第 1 章

数据可视化

机器学习中许多时候需要将数据可视化，方便更直观地表现目前的数据，所以本书先介绍数据图形的绘制，所使用的工具是 matplotlib 绘图库模块，使用前需先安装：

```
pip install matplotlib
```

matplotlib 是一个庞大的绘图库模块，本章我们只导入其中的 pyplot 子模块就可以完成许多图表绘制，如下所示，未来就可以使用 plt 调用相关的方法。

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

本章将叙述 matplotlib 的重点内容，完整使用说明可以参考 matplotlib 的官方网站。

1-1 认识 matplotlib.pyplot 模块的主要函数

下列是绘制图表的常用函数。

函数名称	说明
plot(系列数据)	绘制折线图
scatter(系列数据)	绘制散点图
hist(系列数据)	绘制直方图

下列是坐标轴设定的常用函数。

函数名称	说明
title(标题)	设定坐标轴的标题
axis( )	可以设定坐标轴的最小和最大刻度范围
xlim(x_Min, x_Max)	设定 x 轴的刻度范围
ylim(y_Min, y_Max)	设定 y 轴的刻度范围
label(名称)	设定图表标签图例
xlabel(名称)	设定 x 轴的名称
ylabel(名称)	设定 y 轴的名称
xticks(刻度值)	设定 x 轴刻度值
yticks(刻度值)	设定 y 轴刻度值
tick_params( )	设定坐标轴的刻度大小、颜色
legend( )	设定坐标的图例
text( )	在坐标轴指定位置输出字符串
grid( )	图表增加网格线
show( )	显示图表
cla( )	清除图表

下列是图片的读取与储存的函数。

函数名称	说明
imread(文件名)	读取图片文件
savefig(文件名)	将图片存入文件

1-2　绘制简单的折线图 plot( )

这一节将从最简单的折线图开始解说，常用语法格式如下：

```
plot(x, y, lw=x, ls='x', label='xxx', color)
```

x：*x* 轴系列值，如果省略系列自动标记 0，1，…，可参考 1-2-1 节。
y：*y* 轴系列值，可参考 1-2-1 节。
lw：linewidth 的缩写，折线图的线条宽度，可参考 1-2-2 节。
ls：linestyle 的缩写，折线图的线条样式，可参考 1-2-6 节。
color：缩写是 c，可以设定色彩，可参考 1-2-6 节。
label：图表的标签，可参考 1-2-8 节。

1-2-1　画线基础实践

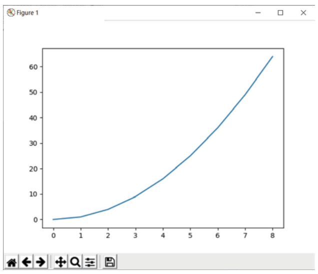
将含数据的列表当作参数传给 plot( )，列表内的数据会被视为 *y* 轴的值，*x* 轴的值会依列表值的索引位置自动产生。

程序实例 ch1_1.py：绘制折线，square[ ] 列表有 9 笔数据代表 *y* 轴值，数据基本上是 *x* 轴索引 0 ～ 8 的平方值序列，这个实例使用列表生成式建立 *x* 轴数据。

```python
# ch1_1.py
import matplotlib.pyplot as plt

x = [x for x in range(9)]        # 产生0, 1, ... 8列表
squares = [0, 1, 4, 9, 16, 25, 36, 49, 64]
plt.plot(x, squares)             # 列表squares数据是y轴的值
plt.show()
```

执行结果



在绘制线条时，预设颜色是蓝色，更多相关设定 1-2-6 节会讲解。如果 *x* 轴的数据是 0，1，…，*n* 时，在使用 plot( ) 时我们可以省略 *x* 轴数据，可以参考下列程序实例。

程序实例 ch1_2.py：重新设计 ch1_1.py，此实例省略 *x* 轴数据。

```python
# ch1_2.py
import matplotlib.pyplot as plt

squares = [0, 1, 4, 9, 16, 25, 36, 49, 64]
plt.plot(squares)         # 列表squares数据是y轴的值
plt.show()
```

执行结果

与 ch1_1.py 相同。

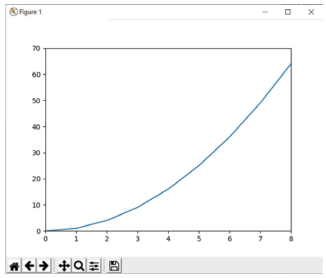
从上述执行结果可以看到左下角的轴刻度不是 (0,0)，我们可以使用 axis( ) 设定 *x*、*y* 轴的最小和最大刻度。

程序实例 ch1_3.py：重新设计 ch1_2.py，将 x 轴刻度设为 0 ～ 8，y 轴刻度设为 0 ～ 70。

```
# ch1_3.py
import matplotlib.pyplot as plt

squares = [0, 1, 4, 9, 16, 25, 36, 49, 64]
plt.plot(squares)          # 列表squares数据是y轴的值
plt.axis([0, 8, 0, 70])    # x轴刻度0～8, y轴刻度0～70
plt.show()
```

执行结果

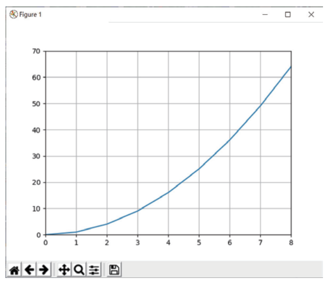


在做数据分析时，有时候会想要在图表内增加网格线，这可以让图表中 x 轴值对应的 y 轴值更加清楚，可以使用 grid( ) 函数。

程序实例 ch1_3_1.py：增加网格线重新设计 ch1_3.py，此程序重点是第 7 行。

```
1  # ch1_3_1.py
2  import matplotlib.pyplot as plt
3
4  squares = [0, 1, 4, 9, 16, 25, 36, 49, 64]
5  plt.plot(squares)              # 列表squares数据是y轴的值
6  plt.axis([0, 8, 0, 70])        # x轴刻度0-8, y轴刻度0-70
7  plt.grid()
8  plt.show()
```

执行结果



1-2-2　线条宽度 linewidth

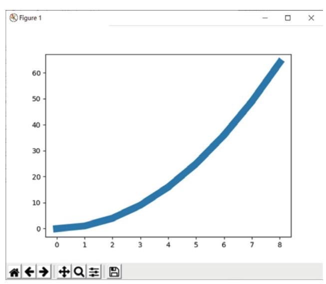
使用 plot( ) 时预设线条宽度是 1，可以多加一个 linewidth（缩写是 lw）参数设定线条的粗细。

程序实例 ch1_4.py：设定线条宽度是 10，使用 lw=10。

```
# ch1_4.py
import matplotlib.pyplot as plt

squares = [0, 1, 4, 9, 16, 25, 36, 49, 64]
plt.plot(squares, lw=10)
plt.show()
```

执行结果



1-2-3　标题的显示

目前 matplotlib 模块默认不支持中文显示，笔者将在 1-5 节讲解如何让图表显示中文，下列是几个显示标题的重要方法。

```
title( 标题名称 , fontsize= 字号 )          #  图表标题
xlabel( 标题名称 , fontsize= 字号 )         #  x 轴标题
ylabel( 标题名称 , fontsize= 字号 )         #  y 轴标题
```

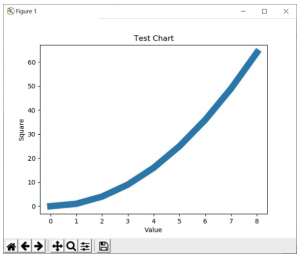
上述方法默认字号大小是 12，但是可以使用 fontsize 参数更改字号。

程序实例 ch1_5.py：使用默认字号为图表与 x、y 轴建立标题。

```python
# ch1_5.py
import matplotlib.pyplot as plt

squares = [0, 1, 4, 9, 16, 25, 36, 49, 64]
plt.plot(squares, lw=10)
plt.title('Test Chart')
plt.xlabel('Value')
plt.ylabel('Square')
plt.show()
```

执行结果

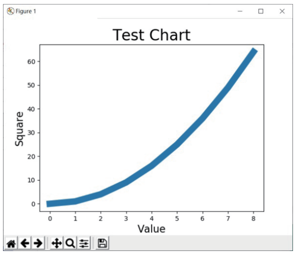


程序实例 ch1_6.py：使用设定字号 24 建立图表标题，字号 16 建立 x、y 轴标题。

```python
# ch1_6.py
import matplotlib.pyplot as plt

squares = [0, 1, 4, 9, 16, 25, 36, 49, 64]
plt.plot(squares, lw=10)
plt.title('Test Chart', fontsize=24)
plt.xlabel('Value', fontsize=16)
plt.ylabel('Square', fontsize=16)
plt.show()
```

执行结果



1-2-4　坐标轴刻度的设定

在设计图表时可以使用 tick_params( ) 设计设定坐标轴的刻度大小、颜色以及应用范围。

```
tick_params(axis= 'xx' , labelsize=xx, color= 'xx' )
# labelsize 的 xx 代表刻度大小
```

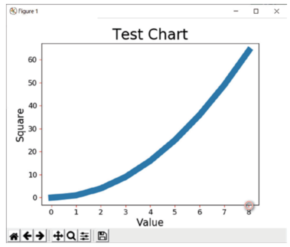
如果 axis 的 xx 是 both，代表应用到 x 轴和 y 轴；如果 xx 是 x，代表应用到 x 轴；如果 xx 是 y，代表应用到 y 轴。color 则是设定刻度的线条颜色，例如：red 代表红色，1-2-6 节将有颜色表。

程序实例 ch1_7.py：使用不同刻度与颜色绘制图表。

```
# ch1_7.py
import matplotlib.pyplot as plt

squares = [0, 1, 4, 9, 16, 25, 36, 49, 64]
plt.plot(squares, lw=10)
plt.title('Test Chart', fontsize=24)
plt.xlabel('Value', fontsize=16)
plt.ylabel('Square', fontsize=16)
plt.tick_params(axis='both', labelsize=12, color='red')
plt.show()
```

执行结果



1-2-5 多组数据的应用

目前所有的图表皆是只有一组数据，其实可以扩充多组数据，只要在 plot( ) 内增加数据列表参数即可。此时 plot( ) 的参数如下：

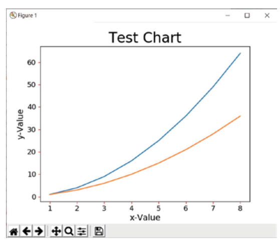
```
plot(seq, 第一组数据, seq, 第二组数据, … )
```

程序实例 ch1_8：设计含多组数据的图表。

```
# ch1_8.py
import matplotlib.pyplot as plt

data1 = [1, 4, 9, 16, 25, 36, 49, 64]   # data1线条
data2 = [1, 3, 6, 10, 15, 21, 28, 36]   # data2线条
seq = [1,2,3,4,5,6,7,8]
plt.plot(seq, data1, seq, data2)        # data1&2线条
plt.title("Test Chart", fontsize=24)
plt.xlabel("x-Value", fontsize=14)
plt.ylabel("y-Value", fontsize=14)
plt.tick_params(axis='both', labelsize=12, color='red')
plt.show()
```

执行结果



上述以不同颜色显示线条是系统默认，我们也可以自定义线条色彩。

1-2-6 线条色彩与样式

如果想设定线条色彩，可以在 plot( ) 内增加下列 color 颜色参数设定，下列是常见的色彩。

色彩字符	色彩说明
‘b’	blue(蓝色)
‘c’	cyan(青色)
‘g’	green(绿色)
‘k’	black(黑色)
‘m’	magenta(品红)
‘r’	red(红色)

续表

色彩字符	色彩说明
‘w’	white(白色)
‘y’	yellow(黄色)

下列是常见的样式。

字符	说明
‘-’ 或 ‘solid’	实线
‘- -’ 或 ‘dashed’	虚线
‘-.’ 或 ‘dashdot’	虚点线
‘:’ 或 ‘dotted’	点线
‘.’	点标记
‘,’	像素标记
‘o’	圆标记
‘v’	反三角标记
‘^’	三角标记
‘<’	左三角形
‘>’	右三角形
‘s’	方形标记
‘p’	五角标记
‘*’	星星标记
‘+’	加号标记
‘-’	减号标记
‘x’	X 标记
‘H’	六边形 1 标记
‘h’	六边形 2 标记

上述可以混合使用，例如 ‘r-.’ 代表红色虚点线。

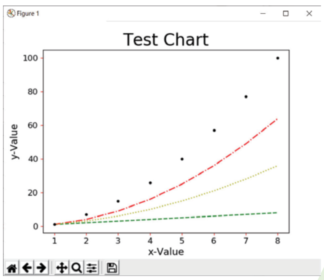
程序实例 ch1_9.py：采用不同色彩与线条样式绘制图表。

```python
# ch1_9.py
import matplotlib.pyplot as plt

data1 = [1, 2, 3, 4, 5, 6, 7, 8]                    # data1线条
data2 = [1, 4, 9, 16, 25, 36, 49, 64]               # data2线条
data3 = [1, 3, 6, 10, 15, 21, 28, 36]               # data3线条
data4 = [1, 7, 15, 26, 40, 57, 77, 100]             # data4线条

seq = [1, 2, 3, 4, 5, 6, 7, 8]
plt.plot(seq, data1, 'g--', seq, data2, 'r-.', seq, data3, 'y:', seq, data4, 'k.')
plt.title("Test Chart", fontsize=24)
plt.xlabel("x-Value", fontsize=14)
plt.ylabel("y-Value", fontsize=14)
plt.tick_params(axis='both', labelsize=12, color='red')
plt.show()
```

执行结果

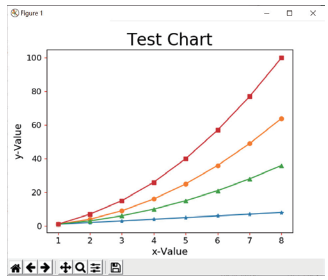


上述第 10 行最右边的‘k.’代表绘制黑点而不是绘制线条，读者也可以使用不同颜色绘制散点图，1-3 节也会介绍另一个方法 scatter( ) 绘制散点图。上述格式应用是很灵活的，如果我们使用‘-*’可以绘制线条，同时在指定点加上星星标记。注：如果没有设定颜色，系统会自行配置颜色。

程序实例 ch1_10.py：重新设计 ch1_9.py 绘制线条，同时为各个点加上标记，程序重点是第 10 行。

```
1  # ch1_10.py
2  import matplotlib.pyplot as plt
3
4  data1 = [1, 2, 3, 4, 5, 6, 7, 8]              # data1线条
5  data2 = [1, 4, 9, 16, 25, 36, 49, 64]         # data2线条
6  data3 = [1, 3, 6, 10, 15, 21, 28, 36]         # data3线条
7  data4 = [1, 7, 15, 26, 40, 57, 77, 100]       # data4线条
8
9  seq = [1, 2, 3, 4, 5, 6, 7, 8]
10 plt.plot(seq, data1, '-*', seq, data2, '-o', seq, data3, '-^', seq, data4, '-s')
11 plt.title("Test Chart", fontsize=24)
12 plt.xlabel("x-Value", fontsize=14)
13 plt.ylabel("y-Value", fontsize=14)
14 plt.tick_params(axis='both', labelsize=12, color='red')
15 plt.show()
```

执行结果



1-2-7 刻度设计

目前，所有图表的 x 轴和 y 轴的刻度皆是 plot( ) 方法针对所输入的参数默认设定的，请先参考下列实例。

程序实例 ch1_11.py：假设 3 大品牌车辆 2021—2023 年的销售数据如下：

Benz	3367	4120	5539
BMW	4000	3590	4423
Lexus	5200	4930	5350

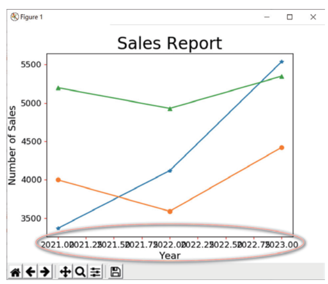
请将上述数据绘制成图表。

```
# ch1_11.py
import matplotlib.pyplot as plt

Benz = [3367, 4120, 5539]                # Benz线条
BMW = [4000, 3590, 4423]                 # BMW线条
Lexus = [5200, 4930, 5350]               # Lexus线条

seq = [2021, 2022, 2023]                 # 年度
plt.plot(seq, Benz, '-*', seq, BMW, '-o', seq, Lexus, '-^')
plt.title("Sales Report", fontsize=24)
plt.xlabel("Year", fontsize=14)
plt.ylabel("Number of Sales", fontsize=14)
plt.tick_params(axis='both', labelsize=12, color='red')
plt.show()
```

执行结果

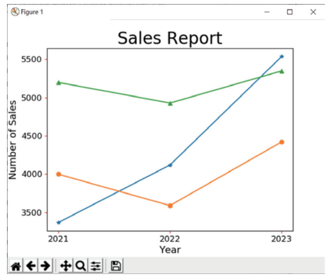


上述程序最大的遗憾是 x 轴的刻度，对我们而言，其实只要有 2021、2022、2023 这 3 个刻度即可，还好可以使用 pyplot 模块的 xticks( )、yticks( ) 分别设定 x、y 轴刻度，可参考下列实例。

程序实例 ch1_12.py：重新设计 ch1_11.py，自行设定刻度，这个程序的重点是第 9 行，将 seq 列表当作参数放在 plt.xticks( ) 内。

```
1  # ch1_12.py
2  import matplotlib.pyplot as plt
3
4  Benz = [3367, 4120, 5539]              # Benz线条
5  BMW = [4000, 3590, 4423]               # BMW线条
6  Lexus = [5200, 4930, 5350]             # Lexus线条
7
8  seq = [2021, 2022, 2023]               # 年度
9  plt.xticks(seq)                        # 设定x轴刻度
10 plt.plot(seq, Benz, '-*', seq, BMW, '-o', seq, Lexus, '-^')
11 plt.title("Sales Report", fontsize=24)
12 plt.xlabel("Year", fontsize=14)
13 plt.ylabel("Number of Sales", fontsize=14)
14 plt.tick_params(axis='both', labelsize=12, color='red')
15 plt.show()
```

执行结果



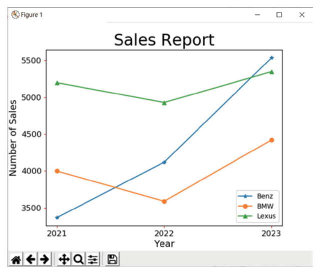
1-2-8　图例 legend( )

本章所建立的图表，应该说已经很好了，缺点是缺乏各种线条代表的意义，在 Excel 中称图例 (legend)，下列笔者将直接以实例说明。

程序实例 ch1_13.py：为 ch1_12.py 建立图例。

```
1  # ch1_13.py
2  import matplotlib.pyplot as plt
3
4  Benz = [3367, 4120, 5539]              # Benz线条
5  BMW = [4000, 3590, 4423]               # BMW线条
6  Lexus = [5200, 4930, 5350]             # Lexus线条
7
8  seq = [2021, 2022, 2023]               # 年度
9  plt.xticks(seq)                        # 设定x轴刻度
10 plt.plot(seq, Benz, '-*', label='Benz')
11 plt.plot(seq, BMW, '-o', label='BMW')
12 plt.plot(seq, Lexus, '-^', label='Lexus')
13 plt.legend(loc='best')
14 plt.title("Sales Report", fontsize=24)
15 plt.xlabel("Year", fontsize=14)
16 plt.ylabel("Number of Sales", fontsize=14)
17 plt.tick_params(axis='both', labelsize=12, color='red')
18 plt.show()
```

执行结果



这个程序最大不同在第 10 ～ 12 行，下列是以第 10 行解释。

```
plt.plot(seq, Benz, '-*', label='Benz')
```

上述调用 plt.plot( ) 时需同时设定 label，最后使用第 13 行的方式执行 legend( ) 图例的调用。其中参数 loc 可以设定图例的位置，可以有下列设定方式：

'best'：0
'upper right'：1
'upper left'：2
'lower left'：3
'lower right'：4
'right'：5（与 'center right' 相同）
'center left'：6
'center right'：7
'lower center'：8
'upper center'：9
'center'：10

如果省略 loc 设定，则使用预设‘best’，在应用时可以使用设定整数值，例如：设定 loc=0 与上述效果相同。若是顾虑程序可读性，建议使用文字字符串方式设定，当然也可以直接设定数字。

程序实例 ch1_13_1.py：在 ch1_13.py 的基础上省略 loc 设定。

```
plt.legend()
```

执行结果 与 ch1_13.py 相同。

程序实例 ch1_13_2.py：在 ch1_13.py 的基础上设定 loc=0。

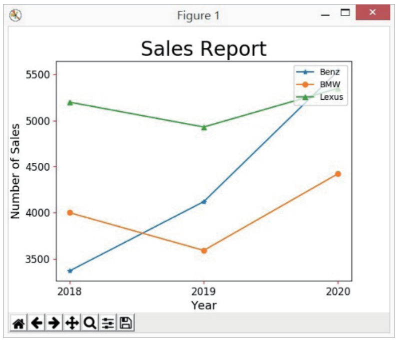
```
plt.legend(loc=0)
```

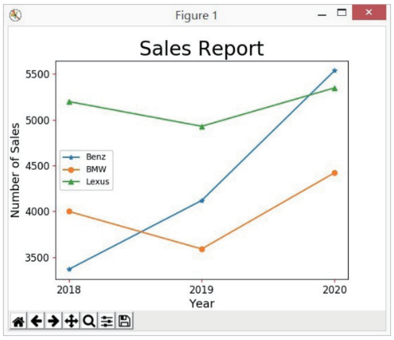
执行结果 与 ch1_13.py 相同。

程序实例 ch1_13_3.py：在 ch1_13.py 的基础上设定图例在右上角。

```
plt.legend(loc='upper right')
```

执行结果 下方左图。



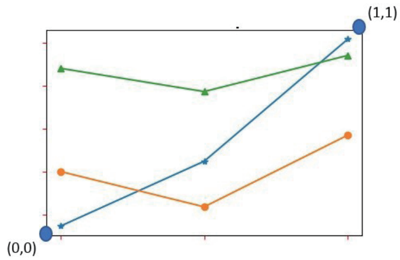


程序实例 ch1_13_4.py：在 ch1_13.py 的基础上设定图例在左边中央。

```
plt.legend(loc=6)
```

执行结果 如上右图。

经过上述解说，我们已经可以将图例放在图表内了。如果想将图例放在图表外，需要先理解坐标，在图表内左下角位置坐标是 (0,0)，右上角位置坐标是 (1,1)，概念如下：

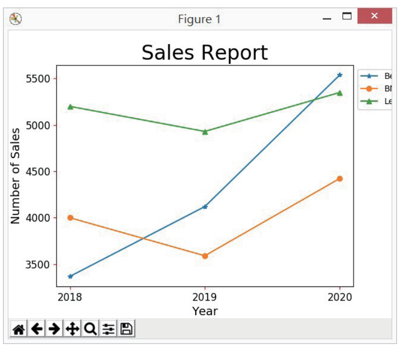


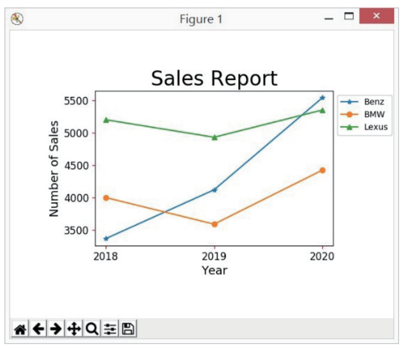
首先需使用 bbox_to_anchor( ) 当作 legend( ) 的一个参数，设定锚点 (anchor)，也就是图例位置，例如：如果我们想将图例放在图表右上角外侧，需设定 loc=‘upper left’，然后设定 bbox_to_anchor(1,1)。

程序实例 ch1_13_5.py：在 ch1_13.py 的基础上将图例放在图表右上角外侧。

```
plt.legend(loc='upper left', bbox to anchor=(1,1))
```

执行结果　下方左图。





上述最大的缺点是由于图表与 Figure 1 的留白不足，造成无法完整显示图例。matplotlib 模块内有 tight_layout( ) 函数，可利用设定 pad 参数在图表与 Figure 1 间设定留白。

程序实例 ch1_13_6.py：设定 pad=7，重新设计 ch1_13_5.py。

```
plt.legend(loc='upper left',bbox_to_anchor=(1,1))
plt.tight_layout(pad=7)
```

执行结果　可参考如上右图。

很明显图例显示不完整的问题改善了。如果将 pad 改为 h_pad/w_pad 可以分别设定高度/宽度的留白。

1-2-9　保存与开启文件

图表设计完成，可以使用 savefig( ) 保存文件，这个方法需放在 show( ) 的前方，表示先储存再显示图表。

程序实例 ch1_14.py：扩充 ch1_13.py，在屏幕显示图表前，先将图表存入目前文件夹的 out1_14.jpg。

```
# ch1_14.py
import matplotlib.pyplot as plt

Benz = [3367, 4120, 5539]                # Benz线条
BMW = [4000, 3590, 4423]                 # BMW线条
Lexus = [5200, 4930, 5350]               # Lexus线条

seq = [2021, 2022, 2023]                 # 年度
plt.xticks(seq)                          # 设定x轴刻度
plt.plot(seq, Benz, '-*', label='Benz')
plt.plot(seq, BMW, '-o', label='BMW')
plt.plot(seq, Lexus, '-^', label='Lexus')
plt.legend(loc='best')
plt.title("Sales Report", fontsize=24)
plt.xlabel("Year", fontsize=14)
plt.ylabel("Number of Sales", fontsize=14)
plt.tick_params(axis='both', labelsize=12, color='red')
plt.savefig('out1_14.jpg', bbox_inches='tight')
plt.show()
```

执行结果

读者可以在 ch1 文件夹看到 out1_14.jpg 文件。

上述 plt.savefig( ) 中第一个参数是所存的文件名，第二个参数是将图表外多余的空间删除。

要开启文件可以使用 matplotlib.image 模块，可以参考下列实例。

程序实例 ch1_15.py：开启 out1_14.jpg 文件。

```
# ch1_15.py
import matplotlib.pyplot as plt
import matplotlib.image as img

fig = img.imread('out1_14.jpg')
plt.imshow(fig)
plt.show()
```

执行结果

上述程序可以顺利开启 out1_14.jpg 文件。

1-2-10 在图上标记文字

在绘制图表过程中，有时需要在图上标记文字，这时可以使用 text( ) 函数，此函数基本格式如下：

text (x, y, ‘文字符串’)

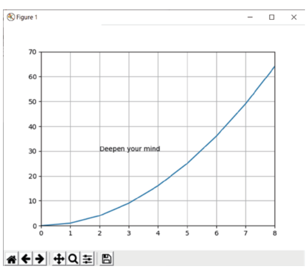
x，y 是文字输出的左下角坐标，它不是绝对坐标，是相对坐标，大小会随着坐标刻度增减。

程序实例 ch1_15_1.py：增加文字重新设计 ch1_3_1.py。

```
# ch1_15_1.py
import matplotlib.pyplot as plt

squares = [0, 1, 4, 9, 16, 25, 36, 49, 64]
plt.plot(squares)              # 列表squares数据是y轴的值
plt.axis([0, 8, 0, 70])        # x轴刻度0～8, y轴刻度0～70
plt.text(2, 30, 'Deepen your mind')
plt.grid()
plt.show()
```

执行结果



1-3 绘制散点图 scatter( )

前方介绍了可以使用 plot( ) 绘制散点图，本节将介绍绘制散点图的常用方法 scatter( )。

1-3-1　基本散点图的绘制

绘制散点图可以使用 scatter( )，基本语法如下（更多参数后面章节会解说）：

```
scatter(x, y, s, c, cmap)
```

x，y：在 (x,y) 位置绘图。

s：绘图点的大小，预设是 20。

c：颜色，可以参考 1-2-6 节。

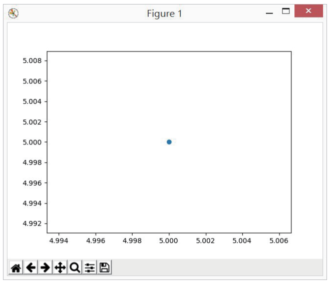
cmap：彩色图表，可以参考 1-4-5 节。

程序实例 ch1_16.py：在坐标轴 (5,5) 绘制一个点。

```python
# ch1_16.py
import matplotlib.pyplot as plt

plt.scatter(5, 5)
plt.show()
```

执行结果



1-3-2　绘制系列点

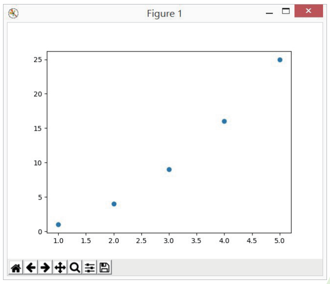
如果我们想绘制系列点，可以将系列点的 *x* 轴值放在一个列表，*y* 轴值放在另一个列表，然后将这 2 个列表作为参数放在 scatter( ) 即可。

程序实例 ch1_17.py：绘制系列点的应用。

```python
# ch1_17.py
import matplotlib.pyplot as plt

xpt = [1,2,3,4,5]
ypt = [1,4,9,16,25]
plt.scatter(xpt, ypt)
plt.show()
```

执行结果

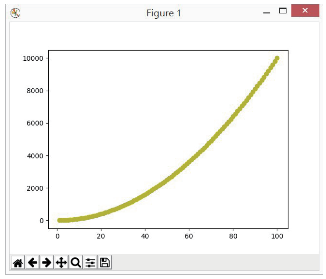


在程序设计时，有些系列点的坐标可能是由程序产生，其实应用方式是一样的。另外，可以在 scatter( ) 内增加 color（也可用 c）参数，可以设定点的颜色。

程序实例 ch1_18.py：绘制黄色的系列点，这个系列点有 100 个点，x 轴的点由 range(1,101) 产生，相对应 y 轴的值则是 x 的平方值。

```
1  # ch1_18.py
2  import matplotlib.pyplot as plt
3
4  xpt = list(range(1,101))
5  ypt = [x**2 for x in xpt]
6  plt.scatter(xpt, ypt, color='y')
7  plt.show()
```

执行结果



上述程序第 6 行是直接指定色彩，也可以使用 RGB(Red，Green，Blue) 颜色模式设定色彩，RGB( ) 内每个参数数值是 0 ～ 1。

1-3-3　设定绘图区间

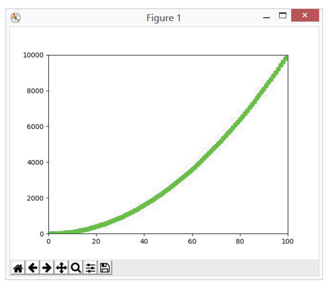
可以使用 axis( ) 设定绘图区间，语法格式如下：

```
axis([xmin, xmax, ymin, ymax])        # 分别代表 x 轴和 y 轴的最小和最大区间
```

程序实例 ch1_19.py：设定绘图区间为 [0,100,0,10000] 的应用，读者可以将这个执行结果与 ch1_18.py 做比较。

```
1  # ch1_19.py
2  import matplotlib.pyplot as plt
3
4  xpt = list(range(1,101))
5  ypt = [x**2 for x in xpt]
6  plt.axis([0, 100, 0, 10000])              # 参数是列表
7  plt.scatter(xpt, ypt, color=(0, 1, 0))    # 绿色
8  plt.show()
```

执行结果



上述程序第 5 行是依据 xpt 列表产生 ypt 列表值的方式，由于网络上有很多文章使用数组方式产生图表列表，所以下一节笔者将对此做出说明，期待可为读者建立基础。

1-4 numpy 模块

numpy 是 Python 的一个扩充模块，可以支持多维度空间的数组与矩阵运算，本节笔者将对其最简单的产生数组的功能做解说，由此可以将这个功能扩充到数据图表的设计。使用前我们需导入 numpy 模块，如下所示：

```
import numpy as np
```

1-4-1　建立一个简单的数组 linspace( ) 和 arange( )

在 numpy 模块中最基本的就是 linspace( ) 方法，使用它可以很方便地产生等距的数组，它的语法如下：

```
linspace(start, end, num)
```

start 是起始值，end 是结束值，num 是设定产生多少个等距点的数组值，num 的默认值是 50。

在网络上阅读他人使用 Python 设计的图表时，常看到的产生数组的方法是 arange( )。其语法如下：

```
arange(start, stop, step)                    # start 和 step 可以省略
```

start 是起始值，如果省略默认值是 0。stop 是结束值，但是所产生的数组不包含此值。step 是数组相邻元素的间距，如果省略默认值是 1。

程序实例 ch1_20.py：建立 0, 1, …, 9, 10 的数组。

```
# ch1_20.py
import numpy as np

x1 = np.linspace(0, 10, num=11)      # 使用linspace()产生数组
print(type(x1), x1)
x2 = np.arange(0,11,1)               # 使用arange()产生数组
print(type(x2), x2)
x3 = np.arange(11)                   # 简化语法产生数组
print(type(x3), x3)
```

执行结果

```
=========== RESTART: D:/Python Machine Learning Math/ch1/ch1_20.py ===========
<class 'numpy.ndarray'> [ 0.  1.  2.  3.  4.  5.  6.  7.  8.  9. 10.]
<class 'numpy.ndarray'> [ 0  1  2  3  4  5  6  7  8  9 10]
<class 'numpy.ndarray'> [ 0  1  2  3  4  5  6  7  8  9 10]
```

1-4-2　绘制波形

中学数学中我们有学过 sin( ) 和 cos( ) 概念，其实有了数组数据，我们可以很方便地绘制正弦和余弦的波形变化。单纯绘点可以使用 scatter( ) 方法，此方法使用格式如下：

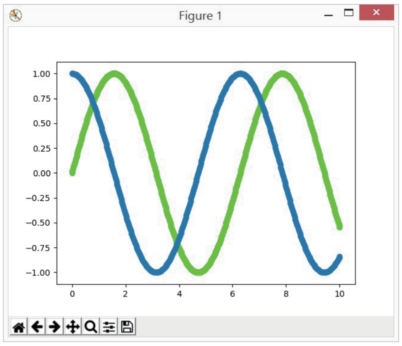
scatter(x, y, marker= ‘.’ , c(或 color)= ‘颜色’)　　# marker 如果省略会使用预设

程序实例 ch1_21.py：绘制 sin( ) 和 cos( ) 的波形，在这个实例中调用 plt.scatter( ) 方法 2 次，相当于也可以绘制 2 次波形图表。

```python
# ch1_21.py
import matplotlib.pyplot as plt
import numpy as np

xpt = np.linspace(0, 10, 500)          # 建立含500个元素的数组
ypt1 = np.sin(xpt)                      # y数组的变化
ypt2 = np.cos(xpt)
plt.scatter(xpt, ypt1, color=(0, 1, 0)) # 绿色
plt.scatter(xpt, ypt2)                  # 预设颜色
plt.show()
```

执行结果



其实一般在绘制波形时，最常用的还是 plot( ) 方法。

程序实例 ch1_22.py：使用系统默认颜色，绘制不同波形的应用。

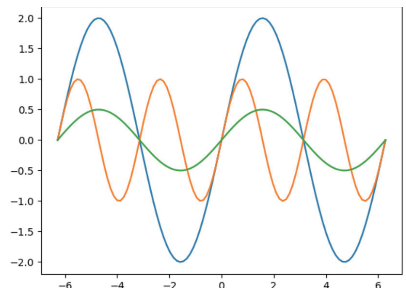
```python
# ch1_22.py
import matplotlib.pyplot as plt
import numpy as np

left = -2 * np.pi
right = 2 * np.pi
x = np.linspace(left, right, 100)

f1 = 2 * np.sin(x)                # y数组的变化
f2 = np.sin(2*x)
f3 = 0.5 * np.sin(x)

plt.plot(x, f1)
plt.plot(x, f2)
plt.plot(x, f3)
plt.show()
```

执行结果



1-4-3　建立不等宽度的散点图

在 scatter( ) 方法中，(x,y) 的数据可以是列表也可以是矩阵，预设所绘制点大小 s 的值是 20，这个 s 可以是一个值也可以是一个数组数据，当它是一个数组数据时，利用更改数组值的大小，我们就可以建立不同大小的散点图。

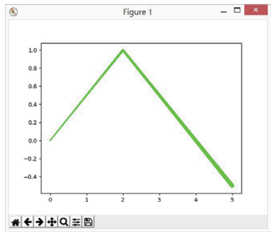
在我们使用 Python 绘制散点图时，如果在两个点之间绘制了上百或上千个点，则可以产生绘制线条的视觉效果，如果每个点的大小不同，且依一定规律变化，则有特别的效果。

程序实例 ch1_23.py：建立一个不等宽度的图形。

```
# ch1_23.py
import matplotlib.pyplot as plt
import numpy as np

xpt = np.linspace(0, 5, 500)
ypt = 1 - 0.5*np.abs(xpt-2)
lwidths = (1+xpt)**2
plt.scatter(xpt, ypt, s=lwidths, color=(0, 1, 0))
plt.show()
```

执行结果



1-4-4　填满区间

在绘制波形时，如要填满区间，此时可以使用 matplotlib 模块的 fill_between( ) 方法，基本语法如下：

```
fill_between(x, y1, y2, color, alpha, options, … )  # options 是其他参数
```

上述会填满所有相对 x 轴数列 $y1$ 至 $y2$ 的区间，如果不指定填满颜色，则会使用预设的线条颜色填满，通常填满颜色会用较淡的颜色，所以可以设定 alpha 参数将颜色调淡。

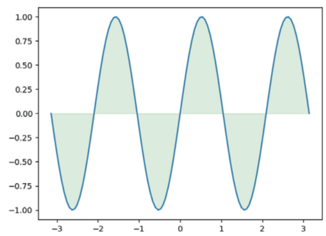
程序实例 ch1_24.py：填满区间 0 ～ y，所使用的 y 轴值是函数式 $\sin(3x)$。

```
# ch1_24.py
import matplotlib.pyplot as plt
import numpy as np

left = -np.pi
right = np.pi
x = np.linspace(left, right, 100)
y = np.sin(3*x)                          # y数组的变化

plt.plot(x, y)
plt.fill_between(x, 0, y, color='green', alpha=0.1)
plt.show()
```

执行结果



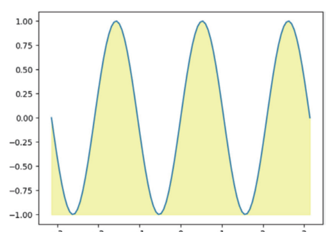
程序实例 ch1_25.py：填满区间 −1 ～ y，所使用的 y 轴值是函数式 $\sin(3x)$。

```
# ch1_25.py
import matplotlib.pyplot as plt
import numpy as np

left = -np.pi
right = np.pi
x = np.linspace(left, right, 100)
y = np.sin(3*x)                          # y数组的变化

plt.plot(x, y)
plt.fill_between(x, -1, y, color='yellow', alpha=0.3)
plt.show()
```

执行结果



1-4-5 色彩映射

至今我们针对一组数组或列表所绘制的图表皆是单色，以 ch1_23.py 第 8 行为例，色彩设定是 color=(0,1,0)，这是固定颜色的用法。在色彩的使用中，允许色彩随着数据而做变化，此时色彩的变化是根据所设定的色彩映射值 (color mapping) 而定，例如有一个色彩映射值是 rainbow，内容如下：

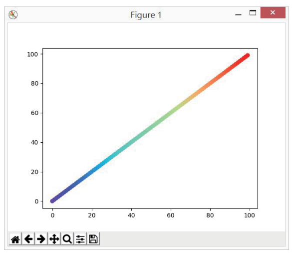
在数组或列表中，数值低的值颜色在左边，会随数值变高往右边移动。当然在程序设计中，我们需要在 scatter( ) 中增加 color 设定参数 c，这时 color 的值就变成一个数组或列表。然后我们需要增加参数 cmap（英文是 color map），这个参数主要是指定使用哪一种色彩映射值。

程序实例 ch1_26.py：色彩映射的应用。

```
# ch1_26.py
import matplotlib.pyplot as plt
import numpy as np

x = np.arange(100)
y = x
t = x
plt.scatter(x, y, c=t, cmap='rainbow')
plt.show()
```

执行结果

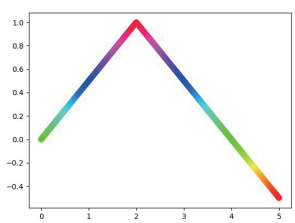


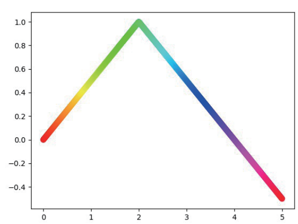
色彩映射也可以设定根据 x 轴的值做变化，或根据 y 轴的值做变化，整个效果是不一样的。

程序实例 ch1_27.py：重新设计 ch1_23.py，主要是设定固定点的宽度为 50，将色彩改为依 y 轴值变化，同时使用 hsv 色彩映射表。

```
8  plt.scatter(xpt, ypt, s=50, c=ypt, cmap='hsv')          # 色彩随y轴值变化
```

执行结果 如下方左图。





程序实例 ch1_28.py：重新设计 ch1_27.py，主要是将色彩改为依 x 轴值变化。

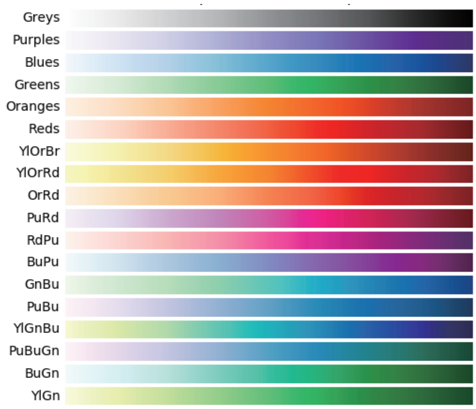
```
8  plt.scatter(xpt, ypt, s=50, c=xpt, cmap='hsv')          # 色彩随x轴值变化
```

执行结果 如上右图。

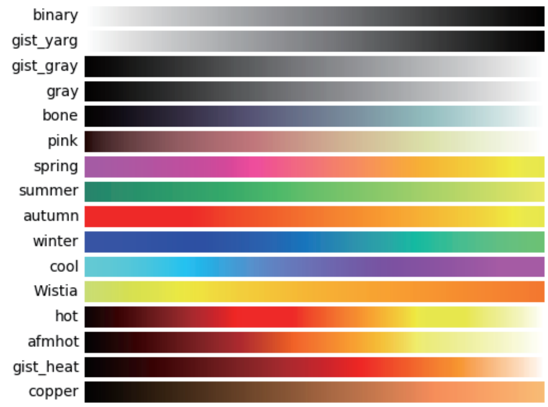
机器学习数学基础一本通（Python版） 四校 正文.indd 18 2021/3/8 15:13:51

目前 matplotlib 协会所提供的色彩映射内容如下：

❑ 序列色彩映射表

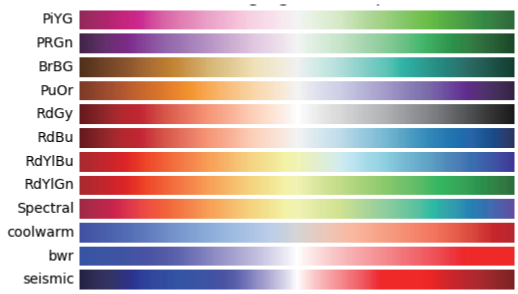


❑ 序列 2 色彩映射表

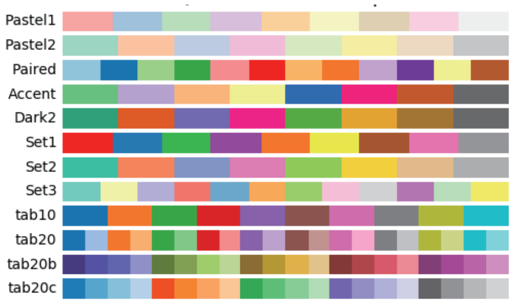


❑ 直觉一致的色彩映射表

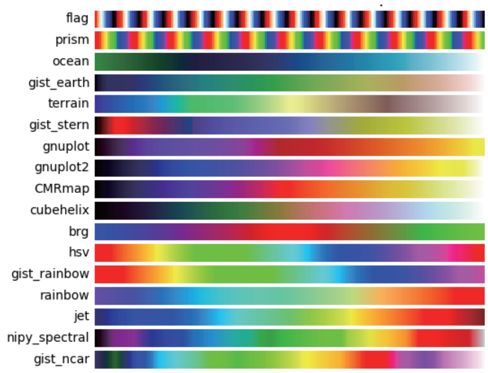
❑ 发散式的色彩映射表



❑ 定性色彩映射表



❑ 杂项色彩映射表



在大数据研究应用中，可以将数据以图表显示，然后用色彩判断整个数据的趋势。在结束本节之前，笔者举一个使用 colormap 绘制数组数据的实例，这个程序会使用下列方法。

```
imshow(img, cmap='xx')
```

参数 img 可以是图片，也可以数组数据，此例是数组数据。这个函数常用在机器学习检测神经网络的输出中。

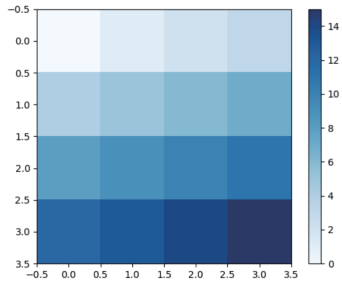
程序实例 ch1_29.py：绘制矩形数组数据。

```
# ch1_29.py
import matplotlib.pyplot as plt
import numpy as np

img = np.array([[0, 1, 2, 3],
                [4, 5, 6, 7],
                [8, 9 , 10, 11],
                [12, 13, 14, 15]])

plt.imshow(img, cmap='Blues')
plt.colorbar()
plt.show()
```

执行结果



1-5 图表显示中文

matplotlib 无法显示中文，主要在于安装此模块时所配置的下列文件：

```
~Python37\Lib\site-packages\matplotlib\mpl-data\matplotlibrc
```

在此文件内的 font_sans-serif 中没有配置中文字体，我们可以在此字段增加中文字体，但是笔者不鼓励更改系统内建文件。笔者将使用动态配置方式处理，让图表显示中文字体。其实可以在程序内增加下列程序代码，rcParams( ) 方法可以为 matplotply 配置中文字体参数，就可以显示中文了。

```
from pylab import mlp                                  # matplotlib 的子模块
mlp.rcParams[“font.sans-serif”] = [“SimHei”]           # 黑体
mlp.rcParams[“axes.unicode_minus”] = False             # 可以显示负号
```

另外，每个要显示的中文字符串需要在前面加上 u。

程序实例 ch1_30.py：重新设计 ch1_13.py，以中文显示报表。

```
# ch1_30.py
import matplotlib.pyplot as plt
from pylab import mpl

mpl.rcParams["font.sans-serif"] = ["SimHei"]          # 使用黑体

Benz = [3367, 4120, 5539]                  # Benz线条
BMW = [4000, 3590, 4423]                   # BMW线条
Lexus = [5200, 4930, 5350]                 # Lexus线条

seq = [2021, 2022, 2023]                   # 年度
plt.xticks(seq)                            # 设定x轴刻度
plt.plot(seq, Benz, '-*', label='Benz')
plt.plot(seq, BMW, '-o', label='BMW')
plt.plot(seq, Lexus, '-^', label='Lexus')
plt.legend(loc='best')
plt.title(u"销售报表", fontsize=24)
plt.xlabel(u"年度", fontsize=14)
plt.ylabel(u"销售量", fontsize=14)
plt.tick_params(axis='both', labelsize=12, color='red')
plt.show()
```

执行结果