=None,marker=None,alpha=None)

该函数用于绘制二维图,主要参数如下。

- x,y: x,y 轴数据。
- s: 散点大小。
- c: 散点颜色,默认是蓝色'b',可以是字符串表示的颜色名称或表示颜色的序列。
- marker: 标记的样式,默认是'o'。
- alpha: 点的透明度,默认是 1,取值范围为 0～1。

2) pl.scatter3d(x,y,z,c=None,depthshade=True,marker=None,alpha=None,alpha=None)

该函数用于绘制三维图,主要参数如下。

- x,y：x,y 轴数据。
- s：散点大小。
- c：散点颜色,默认是蓝色'b',可以是字符串表示的颜色名称或表示颜色的序列。
- depthshade：默认是 True,点的大小和颜色会根据其距离视点的远近而变化。如果取 False,则所有点的大小和颜色相同。
- marker：标记的样式,默认是'o'。
- alpha：点的透明度,默认是 1,取值范围为 0～1。

3.2.3　条形图

条形图是用宽度相同的条形的高度或长短来表示离散变量的各个分组数据多少的图形。条形图分为垂直条形图和水平条形图,通常适用于分类数量相对较少的情况。

1. 条形图的功能

条形图常用于数量统计和频率统计,除此之外,也可以对不同类别的数据进行比较。

(1) 数据对比：可以直观地比较不同变量之间的数据大小,从而快速识别出各变量之间的差异和联系。

(2) 数据排序：可以根据数据大小进行排序,使得数据对比更加明确和直观。

(3) 展示数据分布：条形图还可以展示数据的分布情况。通过条形图的长度和宽度,可以大致了解数据的集中程度和离散程度。

选择格力电器的净利润绘制图表。

2. 绘制垂直条形图

```
plt.bar(year,profit)
plt.xlabel('年份')       #设置 x 轴标签
plt.ylabel('净利润')     #设置 y 轴标签
plt.legend(['净利润'])   #添加图例
plt.title('垂直条形图') #设置图名
plt.show( )              #如图 3.8 所示
```

3. 绘制水平条形图

```
plt.barh(year,profit)    #绘制水平条形图
plt.xlabel('净利润 ')    #设置 x 轴标签
plt.ylabel('年份')       #设置 y 轴标签
plt.legend(['净利润'])   #添加图例
plt.title('水平条形图') #设置图名
plt.show( )              #如图 3.9 所示
```

在本例中,通过条形图展示格力电器 2012—2022 年间的净利润额,可以清晰地看出每一年的净利润情况。由于条形图放大了时间点的长度,相较于散点图和折线图,它更加突出了每个时间点上的数据值。这使得观察者能够更容易比较不同年份之间的净利润差异大小,以及识别出净利润的峰值和变化趋势。

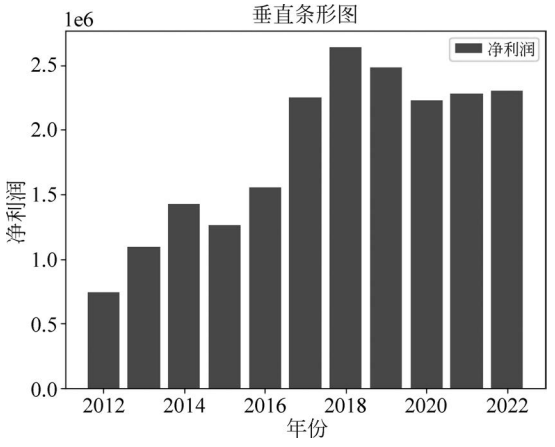

图 3.8　垂直条形图

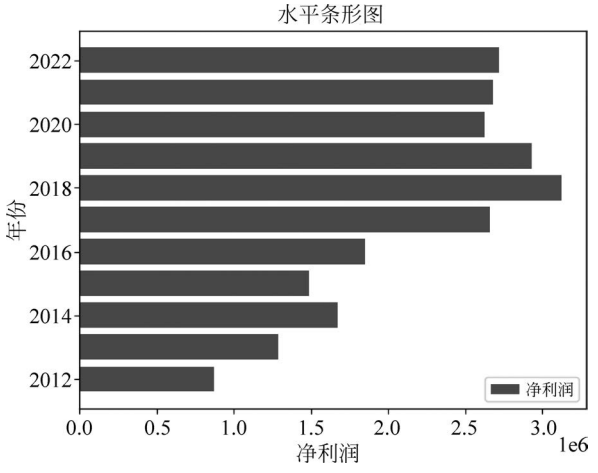

图 3.9　水平条形图

从条形图中可以看出,格力电器在 2018 年的净利润额达到了近十年的最大值,这体现了该年度公司在经营上的良好表现。同时,通过比较不同年份的条形高度,也可以看出在 2012—2022 年这一时间段内,格力电器的整体净利润额是在增长的。

3.2.4　箱线图

箱线图主要用于反映一组或多组连续型数据的中心位置和分布范围。箱线图是用来表示一组数据的分布,它由 6 个数值点组成:异常值(outlier)、中位数(median,即第 50%分位数)、最小值(min)、最大值(max)、上四分位数(即第 75%分位数)、下四分位数(即第 25%分位数)。其显示格式如图 3.10 所示。

1. 箱线图的功能

箱线图能够直观地展示数据的整体分布情况,显示数据中心趋势和离散程度。

(1) 观察数据的总体状态:箱线图可以展示数据的分位数、中位数和平均值等统计信息,从而帮助用户了解数据的整体分布和中心位置。

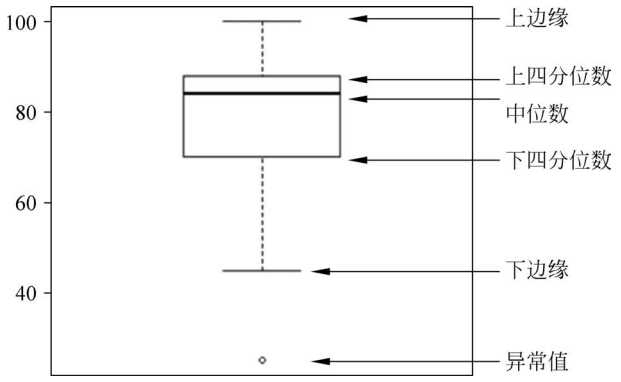

图 3.10　箱线图

(2) 识别异常值：箱线图通过设定内限(即箱子的上下边缘)和外限(通常是内限的 1.25 倍)来识别异常值。超过内限的数据被视为异常值，其中，在内限和外限之间的数据称为温和异常值，而在外限之外的数据被称为极端异常值。

(3) 了解数据的离散程度：箱线图的宽度(即箱子的长度)可以反映数据的离散程度。宽度越大，说明数据分布越分散；宽度越小，则数据分布越集中。

在金融领域中，箱线图可用于分析股票、债券等金融产品的价格波动情况，帮助投资者判断市场的整体趋势和风险水平。

2. 绘制箱线图

```
label = '净利润','营业利润'                                  # 多个数据。没有括号，视 label 为元组
plt.boxplot([profit,yyprofit],tick_labels = label)  # labels 是每组数据的标签
plt.show()                                                  # 如图 3.11 所示
```

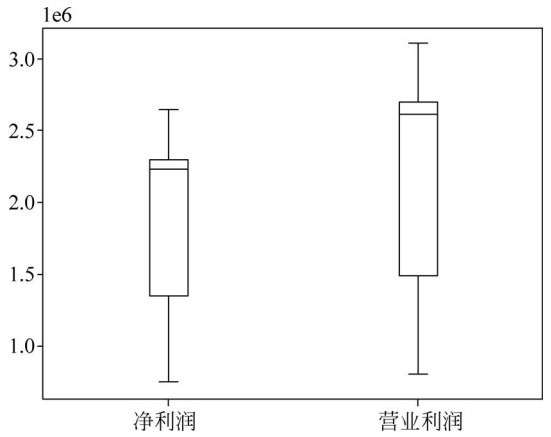

图 3.11　双箱线图

3.2.5　饼图

饼图将一个圆形区域划分为多个扇形，通过每个扇形的角度大小来表示相应类别的数据占总体的比例。

1. 饼图的功能

(1) 类别占比展示：用于展示不同类别或组别的相对比例关系，以及各个类别在总体中的占比情况。

(2) 突出要点：用于突出某个特定类别或组别在总体中的重要性或显著性。

2. 绘制饼图

在实际应用中，饼图广泛应用于多个领域，如销售数据分析、群体构成分析、预算分配、投资组合分析以及用户满意度调查等。

接下来以比亚迪公司在 2023 年 9 月 30 日的部分资产负债表数据(见表 3.4)为例绘制饼图(数据来源：东方财富网)。

表 3.4 比亚迪公司 2023 年 9 月 30 日资产负债表(部分)

资 产 类 别	金额/亿元
总资产	6233.00
流动资产	2751.00
货币资金	557.60
应收账款	529.70
存货	927.10
预付账款	32.17
非流动资产	3482.00
固定资产	2082.00
无形资产	316.50
长期待摊费用	9.18
商誉	6591.00
总负债金额	4822.00
流动负债	4193.00
非流动负债	629.40

```
money = [557.6, 529.7, 927.1, 32.17]#设置内容列表
subject = ['货币资金', '应收账款', '存货', '预付账款']  #设置对应内容名称
cols = ['c', 'm', 'r', 'b']            #设置对应颜色缩写
#绘制饼图
plt.pie(
    money,
    labels = subject,
    colors = cols,
    startangle = 90, #起始绘制角度，startangle 默认值是 None,图从 x 轴正方向逆时针画起;如
#果 startangle 取 90,则从 y 轴正方向画起
    shadow = True,                     #是否设置阴影效果
    autopct = '%1.2f%%')               #设置百分号显示格式
plt.title('流动资产占比')               #设置标题
plt.show()                             #如图 3.12 所示
```

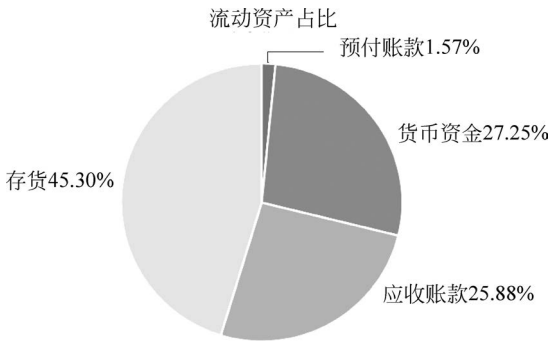

图 3.12　饼图

3.2.6　K 线图

作为金融市场的重要组成部分，股票市场是投资者投资和上市公司筹资的主要场所，推动着资源的优化配置和经济的发展。股票价格的波动反映了宏观经济走向和投资者的预期。因此，分析股票市场涨跌，对于评估股票市场的投资价值、分析股票市场的风险等方面都具有重要的作用。分析股市涨跌的一个重要的工具是 K 线图。

接下来，以比亚迪公司(002594)2022 年的股票周交易数据为例，绘制图表。

```
import akshare as ak
#选取比亚迪公司 2022 年 1 月 1 日～2022 年 12 月 30 日周交易数据
biyadi_weekly = ak.stock_zh_a_hist(symbol = '002594', period = 'weekly',
start_date = '20220101', end_date = '20221230')
path = 'C:\\量化金融\\第 3 章\\第 3 章数据'
biyadi_weekly.to_excel(path + '\\' + '002594 数据.xlsx')  #将数据存储在 Excel 文档中
```

Akshare 是基于 Python 的金融数据获取和分析工具，它提供了广泛的金融市场数据，包括股票、期货、外汇、基金等各类市场数据。通过在 Python 中安装该库就可以获取目标公司相应股票数据。

K 线图主要用于金融领域，特别是在股票、期货、外汇等市场。它以独特的方式展现了金融市场的动态，为投资者提供了有效的分析工具。K 线图会记录一段时间内的开盘价、收盘价、最高价和最低价，以图形的形式直观地展示了价格和交易量的变化情况，不仅可以用来分析价格的涨跌趋势，还是投资者判断买卖的重要工具。

下面以比亚迪公司周交易数据为例，绘制 K 线图。

```
#K 线图
import pandas as pd
import mplfinance as mpf
#mplfinance 是一个 Python 库，构建在 Matplotlib 的基础上，提供了专门用于绘制金融图表的高级
#工具和函数，是绘制 K 线图必不可少的库
#将数据按照日期进行排序，并将日期设置为索引
biyadi_weekly = biyadi_weekly.sort_values('日期')  #对日期进行排序
biyadi_weekly['日期'] = pd.to_datetime(biyadi_weekly['日期']) #将原始数据集中的日期转换为
                                                              #Pandas 日期时间数据类型
biyadi_weekly.set_index('日期',inplace = True)
#绘制 K 线图，如图 3.13 所示
```

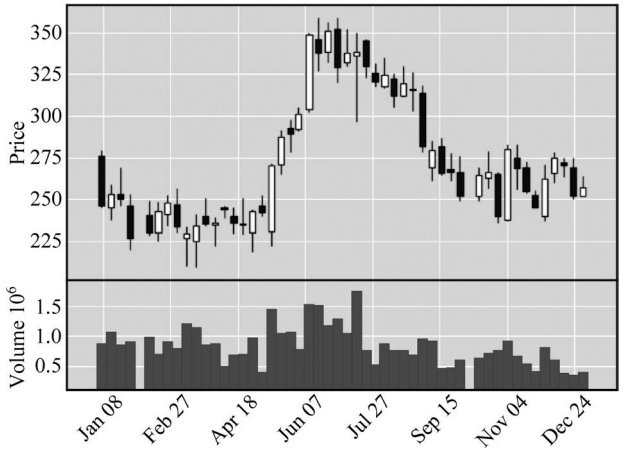

图 3.13　K 线图

```
data = biyadi_weekly[['开盘','最高','最低','收盘','成交量']]
data.columns = ['Open','High','Low','Close','Volume']
data.index.name = 'Data'
data = data.astype(float)
mpf.plot(data,type = 'candle',volume = True,show_nontrading = True)
```

K 线图绘制代码 mpf.plot(data,type ='candle',volume=True,show_nontrading=True)中各参数的作用如表 3.5 所示。

表 3.5　plot()函数的参数及作用

参　数	作　用
type='candle'	指定绘制的图表类型为 candle(即 K 线图)
volume=True	指定是否绘制成交量图
show_nontrading=True	指定是否显示非交易日的数据

3.2.7　雷达图

雷达图是一种显示分析对象多因素(性能)的图形方法,它是以从同一点开始,在轴上表示的三个或更多个取值确定的因素的二维图表达方式。雷达图适用于表示单个或多个分析对象,特别适用于比较分析拥有多个性能数据的对象。

1. 绘制雷达图

```
# - * - coding: utf - 8 - * - #如果没有此代码,则程序运行会报错,错误提示是 SyntaxError: Non
# - UTF - 8 code
#starting with '\xe4' in file,说明 Python 代码文件中包含非 UTF - 8 编码的数据。增加此代码,用
#于编码声明。用#coding = utf - 8 也可以达到同样效果
import numpy as np
import matplotlib.pyplot as plt
#设置中文显示
plt.rcParams['font.sans - serif'] = ['SimHei']
score = [{'语文': 88, '数学': 92, '英语': 95, '体育': 92, '音乐': 99},
         {'语文': 78, '数学': 86, '英语': 95, '体育': 98, '音乐': 89}]
data_length = len(score[0])
#将极坐标根据数据长度进行等分
```

```
angles = np.linspace(0, 2 * np.pi, data_length, endpoint = False)
#此处 linspace()函数的第一个参数传入起始角度,第二个参数传入结束角度,第三个参数传入分
#成多少等份,如果不选,则默认是 50。其他参数根据需要传入,如 endpoint 默认为 True,意味着取
#的点中包括最后一个数据。如果 endpoint 取 False,则不包括
labels = [key for key in score[0].keys()]
#score 是一个列表,列表的元素是由两个字典组成,字典包括两个学生的 5 门课程的成绩;score[0]
#是第 1 个字典;score [0].keys()是第 1 个字典的 dict_keys,即 dict_keys(['语文', '数学',
#'英语', '体育', '音乐'])。for key in score[0].keys(),也就是循环变量 key 依次读取第 1 个字典
#的键,形成 labels,labels = ['语文', '数学', '英语', '体育', '音乐']。因为两个学生的 5 门课的名
#称一样,所以通过读取第 1 个学生的字典就可以形成绘图的标签,不需要再读取第 2 个学生的字典来
#形成标签
Score = [[v for v in result.values()] for result in score]
#for result in score,也就是循环变量 result 依次读取 score。由于 score 是由两个字典组成的列
#表,所以 result 只需要读取两次。result 第 1 次读取 score,得到第 1 个字典{'语文': 88, '数学':
#92, '英语': 95, '体育': 92, '音乐': 99}
#result 读取第 1 个字典时,result.values()的值是 dict_values([88, 92, 95, 92, 99])。此时 v
#依次读取 dict_values,第 1 次读取 88
#result 读取完两个字典后,得到的 Score 是一个列表,列表元素包含两个列表,结构是[[88, 92,
#95, 92, 99], [78, 86, 95, 98, 89]]
#使雷达图数据封闭
score_a = np.concatenate((Score[0], [Score[0][0]]))
#Score[0]是读取列表 Score 的第 1 个元素[88, 92, 95, 92, 99],[Score[0][0]]是[88],注意,此处
#是[88],而不是 88。通过 np.concatenate 将[88, 92, 95, 92, 99]和[88]拼接在一起,形成[88 92
95 92 99 88],该结果是 numpy.ndarray 类型
score_b = np.concatenate((Score[1], [Score[1][0]]))
angles = np.concatenate((angles, [angles[0]]))
#在拼接之前,angles 是[0. 1.25663706 2.51327412 3.76991118 5.02654825],[angles[0]]是[0]。
#拼接后是[0. 1.25663706 2.51327412 3.76991118 5.02654825 0.]
labels = np.concatenate((labels, [labels[0]]))
#设置图形的大小
fig = plt.figure(figsize = (8, 6), dpi = 100)
#新建一个极坐标图,其中,polar 参数必须设置为 True,得到的图形才是极坐标
ax = plt.subplot(111, polar = True)
#绘制雷达图
ax.plot(angles, score_a, color = 'g')
ax.plot(angles, score_b, color = 'b')
#设置雷达图中每一项的标签显示
ax.set_thetagrids(angles * 180 / np.pi, labels)
#设置雷达图的 0°起始位置
ax.set_theta_zero_location('N')
#设置雷达图的坐标刻度范围
ax.set_rlim(0, 100)
#设置雷达图的坐标值显示角度,相对于起始角度的偏移量
ax.set_rlabel_position(270)
ax.set_title('成绩')
plt.legend(['小明', '小华'], loc = 'best')
plt.show()  #如图 3.14 所示
```

2. 重要函数介绍

linspace(start, stop, num = num_points, endpoint = True, retstep = False, axis = 0, dtype=int)是 NumPy 库中用于创建等差数列的函数。其中：

- start：数列的起始点,如果设置为 0,则结果的第一个数为 0。

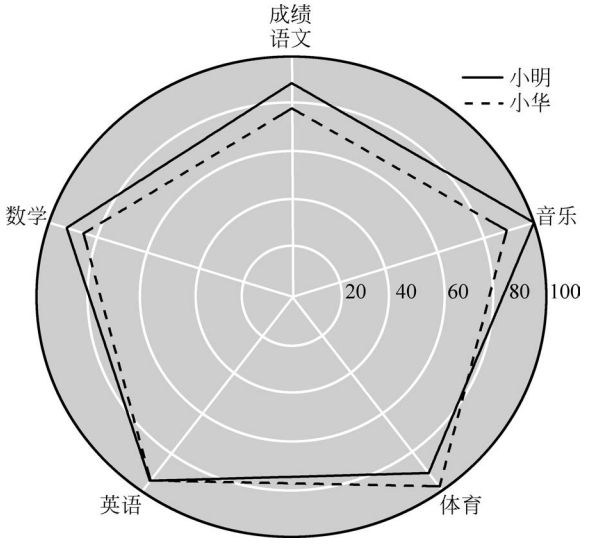

图 3.14　雷达图

- stop：数值范围的终止点。
- num：控制结果中共有多少个元素。
- endpoint：终止值是否被包含在结果数组中，默认为 True，即终止值包含在数组中。
- dtype：输出数组的数据类型。

3.2.8　热力图

热力图是一种用颜色表达数据密度的可视化工具，用于展示数据的分布情况和集中程度，表现数据的趋势和模式。绘图时，一般较大的值由较深的颜色表示，较小的值由较浅的颜色表示。

1. 热力图的功能

(1) 呈现数据的分布和集中程度：通过颜色的深浅，可以直观地看到数据在不同区域的集中程度。

(2) 观察数据的趋势和模式：通过对比不同时间或不同条件下的热力图，可以发现数据的变化趋势和潜在的模式。

(3) 展现区域特征：在商业决策中，可以使用热力图来跟踪用户的区域消费等行为，分析区域销售热点、顾客流动性、从而了解用户的兴趣和行为模式。

2. 绘制热力图

```
#coding = utf8
import numpy as np
import matplotlib.pyplot as plt
data = np.array([[600, 730, 860, 530],                    #销售数据
                 [300, 280, 430, 510],
```

```
                  [200, 450, 650, 780],
                  [340, 560, 330, 440]])
regions = ['Region 1', 'Region 2', 'Region 3', 'Region 4']        #地区标签
products = ['Product 1', 'Product 2', 'Product 3', 'Product 4']  #产品标签
plt.imshow(data, cmap = 'hot', interpolation = 'nearest')         #绘制热力图
plt.xticks(range(len(products)), products)                        #设置坐标轴标签
plt.yticks(range(len(regions)), regions)
plt.colorbar()                                                    #添加颜色条
plt.title('Sales Heatmap')                                        #设置标题
plt.show()                                                        #如图 3.15 所示
```

图 3.15　热力图

3.2.9　属性两两分析图

属性两两分析图(也称为双变量图)是一种用于展示两个属性(也称为变量、特征)之间关系的可视化工具。这种图将两个属性的数据点绘制在二维坐标系上,其中每个数据点表示一个观测值或记录,横坐标和纵坐标分别表示两个属性的值。

1. 属性两两分析图的功能

(1) 展示关系:可以直观地发现两个属性之间的关系。

(2) 识别异常值:在属性两两分析图中,离群的数据点往往更容易被识别出来。

(3) 评估数据分布:通过观察数据点在坐标系上的分布,可以了解每个属性的数据分布情况,如是否偏斜、是否存在多峰等。

在数据分析的初步阶段,当需要研究两个属性之间是否存在相关性时,可以使用属性两两分析图。在机器学习和数据挖掘中,经常使用属性两两分析图来评估不同特征之间的关系,进而选择出对模型预测性影响较大的特征。

注意:属性两两分析图只能展示两个属性之间的关系,如果要分析多个属性之间的关系,可能需要使用其他类型的可视化工具,如平行坐标图、雷达图等。

2. 绘制属性两两分析图

```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
plt.rcParams['font.sans-serif'] = ['SimHei']   #设置中文显示
plt.rcParams['axes.unicode_minus'] = False     #坐标轴显示负号
path = 'C:\\量化金融\\第3章\\第3章数据'
data = pd.read_excel(path + '\\' + '上市公司风险预警样本.xlsx',sheet_name = 2)
#对数据中的指定属性进行两两分析
sns.pairplot(data,vars =['ROE(摊薄)(%)', 'ROE(加权)(%)', 'ROA(%)', 'ROIC(%)'])
plt.show()                                     #显示如图3.16所示
```

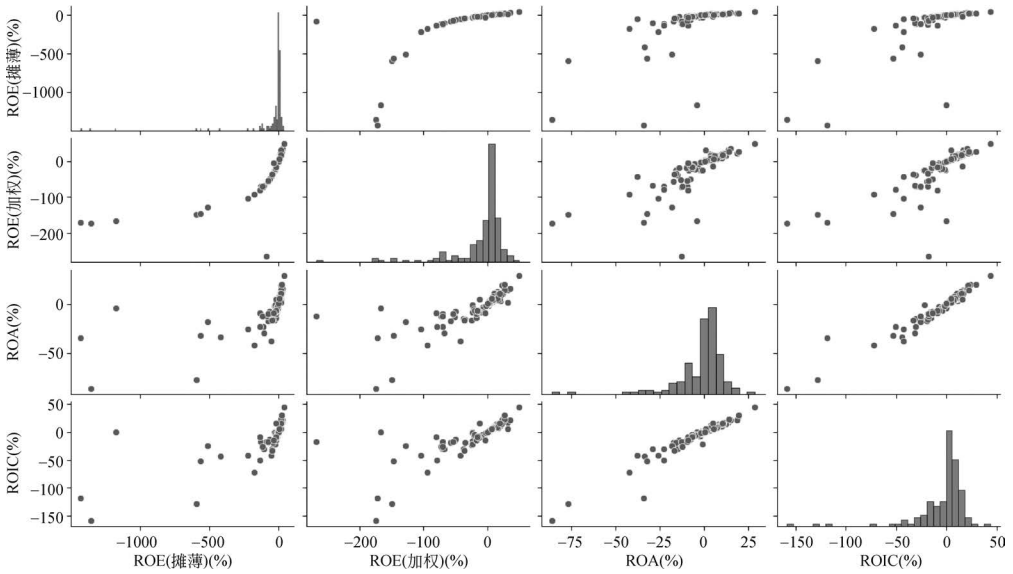

图 3.16　两两分析图

pairplot()函数主要展现的是变量两两之间的关系(线性或非线性,有无较为明显的相关关系)。默认情况下,pairplot()函数绘制的图中对角线上是各个属性的直方图(分布图),而非对角线上是两个不同属性之间的相关图。当然,这并不是一成不变的,可以通过设置sns. pairplot()函数中的相关参数调整图的类型(见表 3.6)。

表 3.6　sns. pairplot()函数的参数及作用

参　数	作　用
var	用于指定要在图中包含的变量。默认情况下,使用 DataFrame 中的所有数值列
data	用于绘图的数据集
kind	用于控制非对角线上的图的类型,可选'scatter'与'reg',如 kind ='scatter'则会在非对角线上绘制散点图
diag_kind	控制对角线上的图的类型,可选'hist'与'kde',如 diag_kind ='hist',在对角线上绘制直方图

3.2.10　气泡图

气泡图是一种强大的数据可视化工具，它主要通过气泡的位置以及面积大小来比较和展示不同类别气泡之间的关系。气泡图能清晰地展示三个连续字段之间的关系，直观地比较出不同数据点之间的数值差异。

通过观察气泡的排列和大小变化，使用者可以分析数据之间的相关性，发现数据中的趋势、模式以及潜在的关联。实质上，气泡图就是具有不同直径绘制而成的散点图。

```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import numpy as np
plt.rcParams['font.sans-serif'] = ['SimHei']      #设置中文显示
plt.rcParams['axes.unicode_minus'] = False        #坐标轴显示负号
path = 'C:\\量化金融\\第 3 章\\第 3 章数据'
data = pd.read_excel(path + '\\' + '上市公司风险预警样本.xlsx',sheet_name =
1,index_col = 0)
print(data)
df1_1 = data.loc['ROE(摊薄)(%)']
df1_2 = data.loc['ROE(加权)(%)']
df1_3 = data.loc['扣费后 ROE(摊薄)(%)']
df1_4 = data.loc['ROA(%)']
fig = plt.figure('Draw')
plt.scatter(df1_1, df1_2,c = 'y')
plt.scatter(df1_1, df1_2, s = df1_3,c = 'b')      #点的直径大小为扣费后 ROE(摊薄)(%)
plt.scatter(df1_1, df1_2, s = df1_3 * 10,c = 'r') #点的直径大小为 10 倍扣费后 ROE(摊薄)(%)
plt.ylabel('ROE(加权)')                           #纵坐标轴标题
plt.show()                                        #显示如图 3.17 所示
```

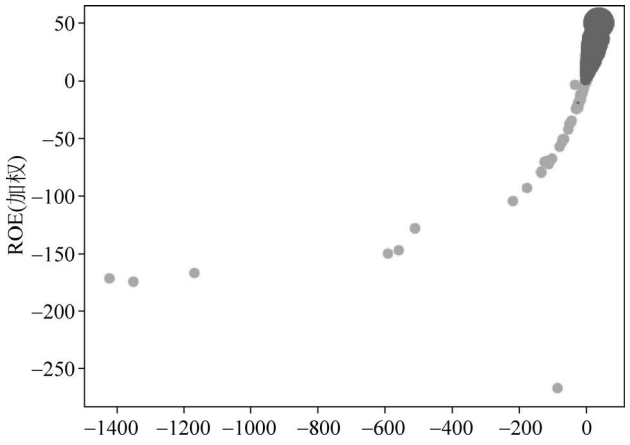

图 3.17　气泡图

由图 3.17 可以直观地看到虽然有个别极端值，但总体而言加权 ROE 和扣费后摊薄 ROE 随着摊薄 ROE 的增加而呈现上涨趋势，存在正相关性。

3.2.11　小提琴图和分簇散点图

在数据分析中,小提琴图(见图 3-18)可以通过绘制每个类别的数据分布,直观地展示数据的集中、分散、偏态等特征。数据可视化中的小提琴图结合了箱线图和核密度估计图的特点,用于显示数据的分布形状、中心和离散度,在分析三个因素之间关系的时候,比较适合分析其中一个属性的属性值是分类数据。

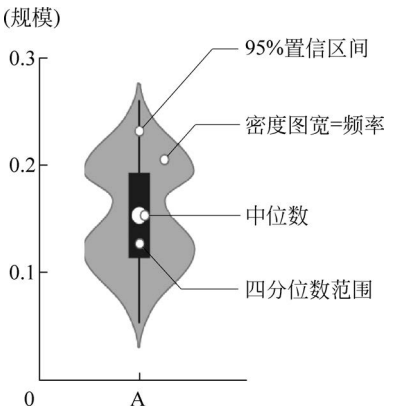

图 3.18　小提琴图

分簇散点图结合了散点图和密度估计技术,可用于展示两个变量之间的关系以及它们各自的分布密度。在分簇散点图中,每个数据点代表两个变量的一个观测值。通过颜色或大小的变化,可以进一步区分不同的数据子集或类别。

通过分簇散点图中数据点在图中的分布模式可以观察到变量之间的关系,识别数据的峰值和谷值,从而了解数据的集中程度和离散程度。

```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
path = 'C:\\量化金融\\第 3 章\\第 3 章数据'
data = pd.read_excel(path + '\\' + '上市公司风险预警样本.xlsx',sheet_name = 2,index_col = 0)
print(data)
plt.rcParams['font.sans-serif'] = ['SimHei']   #设置中文显示
plt.rcParams['axes.unicode_minus'] = False     #坐标轴显示负号
plt.rcParams.update({'font.size':14})
#将画布分为 2 行 2 列
fig,axes = plt.subplots( 2,2,figsize = (18, 12))
data['ROA'] = data['ROA( % )'].map(lambda x: '1' if x > 0 else '0')
#绘制小提琴图
sns.violinplot(x = 'ROA', y = '每股销售额 SPS', hue = '正常', data = data,
palette = 'autumn',ax = axes[0][0]).set_title('ROA and SPS vs 正常')
#绘制分簇散点图
sns.swarmplot(x = 'ROA', y = '每股销售额 SPS',hue = '正常',
data = data,palette = 'autumn',ax = axes[1][0]).legend(loc = 'upper right').set_title('正常')
#绘制小提琴图
sns.violinplot(x = 'ROA', y = '每股净资产 BPS', hue = '正常', data = data,
palette = 'winter', ax = axes[0][1]).set_title('ROA and BPS vs 正常')
#绘制分簇散点图
sns.swarmplot(x = 'ROA', y = '每股净资产 BPS',hue = '正常',
data = data,palette = 'winter',ax = axes[1][1]).legend(loc = 'upper right').set_title('正常')
plt.show()                                     #显示如图 3.19 所示
```

Python 中绘制小提琴图的语法为 sns. violinplot(x=None,y=None,hue=None,data=dataframe,order=None,hue_order=None,bw='scott',scale='area',inner='box',split=False,palette=None),其中各参数的作用如表 3.7 所示。

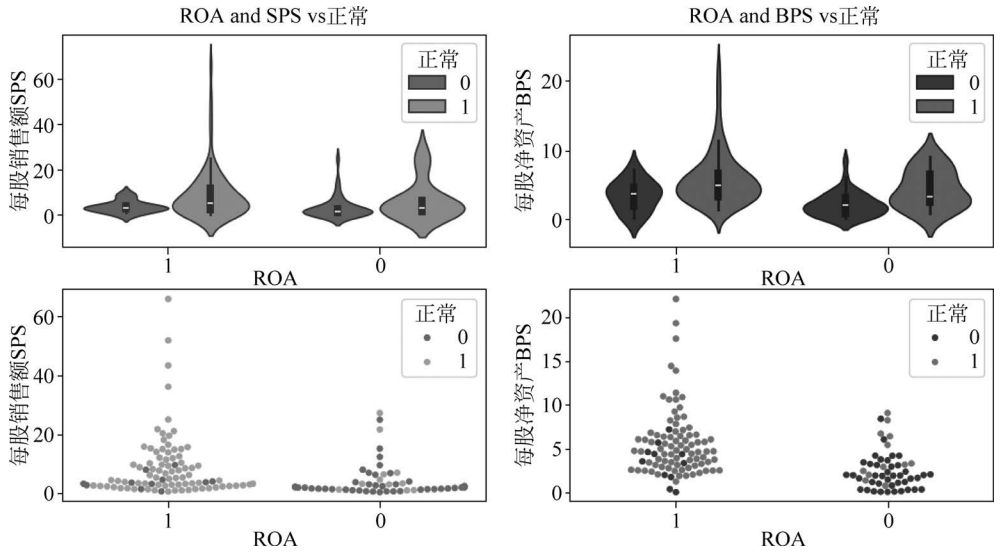

图 3.19　小提琴图和分簇散点图

表 3.7　小提琴图绘制参数及作用

参　数	作　用
x,y	指定数据的 x 轴和 y 轴,也可以在数据集中指定 x 和 y 的列名
hue	用于根据其值对数据进行分组,生成不同颜色的小提琴图
order,hue_order	指定 x 轴或 hue 中类别的显示顺序
bw	控制核密度估计的带宽大小
scale	控制小提琴图的宽度。可以设置为'area'(面积相等,默认值)、'count'(按照样本数量标准化)、'width'(固定宽度)
inner	设置小提琴内部图形的类型,可以是'box'、'quart'、'point'、'stick'或 None。默认为'box'
split	当使用 hue 参数进行分组时,设置为 True 时,将小提琴图拆分成两半,分别表示不同的 hue 类别
palette	指定颜色调色板,用于不同类别的颜色着色

Python 中绘制分簇散点图的语法为 seaborn. swarmplot(x=None,y=None,hue=None,data=None,jitter = None,order=None,hue_order=None,dodge=False,orient=None,color=None,palette=None,size=5,edgecolor='gray',inewidth=0),其中各参数的作用如表 3.8 所示。

表 3.8　分簇散点图绘制参数及作用

参　数	作　用
x,y	指定数据的 x,y 轴
hue	一个列名或变量,用于根据其值对数据进行分组,常用来指定第二次分类的数据类别
data	指定数据集
jitter	当数据点重合较多时,设置为 True 可以使数据分散开
order,hue_order	显式指定分类顺序
dodge	若设置为 True 则沿着分类轴将数据分离出来成为不同色调级别的条带,否则,每个级别的点将相互叠加

续表

参　数	作　用
orient	设置图的绘制方向(垂直或水平)
color	设置颜色
palette	对数据不同的分类进行颜色区别
size	设置标记直径大小
edgecolor	设置每个点的周围线条颜色
linewidth	设置构图元素的线宽度

习题

1. 分别运用 subplot()函数、subplots()函数、subplot2grid()函数在同一张画布上绘制 $y=2x$,$y=\ln x$,$y=\tan x$,$y=x^2$ 的图像。

2. 绘制近 10 天的温度图,要求图像样式为红色点画线,星形标记,并具有坐标轴标题,以 PNG 形式保存。

3. 找到一家感兴趣的上市公司,绘制其近 3 个月的股票交易数据的 K 线图。