BIG DATA

第5章 数据可视化

本章学习目标

- 掌握 Matplotlib 绘制基本图形的方法。
- 掌握 Matplotlib 绘制子图的方法。
- 掌握 pyecharts 绘制基本图形和组合图形的方法。

本章主要介绍 Matplotlib 和 pyecharts 两个可视化图形库，重点讲解用这两个库绘制折线图、柱状图、散点图、直方图等常见图形的方法。

俗话说，千言万语不如一张图。数据可视化(data visualization)通过图形能够清晰有效地表达数据，属于数据探索过程中的一部分。利用可视化技术可以识别异常值或所需的数据转换，也可以发现原始数据中不易观察到的数据联系。因此，数据可视化是数据分析中非常重要的内容。

Python 中有许多附加库可以用来制作静态或动态的可视化图形，其中使用最多的可视化工具是 Matplotlib 库和 pyecharts 库。本章介绍如何使用这两种库绘制常用的数据图表，如折线图、柱状图、散点图和直方图等。

5.1 Matplotlib 可视化

Matplotlib 是可以生成静态、动态和交互式可视化图形的绘图库。Matplotlib 最初由 John D. Hunter 于 2002 年编写，首次发表于 2007 年。Matplotlib 在 Python 环境下能够进行 MATLAB 风格的绘图，所以名字以 Mat 开头，中间的 plot 表示绘图这一作用，而结尾的 lib 则表示它是一个集合。近年来，在 Github 等开源社区的推动下，Matplotlib 成为在 Python 中使用最多的绘图工具包之一，在数据分析和科学计算等领域得到广泛应用。

Matplotlib 中应用最广的是 Matplotlib. pyplot 模块，用户只需要调用 Pyplot 中的函数，就能够绘制折线图、柱状图、散点图、直方图、箱线图、热力图等图形，实现快速绘图并设置图表的各个细节。

5.1.1 Matplotlib 基本图形

在 Jupyter Notebook 中显示图形，需要加入%matplotlib inline 魔法函数(以%开头)。使用 Matplotlib 时，其导入语法格式如下：

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

除此之外，运行本章的代码还需要引入 NumPy 和 Pandas 库，语法格式如下：

```
import numpy as np
```

```
import pandas as pd
from pandas import DataFrame,Series
%matplotlib inline   #在 Notebook 中显示图形
```

需要注意的是：Matplotlib 默认为英文字体，如果绘制图形中出现汉字则无法显示。因此，还需要指定 Matplotlib 的默认字体，需要加入如下的代码：

```
plt.rcParams['font.sans-serif'] = ['SimHei']          #用来显示中文标签
plt.rcParams['axes.unicode_minus'] = False            #用来正常显示负号
```

1. 折线图

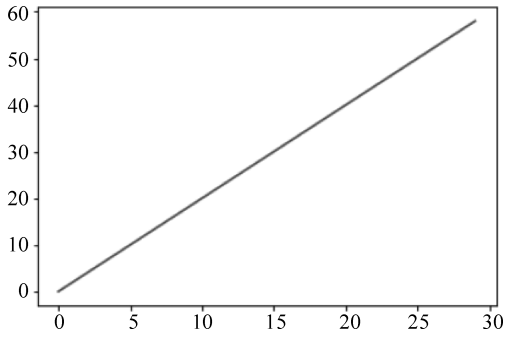
折线图(line chart)是最基本的一种图表，一般用于绘制连续型数据，也可以看作是将散点图按照 x 轴坐标顺序连接起来的图形，常用来表现数据的变化趋势。例如，可以通过绘制折线图来分析产品销量随着年份变化的趋势。

1) 基本折线图

绘制折线图可使用 plot()函数。

【例 5-1】 绘制基本折线图。

```
In [1]: import numpy as np
        import matplotlib.pyplot as plt
        %matplotlib inline
        x = np.arange(30)
        y = x * 2
        plt.plot(x, y)
Out [1]:
    [<matplotlib.lines.Line2D at 0x55143f0>]
```



2) plot()函数的主要参数

plot()函数的参数有很多，详见官方文档。表 5.1 中列出了常用的参数。

表 5.1 plot()函数的主要参数及说明

参 数	类 型	说 明
x,y	ndarray 数组	x 轴与 y 轴对应的数据
color	string	表示线条的颜色
marker	string	表示线上数据点的样式
linestyle	string	表示线条的类型
linewidth	float	表示线条的粗细
alpha	float	0～1 的小数，表示图形的透明度

color 参数用来指定线条的颜色，其取值如表 5.2 所示。

表 5.2　color 参数取值

取　值	代表的颜色	取　值	代表的颜色
'b'	蓝色	'm'	洋红
'g'	绿色	'y'	黄色
'r'	红色	'k'	黑色
'c'	青绿	'w'	白色

marker 参数用于标记坐标点的样式，设置方式如表 5.3 所示。

表 5.3　marker 参数取值

取　值	含　义	取　值	代表的颜色
'.'	点标记	'*'	星形标记
','	像素点标记	'D'	钻石标记
'o'	圆形标记	'+'	加号标记
'v'	三角标记	'x'	x 标记
's'	方形标记	'_'	水平线标记

linestyle 参数用来指定线条的类型，其取值如表 5.4 所示。

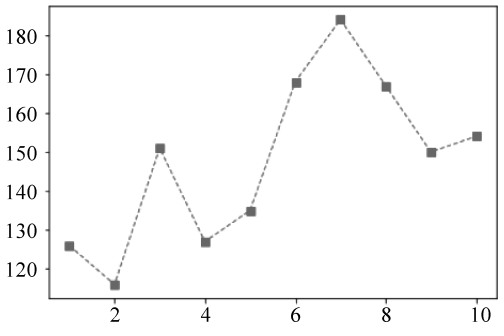
表 5.4　linestyle 参数取值

取　值	线　型
'-'	实线
'--'	短画线
'-.'	点画线
':'	虚线

其他参数设置请详见 Matplotlib 官方文档。

【例 5-2】 plot()函数参数设置。

```
In [2]: times = np.arange(1,11)
        sales = np.random.randint(100,200,10)
        plt.plot(times, sales, color = "r", linewidth = 1.0, marker = 's', linestyle = "--")
        #点标记为方形,线宽为 1 的红色短画线
Out [2]:
        [<matplotlib.lines.Line2D at 0xb1289d0>]
```



plot()函数的参数也可以不使用关键字，而用样式字符串替代，如'rs'代表红色方形标记。因此，例 5-2 的 plot()函数也可以简化如下：

```
plt.plot(times, sales, 'rs--')
```

一般来说，样式字符串中的标记类型、线类型跟在颜色类型的后面。

此外，Series 和 DataFrame 都有一个 plot 属性，用于绘制基本的图形。默认情况下，plot()函数绘制的是折线图。

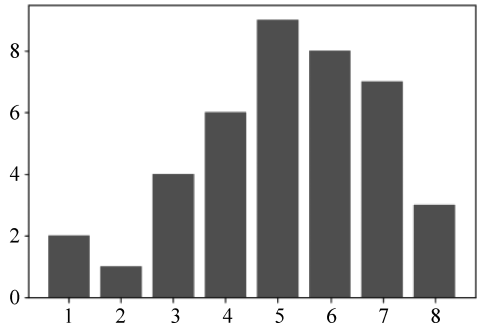
2. 柱状图

1) 基本柱状图

柱状图(bar diagram)主要用于分析在 x 轴上定性数据的分布特征。一般情况下，横轴表示数据类别，纵轴表示数量或者比率。绘制柱状图主要使用 bar()函数。

【例 5-3】 绘制基本柱状图。

```
In [3]: x = [1,2,3,4,5,6,7,8]
        y = [2,1,4,6,9,8,7,3]
        plt.bar(x,y)
Out [3]:
        <BarContainer object of 8 artists>
```



2) bar()函数的主要参数

bar()函数的主要参数如表 5.5 所示。

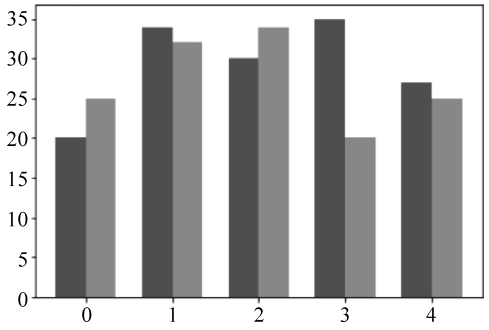
表 5.5 bar()函数的主要参数

参　数	类　型	说　明
x	ndarray 数组	表示 x 轴的坐标
height	ndarray 数组	表示条形高度
width	float	默认值 0.8，表示条形宽度
color	string	表示条形的颜色
align	'center'或 'edge'	默认值'center'，表示条形在 x 轴的对齐方式
bottom	数值或 ndarray 数组	默认值为 0，表示条形在 y 轴的位置
alpha	float	0～1 的小数，表示图形的透明度

利用 width 参数，可以绘制并列柱状图。

【例 5-4】 绘制并列柱状图。

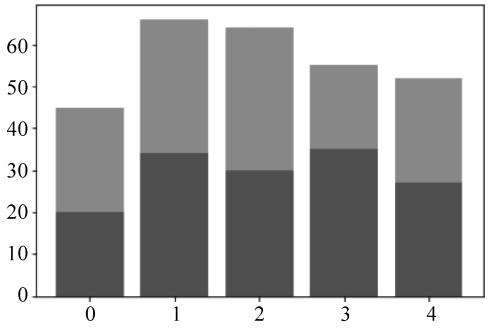
```
In [4]: men_means = [20, 34, 30, 35, 27]
        women_means = [25, 32, 34, 20, 25]
        x = np.arange(len(men_means))
        width = 0.35
        rects1 = plt.bar(x - width/2, men_means, width)
        rects2 = plt.bar(x + width/2, women_means, width)
Out [4]:
```



此外，利用 bottom 参数可以绘制堆积柱状图。

【例 5-5】 绘制堆积柱状图。

```
In [5]: men_means = [20, 34, 30, 35, 27]
        women_means = [25, 32, 34, 20, 25]
        x = np.arange(len(men_means))
        rects1 = plt.bar(x, men_means )
        rects2 = plt.bar(x, women_means,bottom = men_means)
Out [5]:
```



3）设置刻度和标签

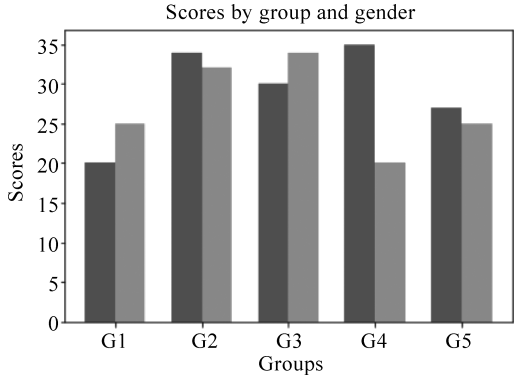
默认情况下，柱状图 x 轴的刻度由函数参数 x 决定。要改变 x 轴的刻度，需要使用 xticks()函数。xticks()函数可以设置 x 轴的刻度。类似地，通过 yticks()函数可以设置 y 轴的刻度。

通过 xlable()函数和 ylabel()函数分别给 x 轴和 y 轴设置名称，通过 title()函数可以给图像添加标题。

【例 5-6】 设置刻度和标签。

```
In [6]: labels = ['G1', 'G2', 'G3', 'G4', 'G5']
        men_means = [20, 34, 30, 35, 27]
```

```
        women_means = [25, 32, 34, 20, 25]
        x = np.arange(len(labels))
        width = 0.35
        rects1 = plt.bar(x - width/2, men_means, width)
        rects2 = plt.bar(x + width/2, women_means, width)
        #添加图表标题,x轴和y轴标签
        plt.xlabel('Groups')
        plt.ylabel('Scores')
        plt.title('Scores by group and gender')
        #设置x轴刻度值和刻度标签
        plt.xticks(x,labels)
Out[6]:
```



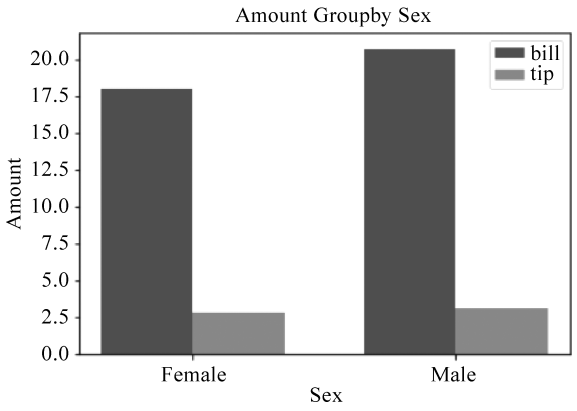
4）添加图例

图例是用来区分绘图区元素的主要工具。有很多方式可以添加图例，最简单的方式是在 bar()函数中传递 label 参数表明图例名称，再通过 legend()函数绘制图例。下面以小费数据为例，介绍如何绘制柱状图。该例中，通过分组计算了不同性别的账单和小费金额的平均值。

【例 5-7】 添加图例。

```
In[7]: tips = pd.read_csv('tips.csv')
        #按照性别,分组计算账单和小费平均金额
        billmean = tips.groupby('sex')['total_bill'].mean()
        tipmean = tips.groupby('sex')['tip'].mean()
        #刻度标签
        labels = ['Female', 'Male']
        x = np.arange(len(labels))
        width = 0.35
        rects1 = plt.bar(x - width/2, billmean, width,label = 'bill')
        rects2 = plt.bar(x + width/2, tipmean, width,label = 'tip')
        #添加图表标题,x轴和y轴标签
        plt.xlabel('Sex')
        plt.ylabel('Amount')
        plt.title('Amount Groupby Sex')
        #设置x轴刻度值和刻度标签
        plt.xticks(x,labels,fontsize = 12)
        #设置图例
```

```
        plt.legend()
Out [7]:
```

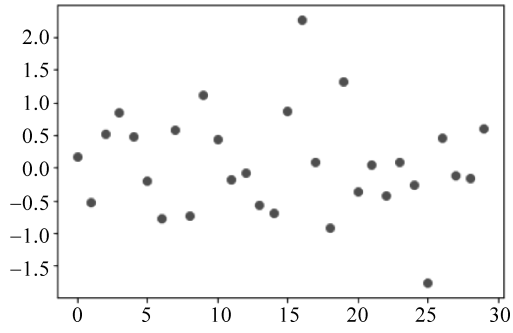


3. 散点图

散点图(scatter diagram)根据两个一维数组绘制坐标点,通过坐标点的分布来判断变量之间是否存在某种关联或总结坐标点的分布模式。在同时考察多个变量间的关系时可以借助散点图矩阵。使用 scatter()函数绘制散点图。

【例 5-8】 绘制散点图。

```
In [8]: x1 = np.arange(30)
        y1 = np.random.randn(30)
        plt.scatter(x1,y1)
Out [8]:
        <matplotlib.collections.PathCollection at 0xe22e2f0>
```



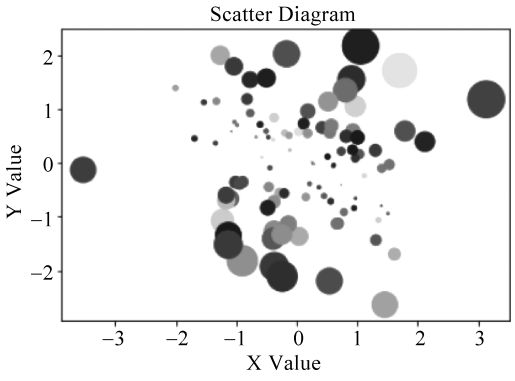
scatter()函数的主要参数如表 5.6 所示。

表 5.6 scatter()函数的主要参数

参　数	类　型	说　明
x,y	数值或一维数组	表示 x 轴与 y 轴对应的数据点
s	数值或一维数组	表示数据点的大小,若传入数组则表示每个点的大小
c	颜色值或一维数组	表示数据点的颜色,若传入数组则表示每个点的颜色
marker	string	表示数据点的样式
alpha	float	0～1 的小数,表示数据点的透明度

【例 5-9】 设置散点图参数。

```
In [9]: a = np.random.randn(100)
        b = np.random.randn(100)
        plt.scatter(a,b,s = np.power(5 * a + 10 * b,2),c = np.random.rand(100),marker = 'o')
        plt.xlabel('X Value')
        plt.ylabel('Y Value')
        plt.title('Scatter Diagram')
Out [9]:
```



4. 直方图和密度图

1) 直方图

直方图(histogram)是一种用来展现连续型数据分布特征的统计图形。利用直方图可以直观地观察数据的集中和分散趋势。直方图主要应用于连续型数据的可视化展示,例如,观察成绩的区间分布情况或者人均收入的分布特征等。

在直方图中,数据点被分成离散的、均匀间隔的箱,并且绘制每个箱中数据点的数量。在数据分析中,可以借助直方图对连续数据进行离散化处理。

使用 hist()函数绘制直方图,其主要参数如表 5.7 所示。

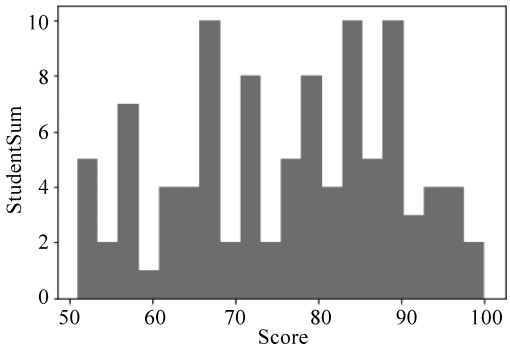
表 5.7 hist()函数的主要参数

参 数	类 型	说 明
x	一维数组	每类数据的大小,无默认值
bins	整形	直方图中箱子的个数,默认值为 10
color	颜色值	设置箱子的颜色,默认值为 None
histtype	集合{'bar', 'barstacked', 'step', 'stepfilled'}	箱子的类型,默认值为 bar
density	bool	是否将得到的直方图向量归一化,默认为 False,不归一化,显示频数; 若为 True,归一化,显示频率
rwidth	float	箱子间的距离,默认值为 None

【例 5-10】 绘制直方图。

```
In [10]: x = np.random.randint(50,101,size = 100)
         plt.hist(x,bins = 20,density = False,color = 'g',alpha = 0.75)
         plt.xlabel("Score")
```

```
        plt.ylabel("StudentSum")
        plt.show()
Out[10]:
```



注意：前面介绍了柱状图和直方图的概念和绘制方法。柱状图和直直方图在展现效果上是非常类似的，区别在于直方图描述的是连续型数据的分布，而柱状图描述的是离散型数据的分布。

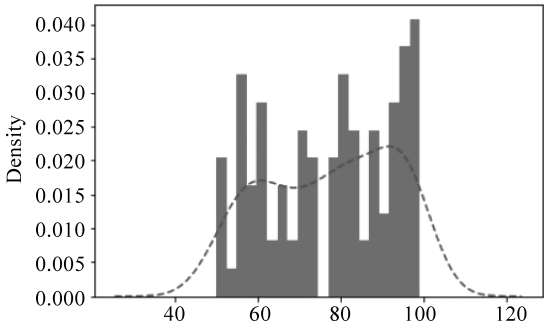
2）密度图

密度图是一种与直方图相关的图表类型，通过计算可能产生观测数据的连续概率分布估计而生成。其过程是将数据的分布近似为一组核（如正态分布），因此密度图也被称为内核密度估计（kernel density estimate，KDE）图。

可以通过 Series 类型的 plot()函数绘制密度图，设置函数参数 kind 为'kde'，例如：

【例 5-11】 绘制密度图。

```
In [11]: x = np.random.randint(50,101,size = 100)
         plt.hist(x,bins = 20,density = True,color = 'g',alpha = 0.75)
         #density = True,直方图向量归一化处理
         s = pd.Series(x) #创建 Series 类型
         s.plot(kind = 'kde',linestyle = '--') #绘制密度图
         plt.show()
Out[11]:
```



5. 饼图

饼图（pie graph）用于表示不同类别的占比情况。通过各类别所占面积的大小可以清楚地反映出各部分之间或者部分与整体之间的比例关系。在 Matplotlib 中使用 pie()函数绘制饼图。

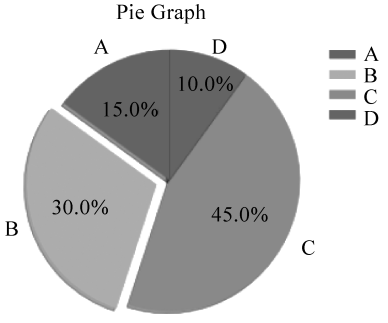
pie()函数的主要参数如表 5.8 所示。

表 5.8 pie()函数的主要参数

参　数	类　型	说　明
x	一维数组	每类数据的大小,无默认值
explode	数组	每部分离圆心的距离,默认值为 None
labels	字符串列表	每部分的标签,默认值为 None
colors	包含颜色字符串的数组	设置饼图的填充色,默认值为 None
autopct	string	设置数值的显示方式,默认值为 None
pctdistance	float	设置百分比标签与圆心的距离,默认值为 0.6
labeldistance	float	设置每部分标签与圆心的距离,默认值为 1.1
shadow	bool	是否添加饼图的阴影效果,默认值为 False
startangle	float	设置饼图的初始摆放角度,默认值为 0
radius	float	饼图的半径,默认值为 1
textprops	dict	设置标签和比例文字的格式,默认值为 None

【例 5-12】 绘制饼图。

```
In [12]:  labels = ['A', 'B', 'C', 'D']                    #每部分的标签
          data = [15, 30, 45, 10]                          #每部分的比例
          explode = (0, 0.1, 0, 0)                         #将第 2 块分离出来
          colors = ['r','g','b','y']                       #设置每部分的颜色
          plt.pie(data, explode = explode, labels = labels, \
              autopct = '%1.1f%%',shadow = True,startangle = 90,\
              textprops = {'fontsize':12,'color':'black'})
          #autopct 在图中显示比例值的格式
          plt.axis('equal')                                #x,y 轴刻度设置一致,保证饼图为圆形
          plt.legend(loc = "upper right")                  #在右上方显示图例
          plt.title("Pie Graph",fontsize = 16)             #显示图表标题
          plt.show()                                       #显示图形
Out [12]:
```



6. 箱线图

箱线图(boxplot)也称为盒须图,是一种常见的用于观察数据分布的图形。通过数据中的 5 个统计量,包括最小值、下四分位数、中位数、上四分位数和最大值来描述数据,可以观察数据是否具有对称性和数据的分散程度,也可以对多个数据集进行比较。

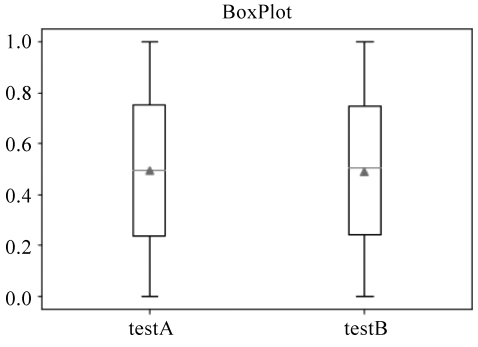
使用 boxplot()函数绘制箱线图,主要参数如表 5.9 所示。

表 5.9　boxplot()函数的主要参数

参　数	类　型	说　明
x	数组或序列	用于绘制箱线图的数据,无默认值
notch	bool	中间箱体是否还有缺口,默认值为 False
sym	string	指定异常点形状,默认值为 None
vert	bool	箱线图是否垂直放置,默认值为 True
widths	float 或者数组	表示线箱体的宽度,默认值为 0.5
labels	字符串序列	箱线图的标签,默认值为 None
showmeans	bool	是否显示均值,默认值为 False
showcaps	bool	是否显示顶端和末端的两条线,默认值为 True

【例 5-13】 绘制箱线图。

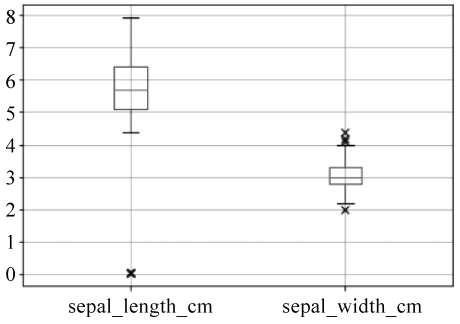
```
In [13]: testA = np.random.rand(500)
         testB = np.random.rand(500)
         labels = ['testA','testB']
         data = [testA,testB]
         plt.boxplot(data,labels = labels,showmeans = True)          #设置标签并显示均值
         plt.title("BoxPlot")
         plt.show()
Out [13]:
```



也可以使用 DataFrame 的 boxplot()函数完成箱线图的绘制。例如,例 5-14 中绘制鸢尾花数据的箱线图。

【例 5-14】 绘制鸢尾花数据箱线图。

```
In [14]: data1 = pd.read_csv('iris - data.csv')
         data1.boxplot(column = ['sepal_length_cm', 'sepal_width_cm'],sym = 'x')
         #对其中两列数据绘制箱线图,异常点用 x 标记
         plt.show()
Out [14]:
```



5.1.2 Matplotlib 自定义设置

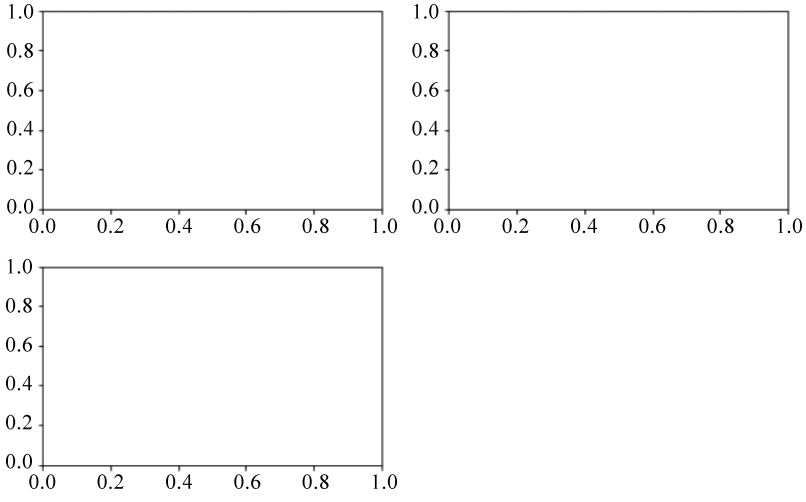
1. 创建画布与子图

Matplotlib 所绘制的图形位于 Figure 对象(画布)中,前面的例子中都是在 Matplotlib 自动创建的 Figure 对象中进行绘图。在默认创建的画布中,只有一个 AxesSubplot 对象(有坐标系的绘图区),因此只能绘制一张图。如果希望在一张画布中绘制多张图,则需要显示地创建一个新的 Figure 对象,并在其中创建多个 AxesSubplot 对象,也可以说是创建多个子图。

使用 plt.figure()函数创建一个新的 Figure 对象,其中 figsize 参数可以设置图表的长宽比。使用 add_subplot()函数创建一个或多个子图。注意:请将例 5-15 的代码放在同一个 Notebook 的单元格中运行。

【例 5-15】 绘制子图。

```
In [15]: fig = plt.figure(figsize = (10,6))                    # 创建画布 fig,并设置画布大小
         ax1 = fig.add_subplot(2,2,1)                          # 创建子图 1
         ax2 = fig.add_subplot(2,2,2)                          # 创建子图 2
         ax3 = fig.add_subplot(2,2,3)                          # 创建子图 3
Out [15]:
```

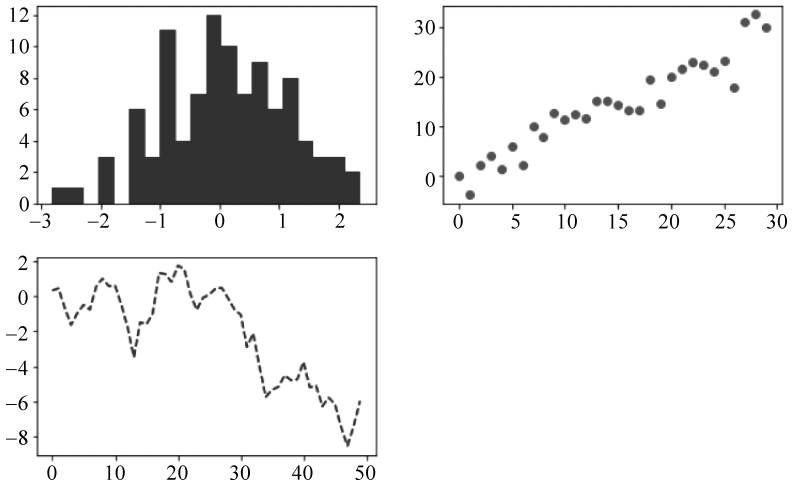


其中,(2,2,1)表示将画布内划分成 2 行 2 列绘图区中的第 1 个绘图区域。选择不同的 ax 变量,便可在对应的 subplot 子图中绘图。

【例 5-16】 在子图中绘制图形。

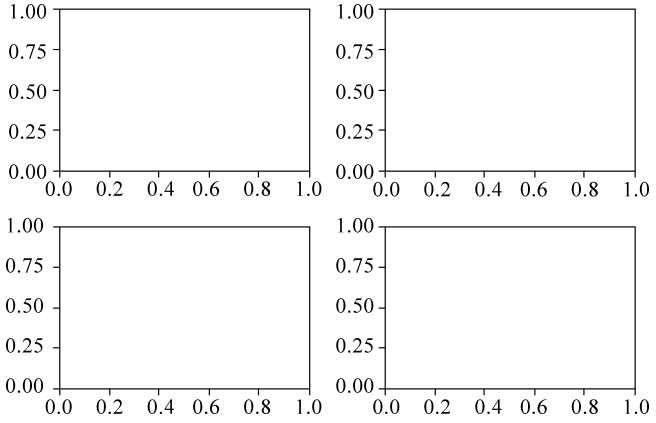
```
In [16]: fig = plt.figure(figsize = (10,6))                    # 创建画布 fig,并设置画布大小
         ax1 = fig.add_subplot(2,2,1)                          # 创建子图 1
         ax2 = fig.add_subplot(2,2,2)                          # 创建子图 2
         ax3 = fig.add_subplot(2,2,3)                          # 创建子图 3
         ax1.hist(np.random.randn(100),bins = 20,color = 'b')
         ax2.scatter(np.arange(30),np.arange(30) + 3 * np.random.randn(30))
         ax3.plot(np.random.randn(50).cumsum(),'b -- ')
         plt.show()
```

```
Out [16]:
```



也可以用 plt. subplots( ) 函数创建一张新的图片，返回包含了已生成子图对象的 NumPy 数组。例如：

```
fig,axes = plt.subplots(2, 2)
```

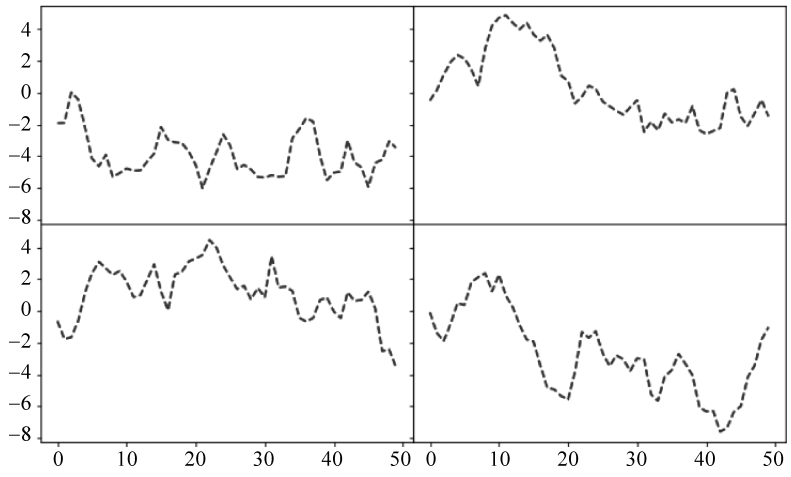


得到的 axes 数组可以像二维数组那样方便地进行索引，如 axes[0,1]。也可以通过使用 sharex 和 sharey 来表明子图分别拥有相同的 x 轴和 y 轴。

默认情况下，Matplotlib 会在子图的外部和子图之间留有一定的间距，可以使用图对象上的 subplots_adjust 方法更改间距，其中的 wspace 和 hspace 参数用于设置子图间的水平间距和垂直间距。

【例 5-17】 调整子图间距。

```
In [17]: fig,axes = plt.subplots(2, 2,sharex = True,sharey = True,figsize = (10,6))
         for i in range(2):
             for j in range(2):
                 axes[i,j].plot(np.random.randn(50).cumsum(),'b -- ')
         plt.subplots_adjust(wspace = 0,hspace = 0)              #间距缩小到零
Out [17]:
```



2. 刻度和标签

在前面介绍柱状图时已经讲解了如何设置刻度和标签，这里再详细介绍关于刻度和标签的内容。

刻度范围是绘图区域中坐标轴的取值区间，包括 x 轴和 y 轴的取值区间。刻度范围是否合适直接决定绘图区域中图形展示效果的优劣。同样地，刻度标签的样式也影响可视化效果的好坏。与刻度和标签相关的函数如表 5.10 所示。

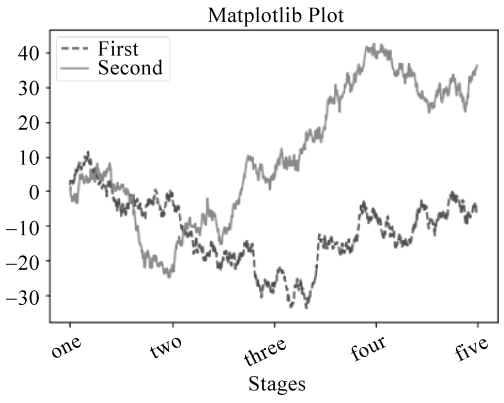
表 5.10　与刻度和标签相关的函数

函 数 示 例	函 数 说 明
plt.xlim	当前图形 x 轴的取值范围，数值区间
plt.ylim	当前图形 y 轴的取值范围，数值区间
plt.xticks	指定 x 轴刻度的数据与取值
plt.xticks	指定 y 轴刻度的数据与取值
ax1.set_xlim	子图 1 中 x 轴的取值范围，数值区间
ax1.set_ylim	子图 1 中 y 轴的取值范围，数值区间
ax1.set_xticks	子图 1 中 x 轴的刻度与取值
ax1.set_yticks	子图 1 中 y 轴的刻度与取值

【例 5-18】 设置刻度和标签。

```
In [18]: fig = plt.figure()                                          #创建画布 fig,并设置画布大小
         ax1 = fig.add_subplot(1,1,1)                                #创建子图
         ax1.plot(np.random.randn(1000).cumsum(),'--',label = 'First')       #绘制图形 1
         ax1.plot(np.random.randn(1000).cumsum(),'-',label = 'Second')       #绘制图形 2
         ticks = ax1.set_xticks([0,250,500,750,1000])
         labels = ax1.set_xticklabels(['one','two','three','four','five'],rotation = 30,fontsize = 10)
         #使用 set_xticklabels 为 x 轴设置刻度标签
         #rotation 使 x 轴刻度标签旋转 30 度
         ax1.set_title('Matplotlib Plot')                            #设置子图标题
         ax1.set_xlabel("Stages")                                    #设置 x 轴名称
         ax1.legend(loc = 'best')                                    #添加图例
```

```
        plt.show()
Out [18]:
```

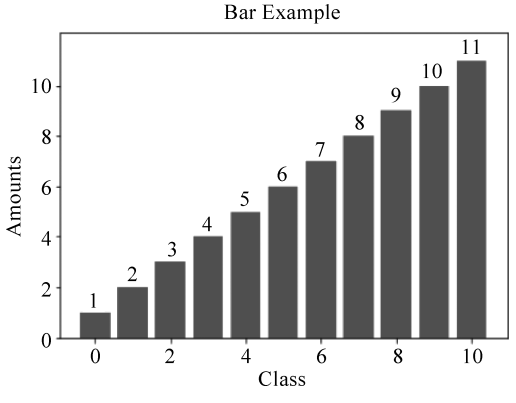


3. 注释文本

除了标准的绘图类型外，有时还需要在图表上添加文字注释，使图表能够更清晰地表达信息。使用 text()函数可以在图表给定的坐标位置(x,y)，并根据可选的样式绘制注释文本。例 5-19 以柱状图添加注释文本为例。

【例 5-19】 添加注释文本。

```
In [19]: a = np.arange(11)
         b = 1 + a
         # 绘制柱状图
         plt.bar(a,b)
         # 利用循环为每个柱形添加文本标注
         for x,y in zip(a,b):
             plt.text(x, y, str(y), ha = 'center',va = 'bottom',fontsize = 10)
             # 注释居中对齐,显示在柱子上方
         plt.xlabel('Class')
         plt.ylabel('Amounts')
         plt.title('Bar Example')
         plt.show()
Out [19]:
```



5.2 pyecharts 可视化

pyecharts 是一个用于生成 Echarts 图表的类库。ECharts 是 Enterprise Charts 的缩写,是百度开源的一个可视化 JavaScript 库,可以生成商业级数据图表。pyecharts 主要基于 Web 浏览器进行显示,图表美观而且具有交互性。能够绘制折线图、柱状图、散点图、K 线图等 30 多种常见图表,也支持多图表、组件的联动和混合展现。

5.2.1 pyecharts 的安装和使用

在使用 pyecharts 时,需要安装相应的库,安装命令如下:

```
pip install pyecharts
```

pyecharts 分为 V0.5.X 和 V1 两个版本,二者互不兼容。本书使用 V1 版本,可通过下列语句查看版本:

```
import pyecharts
print(pyecharts.__version__)
```

所有的图表类型都是按照下面的方式进行绘制:

```
chart_name = ChartType()                    # 指定具体图表类型
chart_name.add()                            # 添加数据及配置项
chart_name.render()
# render()函数会生成本地 HTML 文件,默认会在当前目录生成 render.html 文件
# 也可以传入路径参数,如 bar.render("mycharts.html")
bar.render_notebook()                       # 在 jupyter notebook 中显示图表
```

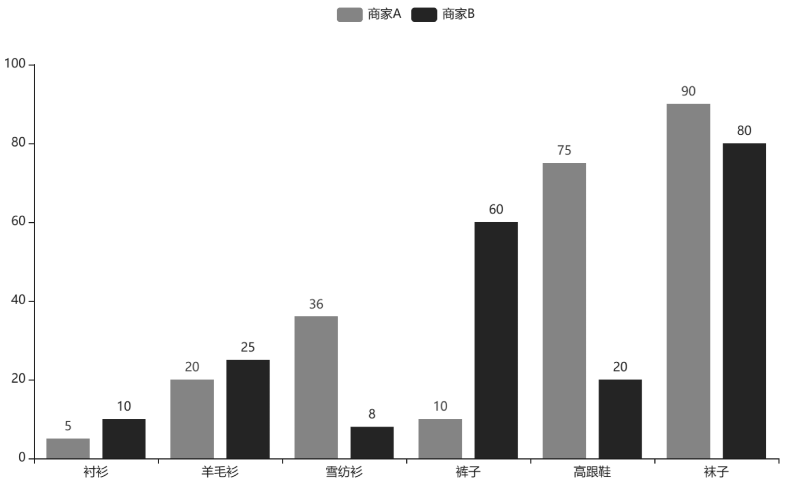
5.2.2 pyecharts 的常用图形

1. 柱状图

使用 Bar()函数可以绘制柱状图。

【例 5-20】 pyecharts 柱状图。

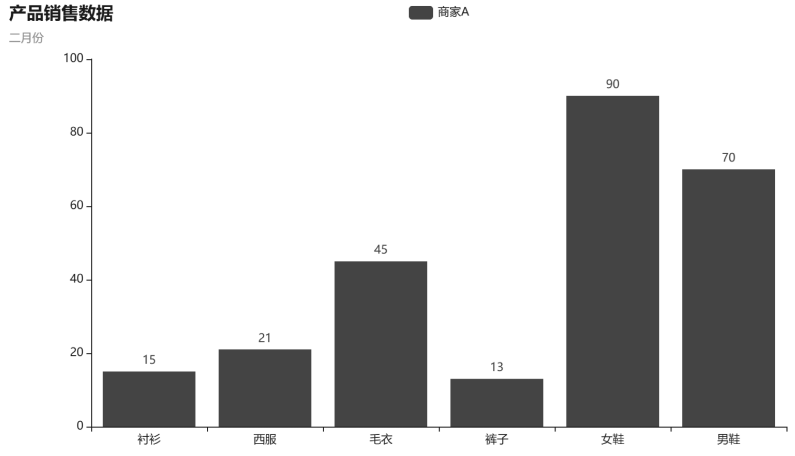
```
In [20]: from pyecharts.charts import Bar
         attr = ["衬衫", "羊毛衫", "雪纺衫", "裤子", "高跟鞋", "袜子"]
         v1 = [5, 20, 36, 10, 75, 90]
         v2 = [10, 25, 8, 60, 20, 80]
         bar = Bar()                        # 柱状图数据堆叠示例
         bar.add_xaxis(attr)                # 加入 x 轴参数
         bar.add_yaxis("商家 A",v1)         # 加入 y 轴参数
         bar.add_yaxis("商家 B",v2)
         bar.render("mycharts.html")
         bar.render_notebook()
Out [20]:
```



pyecharts 从 V1 版本支持链式调用，并且 pyecharts 绘图有两种配置项，包括全局配置项和系列配置项。

【例 5-21】 pyecharts 配置项。

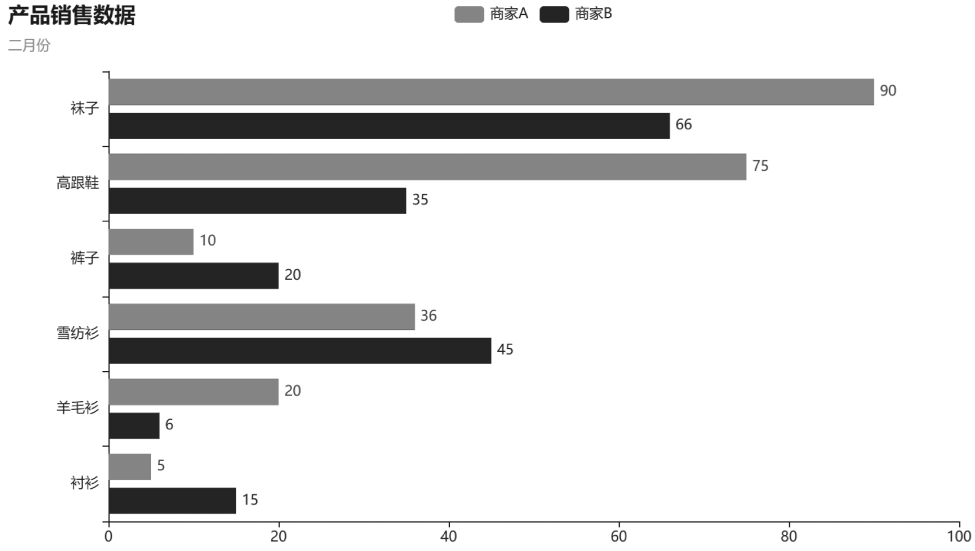
```
In [21]: from pyecharts.charts import Bar
         from pyecharts import options as opts
         #使用 options 配置项
         # V1 版本支持链式调用
         bar = (
             Bar()
             .add_xaxis(["衬衫", "西服", "毛衣", "裤子", "女鞋", "男鞋"])
             .add_yaxis("商家 A",[15, 21, 45, 13, 90, 70])
             .set_global_opts(title_opts = opts.TitleOpts(title = "产品销售数据", subtitle = "二月份"))
             #set_global_opts:全局配置项
             #title:主标题,subtitle:副标题
             #或者直接使用字典参数
             #.set_global_opts(title_opts = {"text": "产品销售数据", "subtext": "二月份"})
         )
         bar.render()
         bar.render_notebook()
Out [21]:
```



添加 bar.reversal_axis()函数可以绘制水平柱状图。

【例 5-22】 绘制水平柱状图。

```
In [22]: from pyecharts.charts import Bar
         from pyecharts import options as opts
         bar = (
            Bar()
             .add_xaxis(["衬衫", "羊毛衫", "雪纺衫", "裤子", "高跟鞋", "袜子"])
             .add_yaxis("商家 A", [5, 20, 36, 10, 75, 90])
             .add_yaxis("商家 B", [15, 6, 45, 20, 35, 66])
             .set_global_opts(title_opts = opts.TitleOpts(title = "产品销售数据", subtitle = "二
月份"))
             .set_series_opts(label_opts = opts.LabelOpts(position = "right"))
             #set_series_opts:系列配置项
             #LabelOpts:标签配置项
             #position = "right":标签在右侧
             .reversal_axis()
         )
         bar.render()
         bar.render_notebook()
Out [22]:
```



2. 饼图

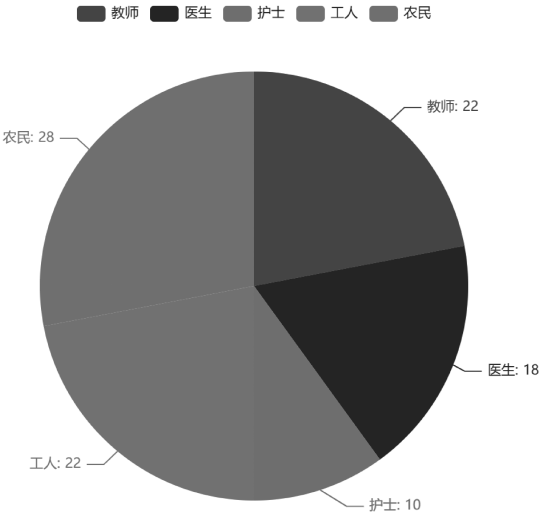
饼图用于表现不同类别的占比情况，使用 Pie()函数可以绘制饼图。

【例 5-23】 绘制饼图。

```
In [23]: from pyecharts import options as opts
         from pyecharts.charts import Pie
         labels = ['教师','医生','护士', '工人','农民']
```

```
        sizes = [22,18,10,22,28]
        c = (
            Pie()
            .add("", [list(z) for z in zip(labels, sizes)])
            .set_global_opts(title_opts = opts.TitleOpts(title = "Pie - 职业分类"))
            .set_series_opts(label_opts = opts.LabelOpts(formatter = "{b}: {c}"))
        )
        #模板变量:{a}(系列名称),{b}(数据项名称),{c}(数值), {d}(百分比)
        c.render()
        c.render_notebook()
Out [23]:
```

Pie-职业分类



在 add()方法中,可以设置如下参数。

color：系列的颜色。

radius：饼图的半径,默认为[0,75],数组的第 1 项是内半径,第 2 项是外半径。

center：饼图的中心(圆心)坐标,默认为[50,50],数组的第 1 项是横坐标,第 2 项是纵坐标。

rosetype：是否展示成南丁格尔图(玫瑰图),有 radius 和 area 两种模式。radius：扇区圆心角展现数据的百分比,半径展现数据的大小；area：所有扇区圆心角相同,仅通过半径展现数据大小。

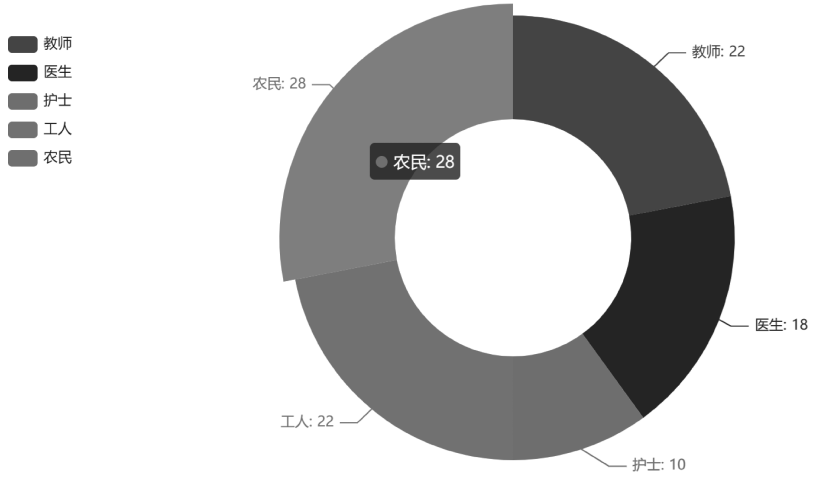
【例 5-24】 利用 radius 参数绘制环形饼图。

```
In [24]: from pyecharts import options as opts
         from pyecharts.charts import Pie
         labels = ['教师','医生','护士', '工人','农民']
         sizes = [22,18,10,22,28]
         c = (
             Pie()
             .add("",[list(z) for z in zip(labels, sizes)],radius = ["40 %", "75 %"],)
             .set_global_opts(
                 title_opts = opts.TitleOpts(title = "Pie - Radius"),
                 legend_opts = opts.LegendOpts(orient = "vertical", pos_top = "15 %", pos_left = "2 %")
             )
```

```
            #LegendOpts:图例配置项
            .set_series_opts(label_opts = opts.LabelOpts(formatter = "{b}: {c}"))
        )
    c.render()
    c.render_notebook()
```

Out [24]:

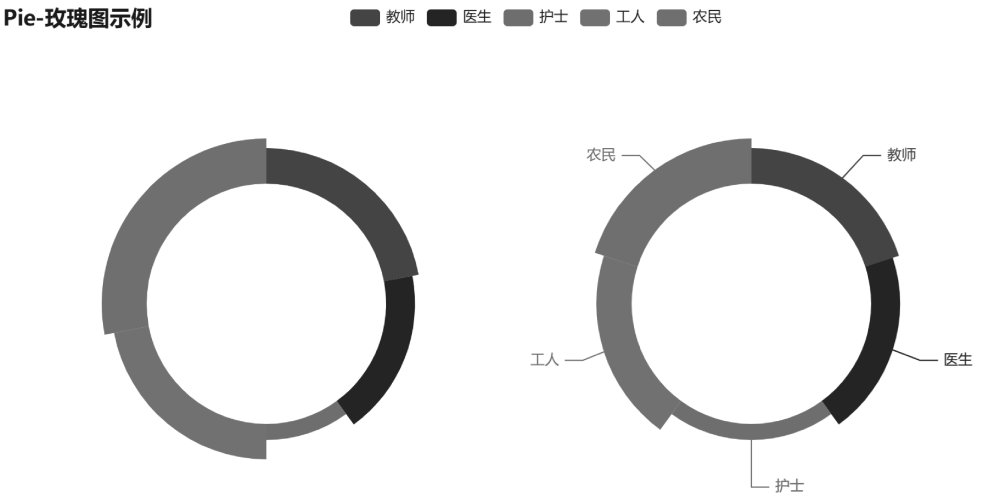
Pie-Radius



【例 5-25】 利用 rosetype 参数绘制玫瑰图。

```
In [25]: from pyecharts import options as opts
         from pyecharts.charts import Pie
         labels = ['教师','医生','护士', '工人','农民']
         sizes = [22,18,10,22,28]
         c = (
             Pie()
             .add(
                 "",
                 [list(z) for z in zip(labels, sizes)],
                 radius = ["40 %", "55 %"],
                 center = ["25 %", "50 %"],
                 rosetype = "radius",
                 label_opts = opts.LabelOpts(is_show = False)          #不显示标签
             )
             .add(
                 "",
                 [list(z) for z in zip(labels, sizes)],
                 radius = ["40 %", "55 %"],
                 center = ["70 %", "50 %"],
                 rosetype = "area",
             )
             .set_global_opts(title_opts = opts.TitleOpts(title = "Pie - 玫瑰图示例"))
         )
         c.render()
         c.render_notebook()
```

```
        c.render_notebook()
Out[25]:
```



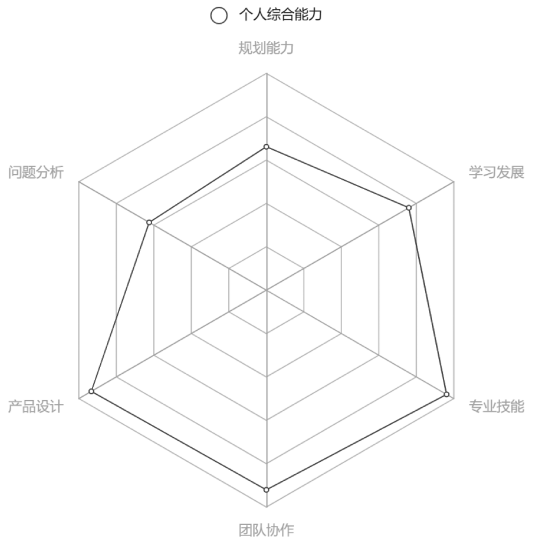
3. 雷达图

pyecharts 使用 Radar()函数绘制雷达图，其中通过 add_schema()方法对雷达图的参数与功能进行配置。

【例 5-26】 绘制雷达图。

```
In[26]: from pyecharts import options as opts
        from pyecharts.charts import Radar
        # 数据为二维数组
        v1 = [[4300, 10000, 28000, 35000, 50000, 19000]]
        randa = (
            Radar()
            .add_schema(
                 schema=[ #设置雷达指示器配置项列表
                    opts.RadarIndicatorItem(name="规划能力", max_=6500),
                    # 设置指示器名称和最大值
                    opts.RadarIndicatorItem(name="问题分析", max_=16000),
                    opts.RadarIndicatorItem(name="产品设计", max_=30000),
                    opts.RadarIndicatorItem(name="团队协作", max_=38000),
                    opts.RadarIndicatorItem(name="专业技能", max_=52000),
                    opts.RadarIndicatorItem(name="学习发展", max_=25000),
                ]
            )
            .add("个人综合能力", v1) #添加系列
            .set_series_opts(label_opts=opts.LabelOpts(is_show=False))
            .set_global_opts(
                title_opts=opts.TitleOpts(title="雷达图",pos_left='10%'),
            )
        )
        randa.render()
        randa.render_notebook()
Out[26]:
```

雷达图



4. K 线图

K 线图可以用来表示股市或期货市场中的开盘价、最高价、最低价和收盘价，反映市场的状况和价格信息。pyecharts 使用 Kline()函数绘制 K 线图。

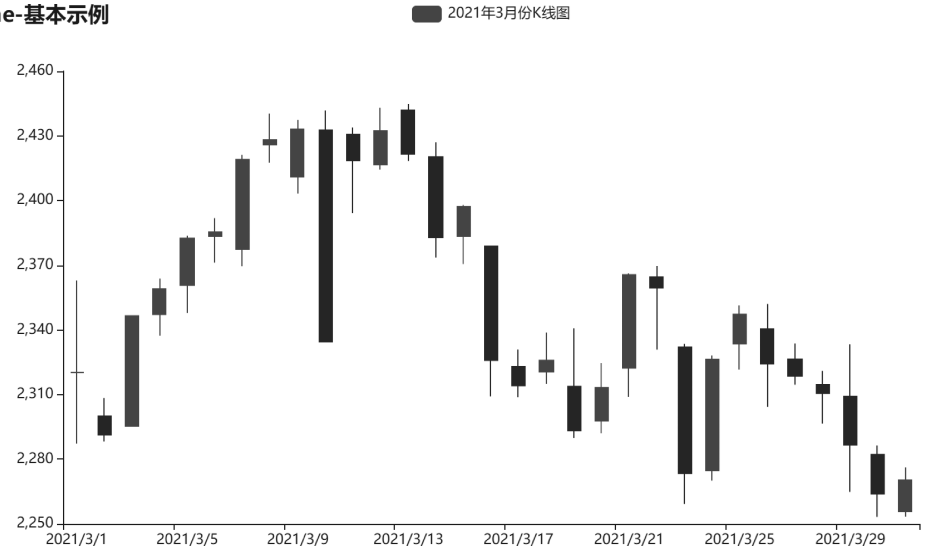
【例 5-27】 绘制 K 线图。

```
In [27]: from pyecharts import options as opts
         from pyecharts.charts import Kline
         data = [
             [2320.26, 2320.26, 2287.3, 2362.94],[2300, 2291.3, 2288.26, 2308.38],
             [2295.35, 2346.5, 2295.35, 2345.92],[2347.22, 2358.98, 2337.35, 2363.8],
             [2360.75, 2382.48, 2347.89, 2383.76],[2383.43, 2385.42, 2371.23, 2391.82],
             [2377.41, 2419.02, 2369.57, 2421.15],[2425.92, 2428.15, 2417.58, 2440.38],
             [2411, 2433.13, 2403.3, 2437.42],[2432.68, 2334.48, 2427.7, 2441.73],
             [2430.69, 2418.53, 2394.22, 2433.89],[2416.62, 2432.4, 2414.4, 2443.03],
             [2441.91, 2421.56, 2418.43, 2444.8],[2420.26, 2382.91, 2373.53, 2427.07],
             [2383.49, 2397.18, 2370.61, 2397.94],[2378.82, 2325.95, 2309.17, 2378.82],
             [2322.94, 2314.16, 2308.76, 2330.88],[2320.62, 2325.82, 2315.01, 2338.78],
             [2313.74, 2293.34, 2289.89, 2340.71],[2297.77, 2313.22, 2292.03, 2324.63],
             [2322.32, 2365.59, 2308.92, 2366.16],[2364.54, 2359.51, 2330.86, 2369.65],
             [2332.08, 2273.4, 2259.25, 2333.54],[2274.81, 2326.31, 2270.1, 2328.14],
             [2333.61, 2347.18, 2321.6, 2351.44],[2340.44, 2324.29, 2304.27, 2352.02],
             [2326.42, 2318.61, 2314.59, 2333.67],[2314.68, 2310.59, 2296.58, 2320.96],
             [2309.16, 2286.6, 2264.83, 2333.29],[2282.17, 2263.97, 2253.25, 2286.33],
             [2255.77, 2270.28, 2253.31, 2276.22],
         ]
         c = (
             Kline()
             .add_xaxis(["2021/3/{}".format(i + 1) for i in range(31)])
             .add_yaxis("2021 年 3 月份 K 线图", data)
             .set_global_opts(
                 # AxisOpts:坐标轴配置项
```

```
            # is_scale = True:坐标刻度不会强制包含零刻度
            yaxis_opts = opts.AxisOpts(is_scale = True),
            xaxis_opts = opts.AxisOpts(is_scale = True),
            title_opts = opts.TitleOpts(title = "Kline - 基本示例"),
        )
    )
    c.render()
    c.render_notebook()
```

Out [27]:

Kline-基本示例



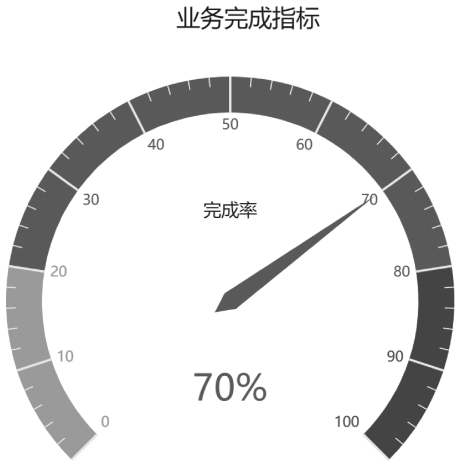
5. 仪表盘图

pyecharts 使用 Gauge()函数绘制仪表盘图。

【例 5-28】 绘制仪表盘图。

```
In [28]: from pyecharts import options as opts
    from pyecharts.charts import Gauge
    c = (
        Gauge()
        .add(
            # 系列名称,用于 tooltip 的显示
            series_name = "业务完成指标",
            #系列数据项,格式为 [(key1, value1), (key2, value2)]
            data_pair = [['完成率',70]],
            # 轮盘内数据项标签配置项
            detail_label_opts = opts.LabelOpts(position = "bottom",
              formatter = "{value} % "))
        .set_global_opts(
            #图例配置项,textstyle_opts:图例组件字体样式
            legend_opts = opts.LegendOpts(is_show = True,
              textstyle_opts = opts.TextStyleOpts(font_size = 20)),
            #提示框配置项,{a}:系列名,{b}:数据名,{c}:数据值
            tooltip_opts = opts.TooltipOpts(is_show = True,
```

```
                formatter = "{a} <br/>{b} : {c} % "),
            )
        )
        c.render()
        c.render_notebook()
Out[28]:
```



6. 词云图

词云图也叫文字云，是对文本中出现频率较高的“关键词”予以视觉化的展现。pyecharts 使用 WordCloud 绘制词云图。

【例 5-29】 绘制词云图示例一。

```
In[29]: import pyecharts.options as opts
        from pyecharts.charts import WordCloud
        data = [
            ("生活资源",1320),("供热", 1023),("供气质量", 777),
            ("生活用水", 688), ("一次供水问题", 588),("交通运输", 516),
            ("城市交通", 515),("环境保护", 483),("房地产管理", 462),
            ("城乡建设", 449),("社会保障与福利", 429),("社会保障", 407),
            ("文体与教育管理", 406),("公共安全", 406),("公交运输管理", 386),
            ("出租车运营管理", 385),("供热管理", 375),("市容环卫", 355),
            ("自然资源管理", 355),("粉尘污染", 335),("噪声污染", 324),
            ("土地资源管理", 304),("物业服务与管理", 304),("医疗卫生", 284),
            ("粉煤灰污染", 284),("占道", 284),("供热发展", 254),
            ("农村土地规划管理", 254),("生活噪声", 253),("供热单位影响", 253),
            ("城市供电", 223),("房屋质量与安全", 223),
        ]
        c = (
            WordCloud()
            #data_pair:系列数据项,[(word1, count1), (word2, count2)]
            # word_size_range:单词字体大小范围
            .add(series_name = "热点分析", data_pair = data,
                 word_size_range = [10, 66])
            .set_global_opts(
                title_opts = opts.TitleOpts(
```

```
                title = "热点分析",
                title_textstyle_opts = opts.TextStyleOpts(font_size = 23),
                pos_left = '10 % '
            ),
            tooltip_opts = opts.TooltipOpts(is_show = True)
        )
    )
c.render()
c.render_notebook()
```

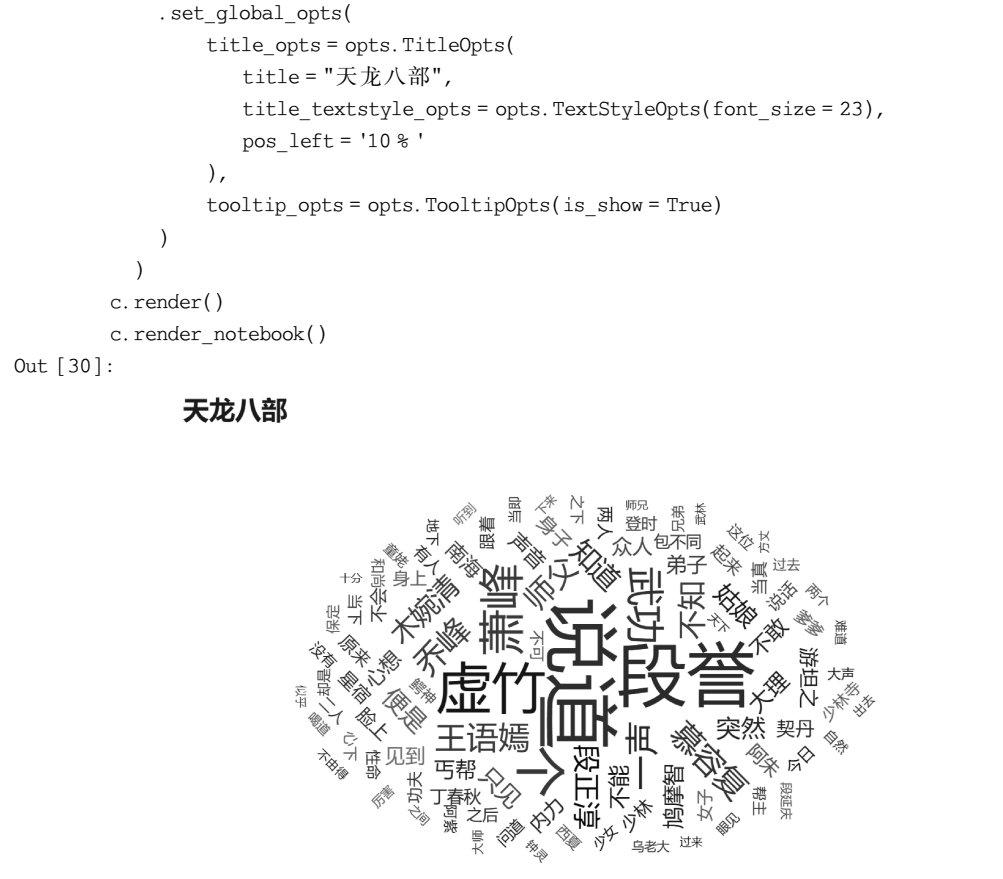
Out [29]:

下面以分析一部武侠小说为例，进一步介绍词云图的绘制。此例中引入 jieba 库用于分词。

【例 5-30】 绘制词云图示例二。

```
In [30]: import pyecharts.options as opts
         from pyecharts.charts import WordCloud
         import jieba                    # 用于分词的库
         txt = open("tianlong.txt",encoding = "utf - 8").read()
         # 加载停用词表
         stopwords = [line.strip() for line in open
                      ("StopWords.txt",encoding = 'utf - 8').readlines() ]
         words = jieba.lcut(txt) # 进行分词
         counts = {}
         for word in words:
             # 不在停用词表中
             if word not in stopwords:
                 if len(word) == 1:     # 不统计长度为 1 的词
                     continue
                 else:
                     # 累计词出现的次数
                     counts[word] = counts.get(word,0) + 1
         # 字典转换成列表,以便排序
         items = list(counts.items())
         # 按照出现的次数从大到小排序
```

```
        items.sort(key = lambda x:x[1],reverse = True)
        items = items[0:101]            #取前 100 个词
        c = (
            WordCloud()
            .add(series_name = "天龙八部", data_pair = items[0:101],
                 word_size_range = [10, 66])
            .set_global_opts(
                title_opts = opts.TitleOpts(
                   title = "天龙八部",
                   title_textstyle_opts = opts.TextStyleOpts(font_size = 23),
                   pos_left = '10 % '
                ),
                tooltip_opts = opts.TooltipOpts(is_show = True)
            )
          )
        c.render()
        c.render_notebook()
Out [30]:
```



7. 组合图表

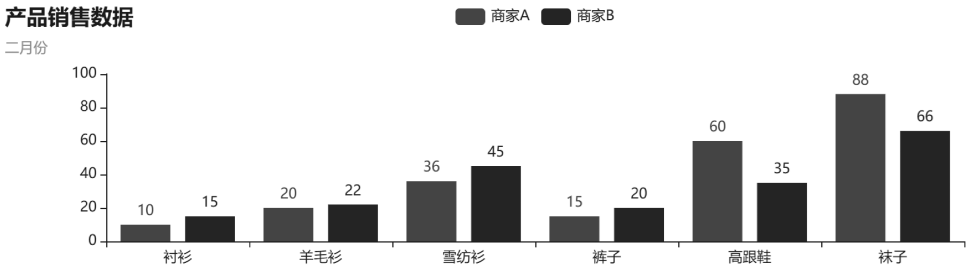
利用 pyecharts 绘制组合图表,可以垂直布局也可以水平布局。使用 Grid()函数绘制组合图表。

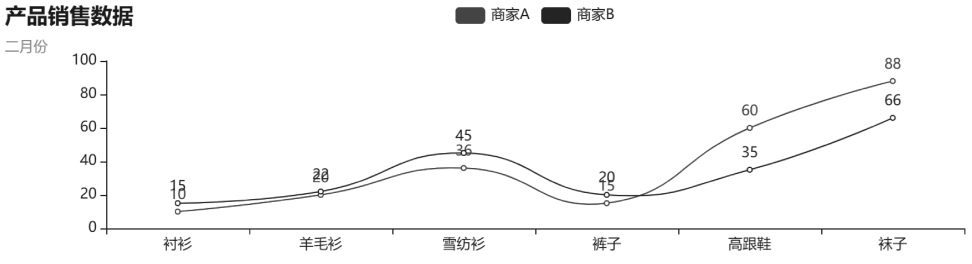
【例 5-31】 利用 pyecharts 绘制组合图表。

```
In [31]: from pyecharts.charts import Bar,Line,Grid
         from pyecharts import options as opts
         list1 = ["衬衫", "羊毛衫", "雪纺衫", "裤子", "高跟鞋", "袜子"]
         list2 = [10, 20, 36, 15, 60, 88]
         list3 = [15, 22, 45, 20, 35, 66]
         bar = (
            Bar()
             .add_xaxis(list1)
             .add_yaxis("商家 A", list2)
             .add_yaxis("商家 B",list3)
```

```
        .set_global_opts(title_opts = opts.TitleOpts(title = "产品销售数据",
                         subtitle = "二月份"))
    )
line = (
    Line()
    .add_xaxis(list1)
    .add_yaxis("商家 A", list2,is_smooth = True)
    .add_yaxis("商家 B",list3,is_smooth = True)
    .set_global_opts(
        title_opts = opts.TitleOpts(
                         title = "产品销售数据",
                         subtitle = "二月份",
                         pos_top = '50 % '),
        legend_opts = opts.LegendOpts(pos_top = '50 % '))
    )
grid = (
     Grid()
     .add(bar,grid_opts = opts.GridOpts(pos_bottom = "60 % "))
     .add(line,grid_opts = opts.GridOpts(pos_top = "60 % "))
)
grid.render_notebook()
```

Out [31]:





5.3 本章小结

本章主要介绍了 Matplotlib 和 pyecharts 两个可视化图形库。在 Matplotlib 中介绍了如何绘制折线图、柱状图、散点图、直方图和密度图，以及 Matplotlib 的自定义设置等；在 pyecharts 中介绍了如何绘制柱状图、饼图、雷达图、K 线图以及词云图等，也介绍了利用 pyecharts 绘制组合图表的方法。