第 5 章

数据分析的可视化

Matplotlib 是一个 Python 的绘图库，它以各种硬拷贝格式和跨平台的交互式环境生成出版质量级别的图形，它能够输出的图形包括折线图、散点图、直方图等。在数据可视化方面，Matplotlib 拥有数量众多的用户，其强大的绘图功能能够帮助我们对数据形成非常清晰直观的认知。

5.1 初识 Matplotlib

1. Figure(图)

Matplotlib 是一个非常优秀的 Python 2D 绘图库，只要给出符合格式的数据，通过 Matplotlib 就可以方便地制作各种高质量的图形，在任何绘图之前，我们需要一个 Figure 对象，可以理解成我们需要一张画板才能开始绘图。

```
import matplotlib.pyplot as plt
fig=plt.figure()
```

2. Axes(轴)

在拥有 Figure 对象之后，在作画前还需要 Axes(轴)，没有轴的话就没有绘图基准，所以需要添加 Axes。也可以理解成为真正可以作画的纸。

```
#图像显示中文说明
plt.rcParams['font.sans-serif'] = [u'SimHei']
fig=plt.figure()
fig=plt.figure()
ax=fig.add_subplot(111)
ax.set(xlim=[0.5,4.5],ylim=[-2,8],title='轴实例',ylabel='Y-轴',xlabel='X-轴')
plt.show()
```

以上代码实现了在一幅图上添加了一个 Axes，然后设置了这个 Axes 的 X 轴以及 Y 轴的取值范围，效果如图 5-1 所示。

上面代码中的 fig.add_subplot(111)语句是用于添加 Axes 的，参数的解释为：在画板的第 1 行第 1 列的第一个位置生成一个 Axes 对象来准备作画。也可以通过 fig.add_subplot(2，2，1)的方式生成 Axes，前面两个参数确定了面板的划分，例如 2，2 会将整个面板划分成 2 * 2 的方格，第三个参数取值范围是[1，2 * 2]，表示第几个 Axes。如下面的代码：

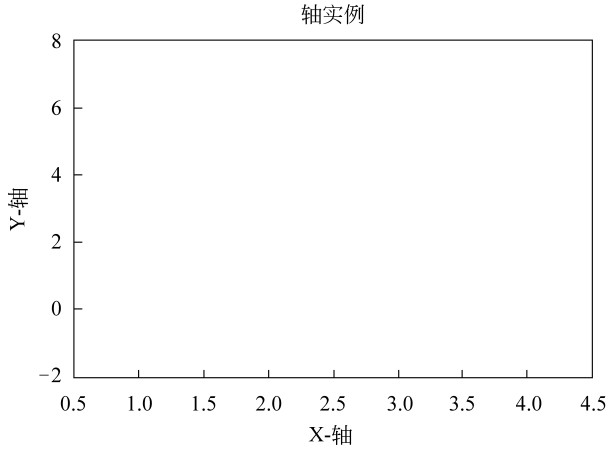

图 5-1　轴的设置

```
fig=plt.figure()
ax1=fig.add_subplot(221)
ax2=fig.add_subplot(222)
ax3=fig.add_subplot(224)
```

运行程序，得到 3 个子图，如图 5-2 所示。

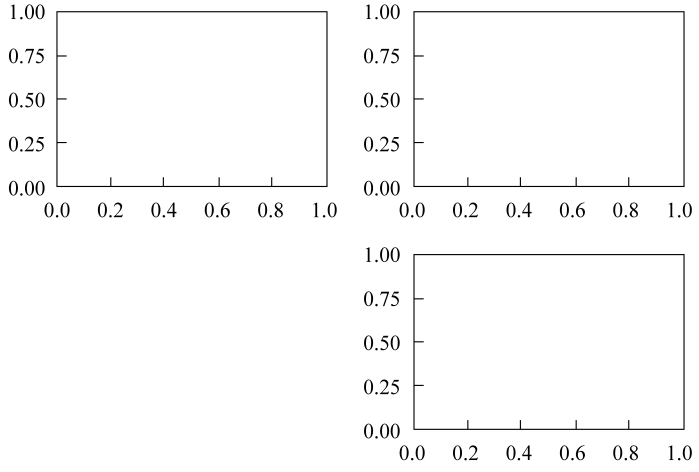

图 5-2　创建的 3 个子图

3. Multiple Axes 轴

从上面的代码中可以发现我们上面添加 Axes 似乎不够简洁，所以提供了下面的方式一次性生成所有的 Axes：

```
fig,axes=plt.subplot(nrows=2,ncols=2)
axes[0,0].set(title='左上限')
axes[0,1].set(title='右上限')
axes[1,0].set(title='左下限')
axes[1,1].set(title='右下限')
```

fig 还是我们熟悉的画板，axes 成了我们常用二维数组的形式访问，这在循环绘图时比较好用。

4. Axes 与.pyplot

相信不少人看过下面的代码，很简单并易懂，但是下面的作画方式只适合简单的绘图，快速地将图绘出。在处理复杂的绘图工作时，我们还是需要使用 Axes 来完成作画的。

```
plt.plot([1,4,7,9],[10,18,25,36],color='lightred',linewidth=3)
plt.xlim(0.5,5.0)
plt.show()
```

5.2 基本二维绘图

5.2.1 折线图

Matplotlib 的用法非常简单，对于最简单的折线图来说，程序只需根据需要给出对应的 X 轴、Y 轴数据。调用 pyplot 子模块下的 plot()函数即可生成简单的折线图。

【例 5-1】 分析某教材从 2012 年到 2018 年的销售数据，此时可考虑将年份作为 X 轴数据，将图书各年份的销量作为 Y 轴数据。程序只要将 2012—2018 年定义成 list 列表作为 X 轴数据，并将对应年份的销量作为 Y 轴数据即可。

如使用如下简单程序来展示从 2012 年到 2018 年某教材的销售数据。

```
import matplotlib.pyplot as plt
#定义 2 个列表分别作为 X 轴、Y 轴数据
x_data = ['2012', '2012', '2013', '2014', '2015', '2016', '2018']
y_data = [ 60200, 63000, 71000, 84000, 90500, 107000,98300]
#第一个列表代表横坐标的值,第二个代表纵坐标的值
plt.plot(x_data, y_data)
#调用 show()函数显示图形
plt.show()
```

运行程序，效果如图 5-3 所示。

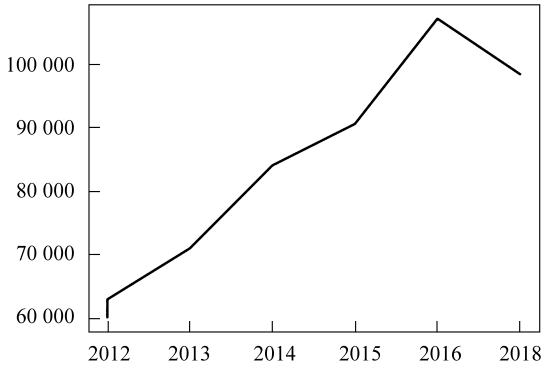

图 5-3 简单折线图

如果在调用 plot()函数时只传入一个 list 列表，该 list 列表的数据将作为 Y 轴数据，那么 Matplotlib 会自动使用 0、1、2、3 作为 X 轴数据。例如，修改以下代码：

```
plt.plot(y_data)
```

运行程序，效果如图 5-4 所示。

plot()函数除了支持创建具有单条折线的折线图外，还支持创建包含多条折线的复式折线图——只要在调用 plot()函数时传入多个分别代表 X 轴和 Y 轴数据的 list 列表即可。例

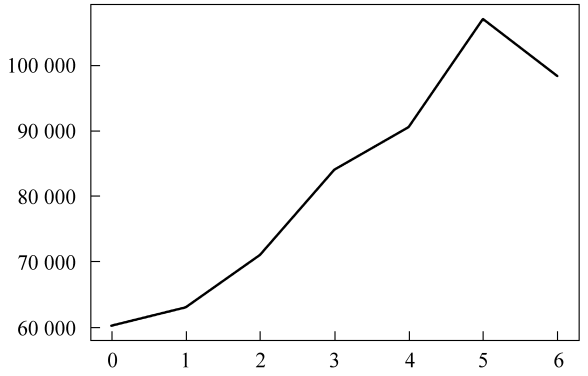

图 5-4　使用默认的 X 轴数据

如以下代码：

```
import matplotlib.pyplot as plt
x_data = ['2012', '2012', '2013', '2014', '2015', '2016', '2018']
#定义 2 个列表分别作为两条折线的 Y 轴数据
y_data = [ 60200, 63000, 71000, 84000, 90500, 107000,98300]
y_data2 = [52000, 54200, 51500,58300, 56800, 59500, 62700]
#传入 2 组数据分别代表 X 轴、Y 轴的数据
plt.plot(x_data, y_data, x_data, y_data2)
#调用 show()函数显示图形
plt.show()
```

在以上代码中，调用 plot()函数时，传入了两组分别代表 X 轴数据、Y 轴数据的 list 列表，因此该程序可以显示两条折线，效果如图 5-5 所示。

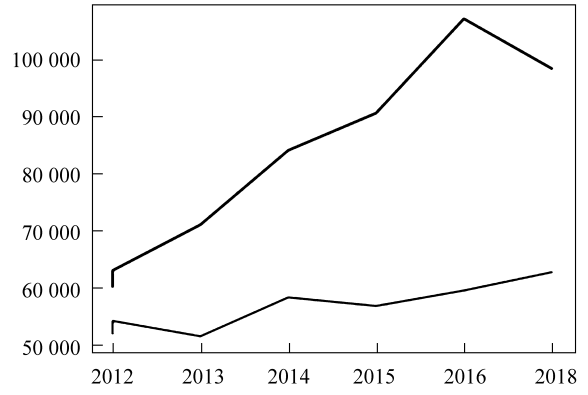

图 5-5　包含多条折线的复式折线图

也可以通过多次调用 plot()函数来生成多条折线。例如，将上面程序中的 plt.plot(x_data，y_data，x_data，y_data2)代码改为如下两行代码，程序同样会生成包含两条折线的复式折线图。

```
plt.plot(x_data, y_data)
plt.plot(x_data, y_data2)
```

在调用 plot()函数时还可以传入额外的参数来指定折线的样式，如线宽、颜色、样式等。例如：

```
import matplotlib.pyplot as plt
x_data = ['2012', '2012', '2013', '2014', '2015', '2016', '2018']
```

```
#定义 2 个列表分别作为两条折线的 Y 轴数据
y_data = [ 60200, 63000, 71000, 84000, 90500, 107000,98300]
y_data2 = [52000, 54200, 51500,58300, 56800, 59500, 62700]
#指定折线的颜色、线宽和样式
plt.plot(x_data, y_data, color = 'red', linewidth = 2.0, linestyle = '-.')
plt.plot(x_data, y_data2, color = 'blue', linewidth = 3.0, linestyle = '--')
#调用 show()函数显示图形
plt.show()
```

代码中，用 color 指定折线的颜色，linewidth 指定线宽，linestyle 指定折线样式。

在使用 linestyle 指定折线样式时，该参数支持如下字符串参数值。

- -：代表实线，这是默认值。
- --：代表虚线。
- ：：代表点线。
- -.：代表短线、点相间的虚线。

运行以上程序，效果如图 5-6 所示。

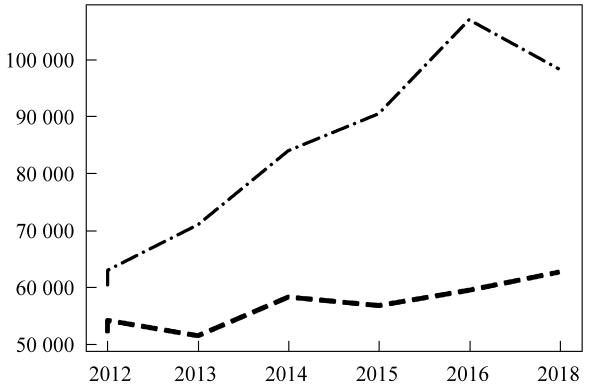

图 5-6　设置了折线图的线型

5.2.2　散点图

在 Matplotlib 中使用函数 matplotlib.pyplot.scatter()绘制散点图，matplotlib.pyplot.scatter 的函数格式如下：

matplotlib.pyplot.scatter(x，y，s＝None，c＝None，marker＝None，cmap＝None，norm＝None，vmin＝None，vmax＝None，alpha＝None，linewidths＝None，verts＝None，edgecolors＝None，hold＝None，data＝None，**kwargs)：参数 x,y 组成了散点的坐标；s 为散点的面积；c 为散点的颜色(默认为蓝色'b')；marker 为散点的标记；alpha 为散点的透明度(0 与 1 之间的数，0 为完全透明，1 为完全不透明)；linewidths 为散点边缘的线宽；如果 marker 为 None，则使用 verts 的值构建散点标记；edgecolors 为散点边缘颜色。

其他参数如 cmap 为 colormap；norm 为数据亮度；vmin、vmax 和 norm 配合使用用来归一化亮度数据，这些都与数据亮度有关。

【例 5-2】 绘制散点图。

```
#绘制普通散点图,如图 5-7 所示
import matplotlib
import matplotlib.pyplot as plt
import numpy as np
```

```
#保证图片在浏览器内正常显示
%matplotlib inline
#10 个点
N = 10
x = np.random.rand(N)
y = np.random.rand(N)
plt.scatter(x, y)
plt.show()
```

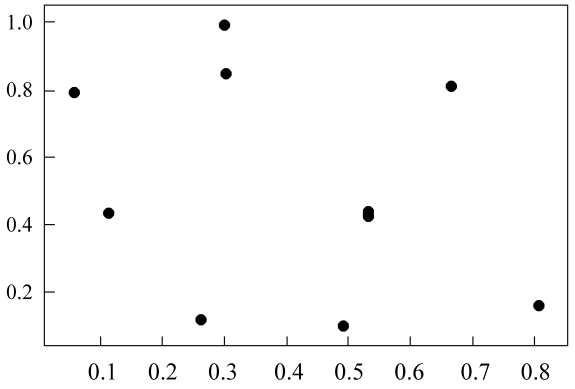

图 5-7　普通散点图

可以通过设置 scatter()函数的相关属性，即可更改散点的大小，代码如下：

```
#每个点随机大小，效果如图 5-8 所示
s = (30 * np.random.rand(N))**2
plt.scatter(x, y, s=s)
plt.show()
```

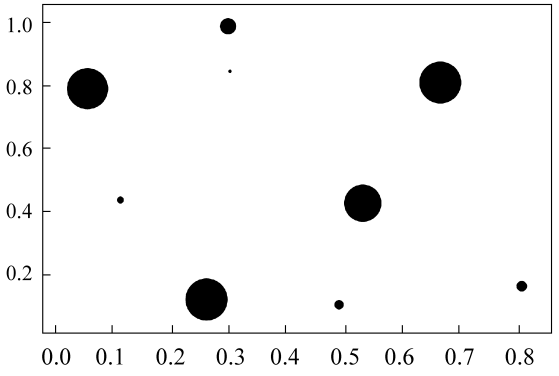

图 5-8　随机更改散点的大小

还可以通过设置参数 c 与 alpha 来更改散点颜色和透明度，代码为：

```
#随机颜色，如图 5-9 所示
c = np.random.rand(N)
plt.scatter(x, y, s=s, c=c, alpha=0.5)
plt.show()
```

设置 scatter()函数的' marker '参数，可更改散点形状，代码为：

```
plt.scatter(x, y, s=s, c=c, marker='^', alpha=0.5)
plt.show()  %效果如图 5-10 所示
```

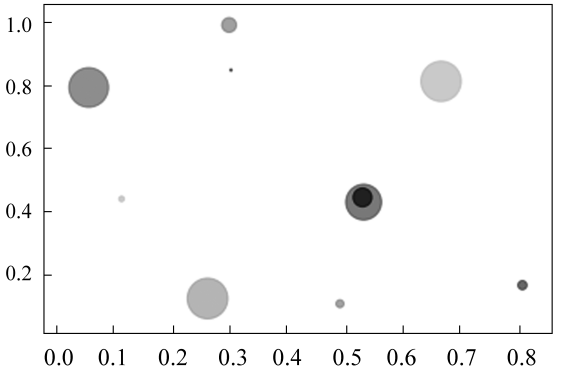

图 5-9　更改散点颜色与透明度

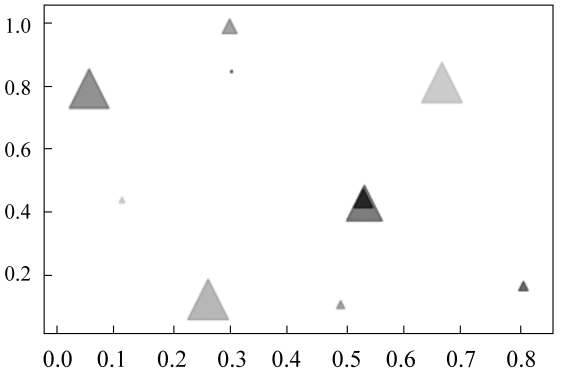

图 5-10　更改散点形状

还可以在一张图上绘制两组数据的散点，实现代码为：

```
#10 个点
N = 10
x1 = np.random.rand(N)
y1 = np.random.rand(N)
x2 = np.random.rand(N)
y2 = np.random.rand(N)
plt.scatter(x1, y1, marker='o')
plt.scatter(x2, y2, marker='^')
plt.show()   %效果如图 5-11 所示
```

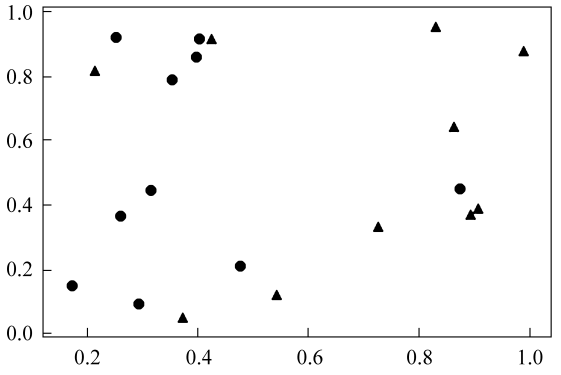

图 5-11　同一张图绘制两组散点图

调用 legend()函数，可为图像设置图例，例如：

```
plt.scatter(x1, y1, marker='o', label="圆形")
plt.scatter(x2, y2, marker='^', label="三角形")
plt.legend(loc='best')
plt.show()                                          #效果如图 5-12 所示
```

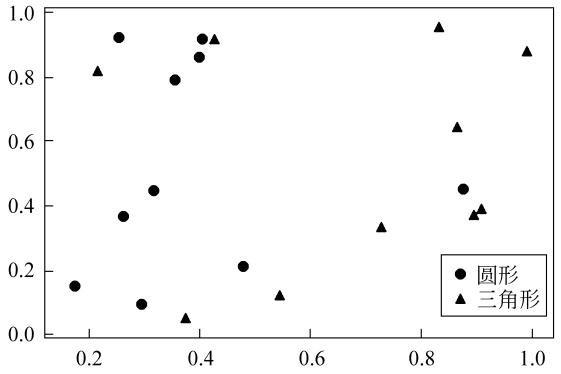

图 5-12　为图像设置图例

5.2.3　条形图

条形图或柱形图是一种图表或图形，它显示带有矩形条的分类数据，其高度或长度与它们所代表的值成比例。可以垂直或水平绘制条形。条形图显示了离散类别之间的比较。图表的一个轴显示要比较的特定类别，另一个轴表示测量值。

Matplotlib API 提供了 bar()函数，可以在 MATLAB 样式中以及面向对象的 API 中使用。与 axis 对象一起使用的 bar()函数使用大小为(x －width ＝ 2；x ＋ width＝2；bottom；bottom ＋ height)来绑定矩形创建条形图，其格式如下：

ax.bar(x，height，width，bottom，align)：其中 x 表示条形的 x 坐标的标量序列。如果 x 是条形中心(默认)或左边缘，则对齐控件。height 是标量或标量序列，表示条的高度。width 是标量或类似数组，可选，条形的宽度默认为 0.8。bottom 是标量或类似数组，可选，条形的 y 坐标默认为 None。align 的可选值为'center'或'edge'，默认值为 center。

【例 5-3】　显示一所学院提供的各种课程的学生人数。

```
import matplotlib.pyplot as plt
import numpy as np
import math
fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
langs = ['C', 'C++', 'Java', 'Python', 'PHP']
students = [22,18,34,28,14]
ax.bar(langs,students)
plt.show()
```

运行程序，效果如图 5-13 所示。

我们可以通过使用条形的厚度和位置来绘制多个条形图。数据变量包含三个系列的四个值。以下代码将显示四个条形图中的三个。这些条的厚度为 0.35 个单位。每个条形图将从前一个移动 0.5 个单位。数据对象是一个多元图，包含过去 4 年在工程学院的三个分支中通过的学生数量。

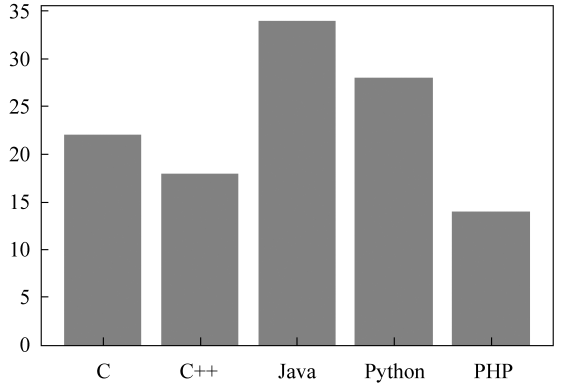

图 5-13　条形图

```
data = [[30, 25, 50, 20],
[40, 23, 51, 17],
[35, 22, 45, 19]]
X = np.arange(4)
fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
ax.bar(X + 0.00, data[0], color = 'b', width = 0.25)
ax.bar(X + 0.35, data[1], color = 'g', width = 0.25)
ax.bar(X + 0.50, data[2], color = 'r', width = 0.25)
plt.show()
```

运行程序,效果如图 5-14 所示。

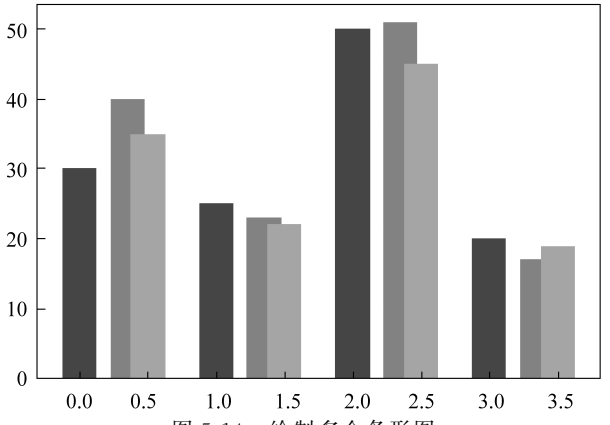

图 5-14　绘制多个条形图

堆积条形图是堆叠表示彼此顶部的不同组的条形图。结果为条形图的高度显示组的组合结果。

pyplot.bar()函数的可选 bottom 参数指定条的起始值。它不是从零运行到一个值,而是图像从底部到顶的值。第一次调用 pyplot.bar()绘制蓝色条形图。第二次调用 pyplot.bar()绘制红色条形图,蓝色条形图的底部位于红色条形图的顶部。

```
N = 5
menMeans = (21, 36, 30, 36, 28)
womenMeans = (25, 32, 34, 20, 25)
ind = np.arange(N)                                    #组 x 的位置
width = 0.35
```

```
fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
ax.bar(ind, menMeans, width, color='r')
ax.bar(ind, womenMeans, width,bottom=menMeans, color='b')
ax.set_ylabel('分数')
ax.set_title('按组和性别分数')
ax.set_xticks(ind, ('G1', 'G2', 'G3', 'G4', 'G5'))
ax.set_yticks(np.arange(0, 81, 10))
ax.legend(labels=['男', '女'])
plt.show()
```

运行程序，效果如图 5-15 所示。

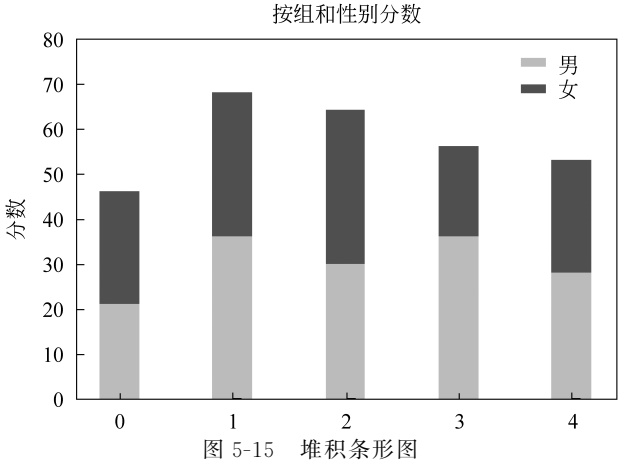

图 5-15　堆积条形图

5.2.4　饼图

饼图广泛地应用于各个领域，用于表示不同分类的占比情况，通过弧度大小来对比各种分类。饼图通过将一个圆饼按照分类的占比划分成多个区块，整个圆饼代表数据的总量，每个区块（圆弧）表示该分类占总体的比例大小，所有区块（圆弧）的加和等于 100%。

在 Matplotlib 中，利用 pie()函数绘制饼图。函数的格式为：

matplotlib.pyplot.pie(x，explode＝None，labels＝None，colors＝None，autopct＝None，labeldistance＝1.1，pctdistance＝0.6，shadow＝False，radius＝1，startangle＝0，counterclock＝True，wedgeprops＝None，textprops＝None，center＝0，0，frame＝False，rotatelabels＝False，＊，normalize＝None，data＝None)：其中 x 是浮点型数组，表示每个扇形的面积。explode 是数组，表示各个扇形之间的间隔，默认值为 0。labels 是列表，表示各个扇形的标签，默认值为 None。colors 是数组，表示各个扇形的颜色，默认值为 None。autopct 设置饼图内各个扇形的百分比显示格式，%d%%为整数百分比，%0.1f 为一位小数，%0.1f%%为一位小数百分比，%0.2f%%为两位小数百分比。labeldistance 表示标签标记的绘制位置相对于半径的比例，默认值为 1.1，如小于 1 则绘制在饼图内侧。pctdistance 类似于 labeldistance，指定 autopct 的位置刻度，默认值为 0.6。shadow 的值为布尔值 True 或 False，设置饼图的阴影，默认为 False，不设置阴影。radius 设置饼图的半径，默认为 1。startangle 表示起始绘制饼图的角度，默认为从 x 轴正方向逆时针画起，如设定 startangle＝90 则从 y 轴正方向画起。counterclock 是布尔值，设置指针方向，默认为 True，即逆时针，False 表示顺时针。wedgeprops 是字典类型，默认值为 None，参数字典传递给 wedge 对象用来画一个饼图，例如