返回目录

第二章　数据处理及可视化初步

本章导读

看看Matplotlib文案，就可以写好本章。那篇文章写得不错。

学习目标：

1. 初步掌握Python图形库库Matplotlib；
2. 初步掌握Python的数据处理库Numpy；
3. 初步掌握Python的数据分析库Pandas；

本章目录

第一节　快速了解

例程2-1是0-6立方值的可视化程序，图2-1是其运行效果。在Python中，Matplotlib是最重要的图形库。例程第2行是引入Matplotlib库并命名为plt。第4行是用于可视化的数据，第5行是选择类型，第6行是显示该图形。当该程序运行后，将显示如图2-1所示效果。

例程2-1

```
第1行 #数据可视化初步
第2行 import matplotlib.pyplot as plt
第3行
第4行 dataList=[0,1,8,27,64,125] #0-6的立方值，可视化的数据
第5行 plt.plot(dataList) #可视化类型
第6行 plt.show() #显示图形
第7行
第8行 #eof
```

图片底部有一些列图标用于控制图形显示，其中最右侧图标可将当前显示图片保存为各种图形或图像格式，如常见：JPG、PNG、SVG甚至PDF格式等，可用于多种场景，如论文等等。

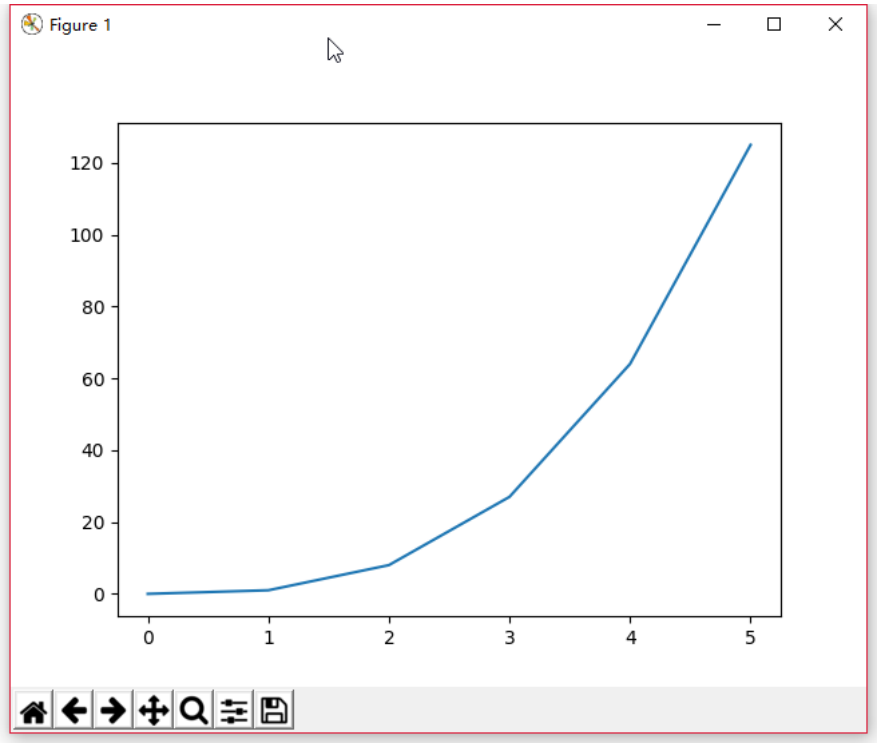

图2-1　例程2-1的运行效果图

图2-2对应显示5、2、4、7的平方，可y值25对应缺为0，16对应为2，明显错误，因为此时对应值不再是默认从0-6的顺序增长，此时可指定x轴对应数据，如例程2-2所示。例程第plt.plot([5,2,4,7],dataList)中的[5,2,4,7]为x轴的数据，dataList为对应y轴数据。只有设定x轴和y轴对应数据，即可绘制出相应图形。

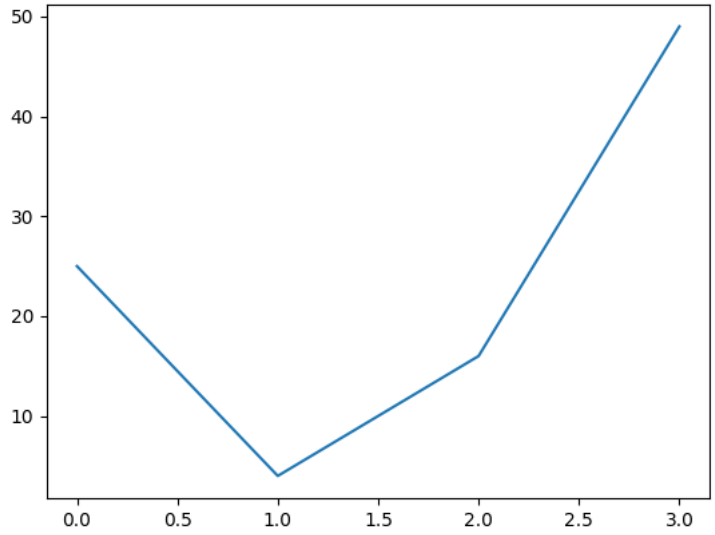

图2-2　5、2、4、7应平方的错误效果图

例程2-2

```
第1行 #数据可视化初步
第2行 import matplotlib.pyplot as plt
第3行
第4行 dataList=[25,4,16,49] #0-6的立方值，可视化的数据
第5行 plt.plot([5,2,4,7],dataList) #可视化类型
第6行 plt.show() #显示图形
第7行
第8行 #eof
```

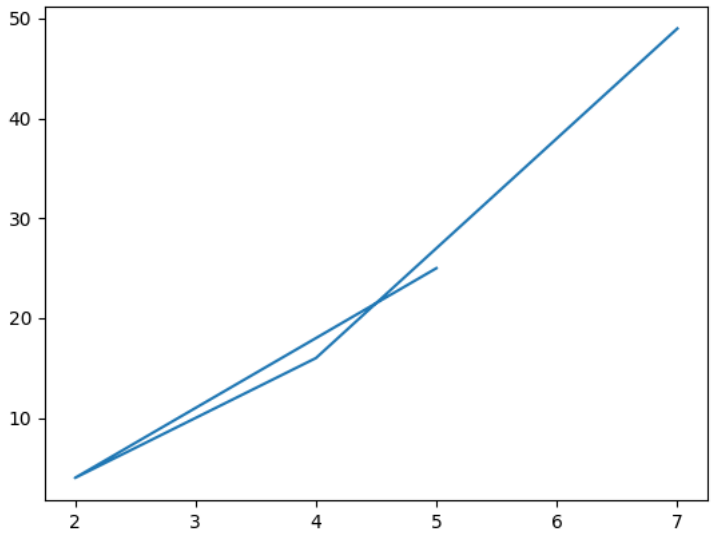

图2-3　例程2-2应平方的错误效果图

Matplotlib默认为折线图，很明显在此处并不合适，可以将折线图变化为散点图，如例程2-3所示，其运行效果如图2-4所示。

例程2-3

```
第1行 #数据可视化初步
第2行 import matplotlib.pyplot as plt
第3行
第4行 dataList=[25,4,16,49] #0-6的立方值，可视化的数据
第5行 plt.plot([5,2,4,7],dataList,"ro") #可视化类型，r表示红色，o表示圆点
第6行 plt.show() #显示图形
第7行
第8行 #eof
```

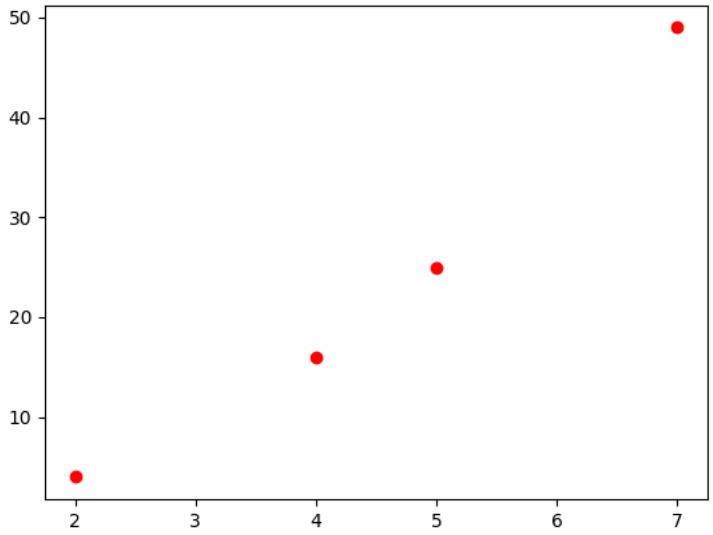

图2-4　例程2-3的运行效果图

如何同时在一张图绘制出1-5的平方和立方呢？如例程2-4所示，图2-5是其运行效果。

例程2-4

```
第1行 #数据可视化初步
第2行 import matplotlib.pyplot as plt
第3行
```

```
第4行 x=[1,2,3,4,5]
第5行 y1=[1,4,9,16,25] #0-6的立方值，可视化的数据
第6行 y2=[1,8,27,64,125]
第7行 plt.plot(x,y1,"ro",x,y2,"b*") #可视化类型
第8行 plt.show() #显示图形
第9行
第10行 #eof
```

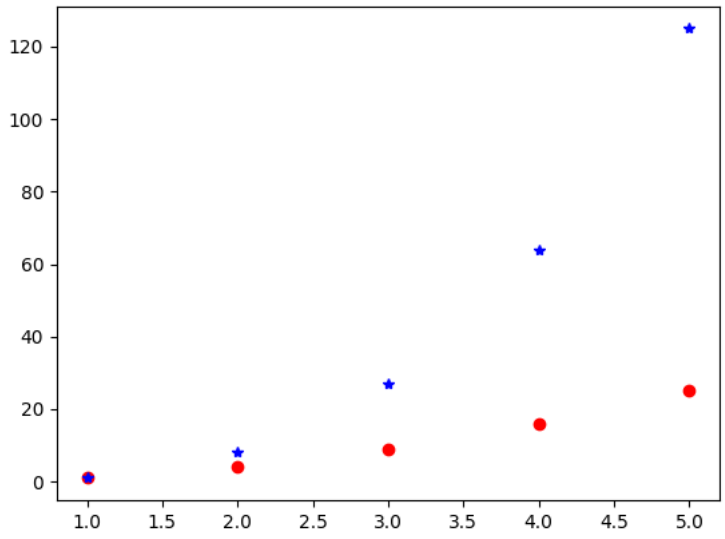

图2-5　例程2-3的运行效果图

在例程2-3、例程2-4中的ro和b*分别表示颜色和线型或标志类型。r表示红色(red简写)，b表示蓝色(blue简写)，类似还有c(Cyan简写，青色)、g(Green简写，绿色)、m(Magenta简写，洋红)、y(Yellow简写，黄色)、k(Black简写，黑色)以及w(White简写，白色)。o表示原型图标，*表示星星图标。更多线型或图标如图2-6所示。在图例中，第一列为图标或线型，第二列(即括号前)为符号，括号中内容为其英文注释。ro中的o对应为circle marker(图例第7行)，以此类推。

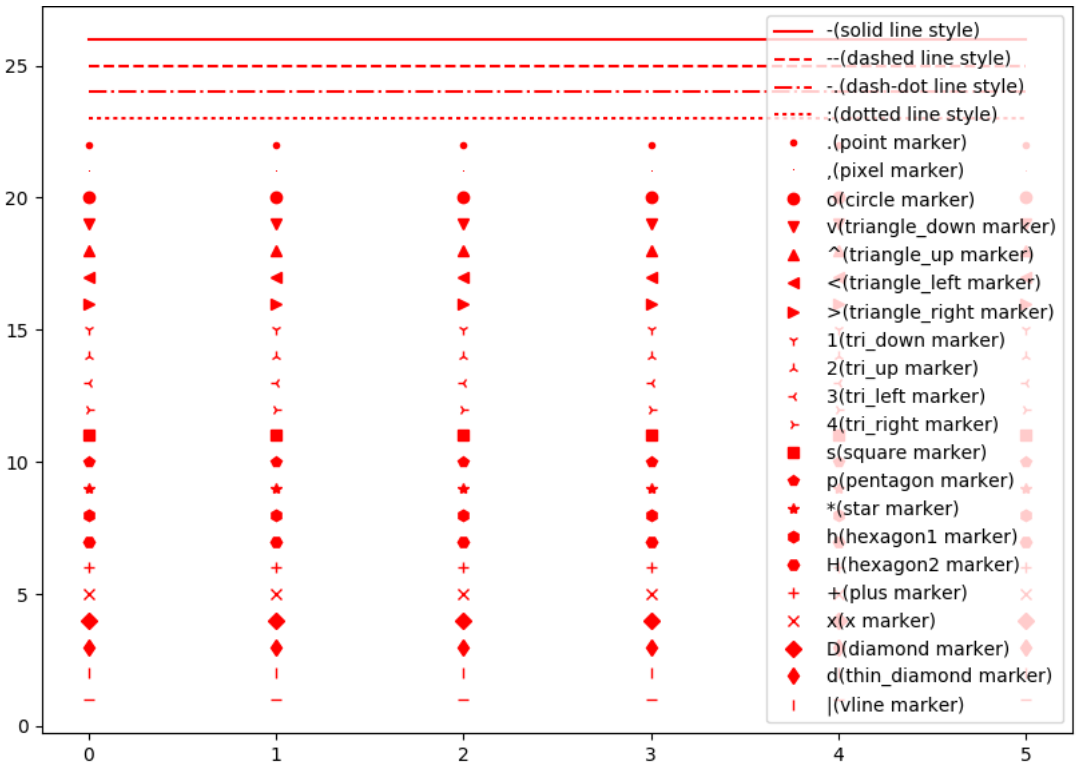

图2-6　例程2-3的运行效果图

自然常数e的x次方曲线理应是圆滑曲线，但由于例程2-5第5行x的取值间隔只能为整数，不能为小数，因此曲线成折线，如图2-7所示。例程2-6与例程2-5类似，但产生圆滑曲线。

例程2-5

```
第1行 #数据可视化初步
第2行 import matplotlib.pyplot as plt
第3行 import math
第4行
第5行 x=[x for x in range(-5,5+1)]
第6行 y=[math.e**x for x in x]
第7行
第8行 plt.plot(x,y)
第9行 plt.show() #显示图形
第10行
第11行 #eof
```

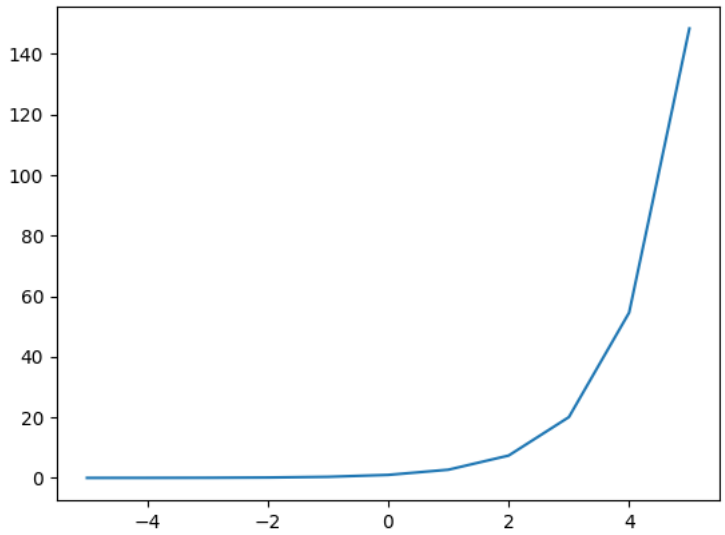

图2-7　自然常数e的x次方曲线(1)

例程2-6

```
第1行 #数据可视化初步
第2行 import numpy as np
第3行 import matplotlib.pyplot as plt
第4行 import math
第5行
第6行 #产生从-5到+5，间隔0.01的数据序列，相当于等差数列
第7行 #类似Python的range
第8行 x=np.arange(-5.0,5.0,0.01)
第9行 y=[math.e**x for x in x]
第10行
第11行 plt.plot(x,y)
第12行 plt.show() #显示图形
第13行
第14行 #eof
```

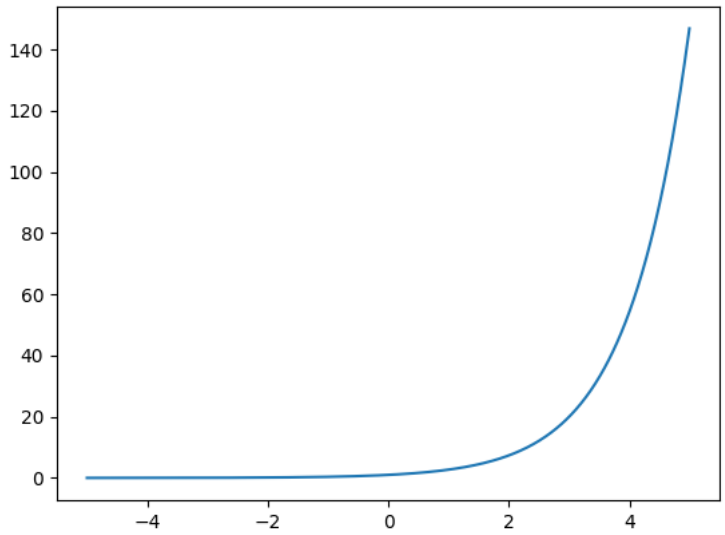

图2-8　自然常数e的x次方曲线(2)

例程2-6第1行用于引入NumPy并命名为np。NumPy系统是Python开源的数值计算扩展库，用于存储和处理大型矩阵，比Python自身的嵌套列表效率更高。从某种意义上讲，NumPy相当于将Python变成免费且更强大的MatLab系统。

在NumPy中，arange()功能与Python的range()功能类似，都是用于产生等差数列。当Python的range只能产生间隔为整数的等差数列。NumPy的arange()则能产生间隔为小数的等差数列，其使用方法如例程2-6第8行所示，其中第1个参数为起点，第2个参数为重点，第3个参数为间隔值。

在NumPy中，例程2-6可以修改如例程2-7所示，执行效率将更高，代码也更加简洁。

例程2-7

```
第1行 #数据可视化初步
第2行 import numpy as np
第3行 import matplotlib.pyplot as plt
第4行 import math
第5行
第6行 #产生从-5到+5，间隔0.01的数据序列，相当于等差数列
第7行 #类似Python的range
第8行 x=np.arange(-5.0,5.0,0.01)
第9行
第10行 plt.plot(x,math.e**x)
第11行 plt.show() #显示图形
第12行
第13行 #eof
```

在NumPy中，产生等差数列还常用linspace()，和arange()相比，linspace()同样有三个参数，第1个和第2个同样为数列的起点和终点，但第3个参数含义不同，arange()为数列的间隔，而linspace()则数量，即在起点和终点之间产生多少个数，其间隔在函数内部计算产生，如例程2-8所示，其产生图形与图2-8相同。

例程2-8

```
第1行 #数据可视化初步
第2行 import numpy as np
第3行 import matplotlib.pyplot as plt
第4行 import math
第5行
第6行 #产生从-5到+5，间隔0.01的数据序列，相当于等差数列
第7行 #类似Python的range
第8行 x=np.linspace(-5.0,5.0,1000) #在-5.0和5.0之间产生1000个等差数列
```

```
第9行
第10行  plt.plot(x,math.e**x)
第11行  plt.show() #显示图形
第12行
第13行  #eof
```

多个图可以在一张图上，如图2-9所示，其实现方法是多次调用plot()即可，每次作图数据不同而已，其x轴、y轴取值范围及其间隔由系统自动计算而成。

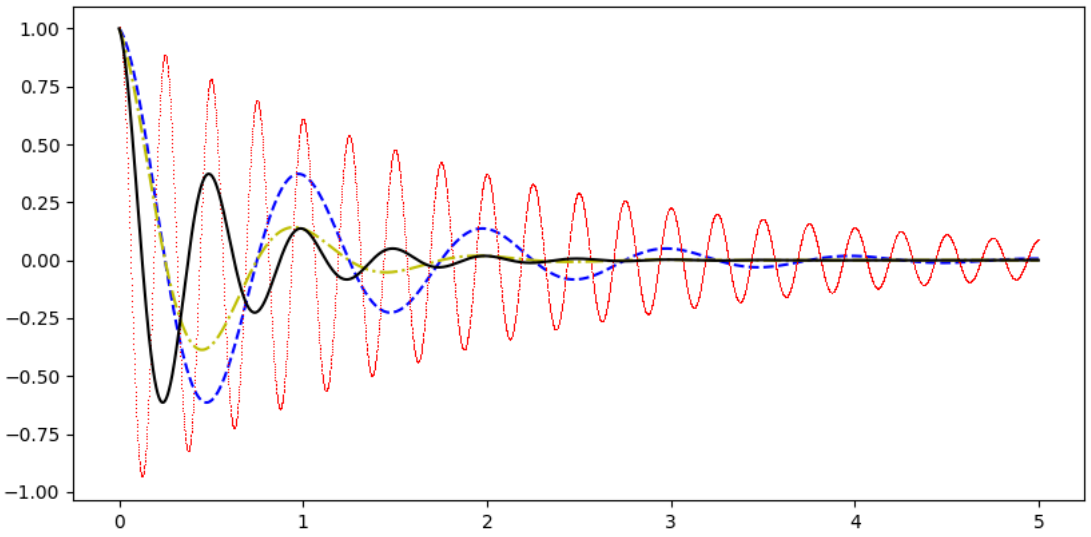

图2-9　多个图形在一张图

例程2-9

```
第1行   #数据可视化初步
第2行   import numpy as np
第3行   import matplotlib.pyplot as plt
第4行
第5行   def f(t,a,b):
第6行       return np.exp(-a*t) * np.cos(2*np.pi*b*t)
第7行
第8行   t=np.arange(0.0, 5.0, 0.001)
第9行
第10行  plt.plot(t, f(t,1,1), 'b--') #蓝色虚线
第11行  plt.plot(t, f(t,0.5,4), 'r,') #红色像素标记
第12行  plt.plot(t, f(t,2,1), 'y-.') #黄色线点图
第13行  plt.plot(t, f(t,2,2), 'k') #黑色连线图
第14行
第15行  plt.show()
第16行
第17行  #eof
```

如图2-10所示，一张图上放入多个小图(子图)，每个图有一个图形(也可以多个图形)，其实现代码如例程2-10所示，其中subplot()用于指定子图。subplot(223)的含义是纵向2个图，横向两个图，当前绘制子图在第3个图。四个图顺序依次为左上图(1,1)编号1、右上图(1,2)编号为2、左下图(2,1)编号3、右下图(2,2)编号为4。subplot()有更复杂组合形式，比如：将最下面两个图合成一个绘图区域等等，将在后续章节讲解。

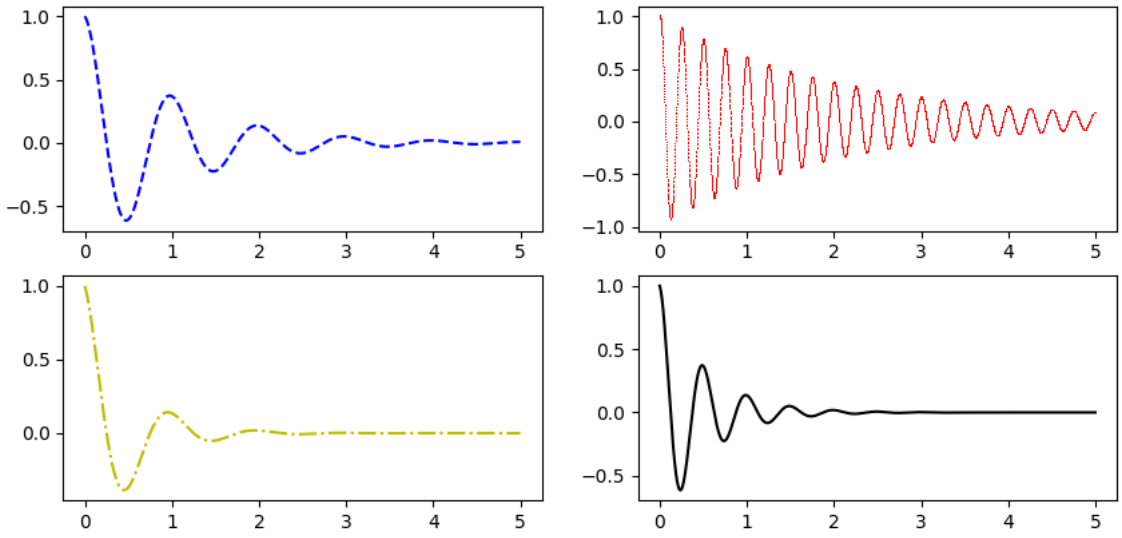

图2-10　多个图形分别在多个图(子图)

例程2-10

```
#数据可视化初步
import numpy as np
import matplotlib.pyplot as plt

def f(t,a,b):
    return np.exp(-a*t) * np.cos(2*np.pi*b*t)

t=np.arange(0.0, 5.0, 0.001)

plt.subplot(221)
plt.plot(t, f(t,1,1), 'b--') #蓝色虚线

plt.subplot(222)
plt.plot(t, f(t,0.5,4), 'r,') #红色像素标记

plt.subplot(223)
plt.plot(t, f(t,2,1), 'y-.') #黄色线点图

plt.subplot(224)
plt.plot(t, f(t,2,2), 'k') #黑色连线图

plt.show()

#eof
```

第二节　配置图形要素

Matplotlib可以根据给定数据自动配置得到合适的图形，其相关图形要素一般说来都能做的足够好。即便如此，图形大小、分辨率、线宽、颜色、风格、x轴、y轴、图例、注释文字等等，都可以通过命令调整，以更好地满足形形色色的需求。

1、工作区

在Matplotlib中，用Figure定义工作区，如例程2-11第8行所示。其中figsize用于设定图形大小，8表示图形宽8×80=640像素，6表示480像素，都为80整数倍。facecolor为图形背景色，edgecolor为边框颜色，注意是工作区而不是绘图区颜色。dpi用于色织分辨率。如果图形用于印刷，可以将设置为300，而如果仅用于显示，通常不予设置。

例程2-11

```
第1行 import numpy as np
第2行 import matplotlib.pyplot as plt
第3行
第4行 x=np.linspace(-2.0*np.pi,2.0*np.pi,1000)
第5行 y1=2*x**2*np.sin(2*x)
第6行 y2=x**2*np.cos(3*x)
第7行
第8行 plt.figure(num="Figure示例",figsize=(8,6),facecolor="#bcbcbc",edgecolor="r",dpi=96)
第9行
第10行 plt.plot(x,y1,"r-.",x,y2,"k")
第11行
第12行 plt.show()
第13行
第14行 #eof
```

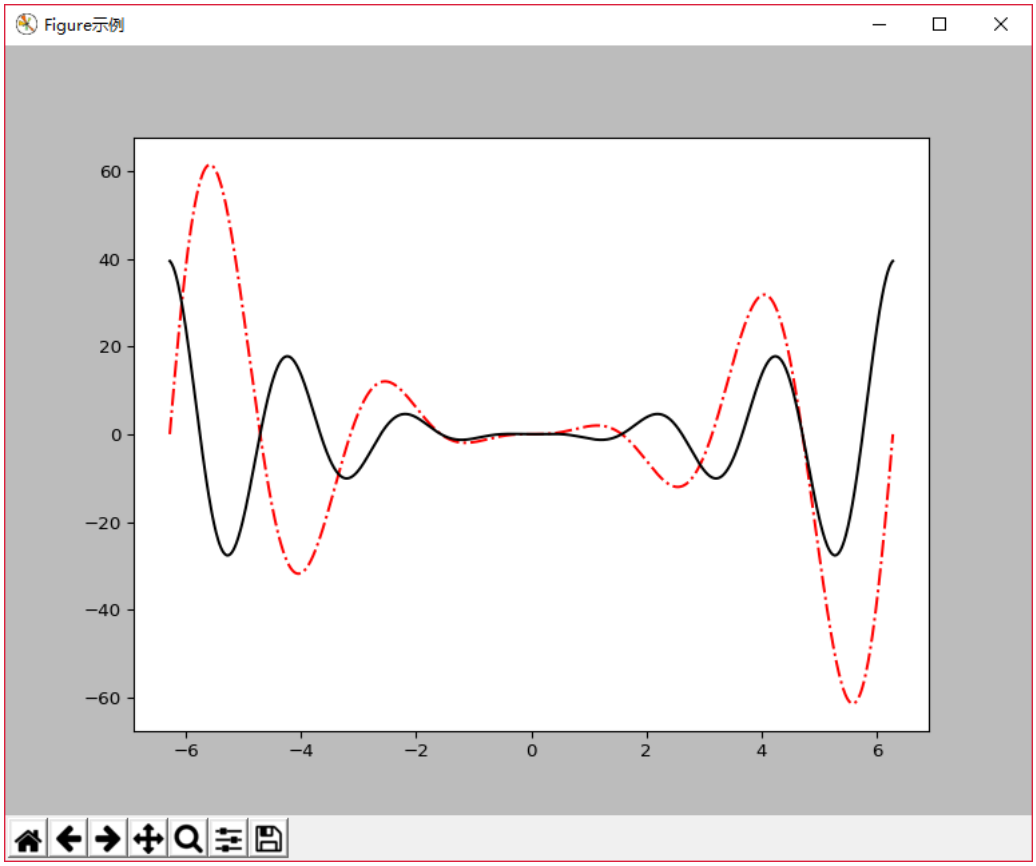

图2-11　例程2-11执行效果图

2、绘图区

例程2-12

```
第1行 import numpy as np
第2行 import matplotlib.pyplot as plt
第3行
第4行 x=np.linspace(-2.0*np.pi,2.0*np.pi,1000)
第5行 y1=2*x**2*np.sin(2*x)
第6行 y2=x**2*np.cos(3*x)
第7行
第8行 plt.figure(num="Figure示例",figsize=(8,6),facecolor="#bcbcbc",edgecolor="r",dpi=96)
第9行
第10行 plt.plot(x,y1,"r-.",x,y2,"k")
```

```
第11行
第12行 plt.title("Multiunction Graph")
第13行 plt.xlabel("X-Value")
第14行 plt.ylabel("Y-Value")
第15行 plt.text(-2.4,-50,"Matplotlib Example",fontsize=16)
第16行
第17行 plt.show()
第18行
第19行 #eof
```

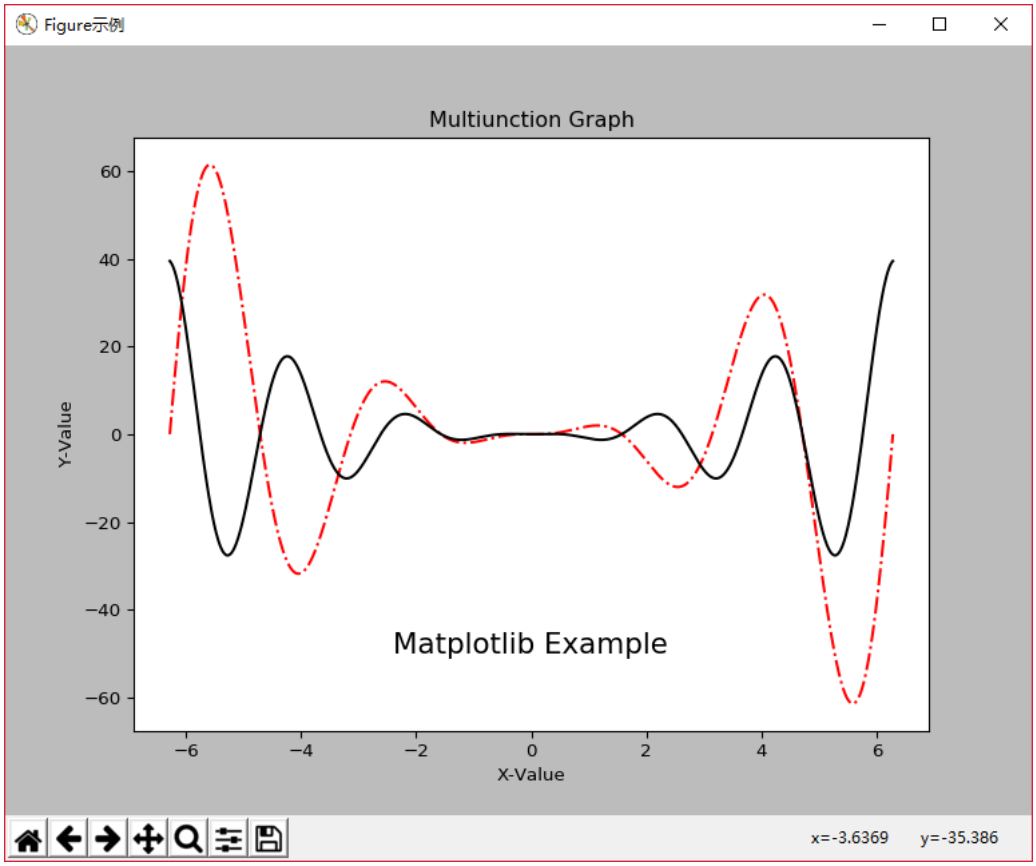

图2-12　例程2-12执行效果图

3、坐标轴

例程2-13

```
第1行 import matplotlib.pyplot as plt
第2行
第3行 cityData=[("HeNan",9613),("ShanDong",9082),("SiChuan",8673),
第4行 ("GuanDong",7859),("JiangSu",7381),("HeBei",6735),("HuNan",6629),
第5行 ("AnHui",6338),("HuBei",5988),("GuangXi",4822),("ZheJiang",4647),
第6行 ("YunNan",4333),("JiangXi",4222),("LiaoNing",4203),("GuiZhou",3837),
第7行 ("HeiLongJiang",3813),("ShaanXi",3674),("FuJian",3466),("ShanXi",3294),
第8行 ("ChongQing",3107),("JiLin",2699),("GanSu",2593),("NeiMeng",2379),
第9行 ("XinJiang",1905),("ShangHai",1625),("BeiJing",1423),("TianJin",1007),
第10行 ("HaiNan",803),("NingXia",572),("QingHai",529),("XiZang",267)]
第11行
第12行 xValue=[x for x in range(len(cityData))]
第13行 print(xValue)
第14行 yValue=[y[1] for y in cityData]
第15行
第16行 plt.plot(xValue,yValue,"ro")
```

```
第17行 # 设置坐标轴的取值范围
第18行
第19行 plt.xlim((0, 31))
第20行 plt.ylim((0,10000))
第21行
第22行 # 设置坐标轴的lable
第23行 plt.xlabel("Province Name")
第24行 plt.ylabel('Population')
第25行 # 设置x坐标轴刻度, 原来为0.25, 修改后为0.5
第26行 plt.xticks(range(1,31))
第27行 # 设置y坐标轴刻度及标签, $$是设置字体
第28行
第29行 plt.xticks(range(0,31), [provinceName[0] for provinceName in cityData],rotation=-90)
第30行 #plt.xlabels.rotation=45
第31行 plt.grid(True)
第32行 plt.legend(loc="best")
第33行
第34行 plt.show()
第35行
第36行 #eof
```

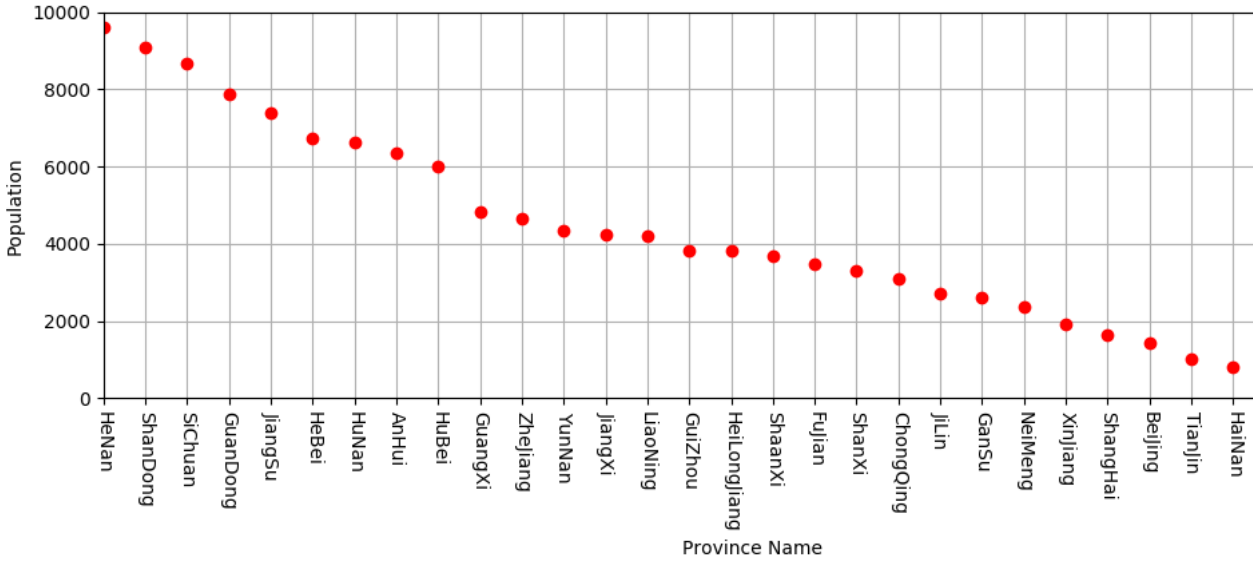

图2-13 大陆各省人口图

4、图例

如图2-14所示，顶部中间显示有图例，显示图例的代码如例程2-14所示。legend()参数loc用于设置图例的位置，其可选参数如例程注释所列。shadow用于设置图例是否有阴影，默认为None，设置为True则有阴影。常用命名参数如下：

1. **bbox_to_anchor**：用于更加灵活设置图例位置，取值如bbox_to_anchor=(0.5,0.5)；
2. **ncol**：图例列数，取值为大于0的整数，默认为1，即用1列显示图例，如果是多个图例，则上下排列。如果设置为2或更大整数，则图例可以是多列；
3. **fontsize**：用于设置字号，取值为整数或浮点数，也可以设置为‘xx-small’(超小号)，‘x-small’(较小号)，‘small’(小号)，‘medium’(中号)，‘large’(大号)，‘x-large’(较大号)，‘xx-large’(超大号)中的一项；
4. **facecolor**：用于设置图例框的填充色，颜色如同其他颜色值设定；
5. **edgecolor**：用于设置图例边框颜色，颜色如同其他颜色值设定；
6. **title**：用于设置图例标题，取值为字符串，默认为None；

另外，图例的中的公式等等，支持如Latex(即数学公式更像数学公式)，将在后续章节讲解。

例程2-14

```
第1行 #数据可视化初步--图例
```

```
import numpy as np
import matplotlib.pyplot as plt

x=np.linspace(-8.0*np.pi,8.0*np.pi,10000)

y1=x*np.sin(x)
y2=np.cos(x)+np.sin(x)

plt.plot(x,y1,label="y1=x*np.sin(x)")
plt.plot(x,y2,label="y2=np.cos(x)+np.sin(x)")

plt.legend(loc="upper center",shadow=True) #显示图例
#loc可以取值best(默认，最佳)
#upper right(上右)、upper left(上左)、upper center(上中)
#lower right(下右)、lower left(下左)、lower center(下中)
#center(中)、center right(中右)、center left(中左)
#right(右)

plt.show()

#eof
```

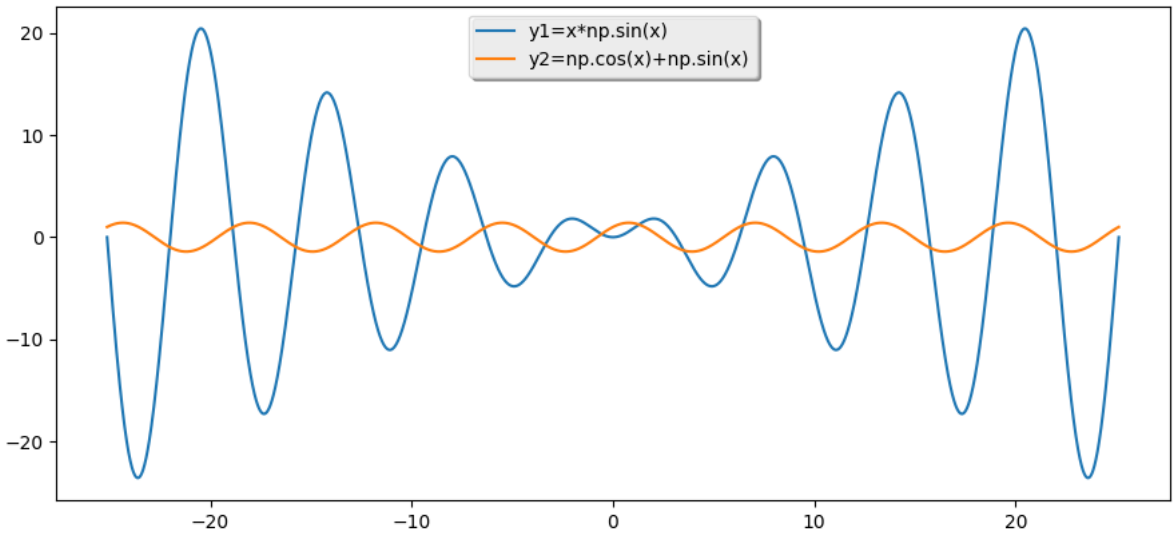

图2-14　例程2-14执行效果

5、文本标注和箭头

生成图形时，有时会在图上标注文字，有时还会用箭头清晰指明标注文字和曲线对应关系。添加文本用pyplot.text()或subplot.text()，生成箭头用pyplot.annotate()或subplot.annotate()。

第三节　图表类型

1、柱状图

例程2-15

```
import matplotlib.pyplot as plt

qiuContent="""Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
```

```
第6行  Complex is better than complicated.
第7行  Flat is better than nested.
第8行  Sparse is better than dense.
第9行  Now is better than never.
第10行 """
第11行
第12行 #删除换行、句号、逗号和问号
第13行 qiuContent=qiuContent.lower().replace("\n","").replace(" ","").replace(".","").replace(" ? ","")
第14行 hzCountMap={}
第15行
第16行 for i in qiuContent:
第17行     if i not in hzCountMap:
第18行         hzCountMap[i]=1
第19行     else:
第20行         hzCountMap[i]+=1
第21行
第22行 hzList=list(sorted(hzCountMap.items(),key=lambda x:x[1],reverse=True))
第23行 y=[y[1] for y in hzList]
第24行 hz=[y[0] for y in hzList]
第25行
第26行 plt.bar([x for x in range(len(y))],y,tick_label=hz)
第27行 plt.show()
第28行
第29行 #eof
```

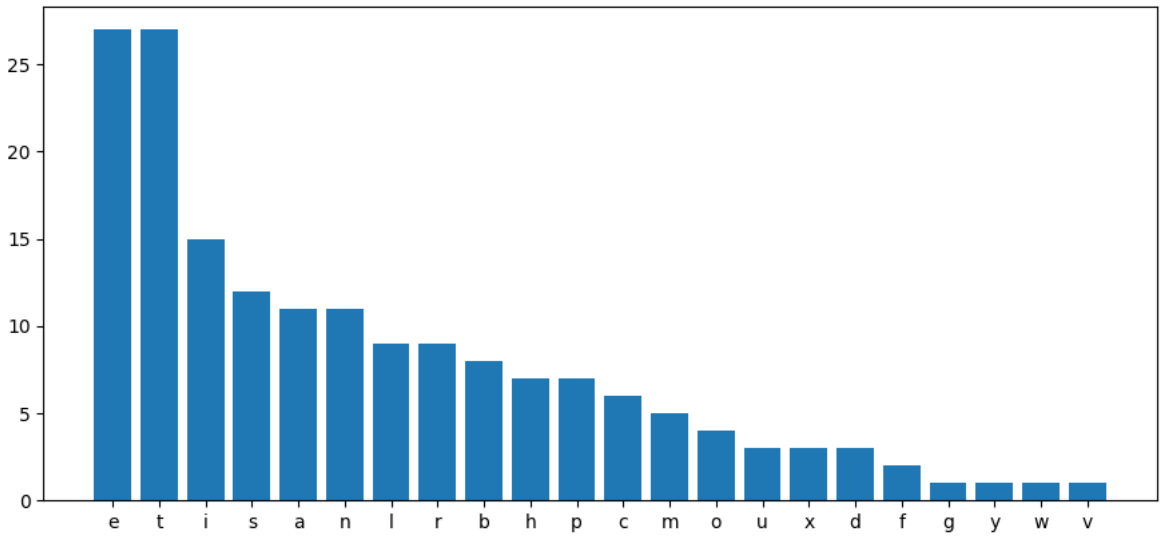

图2-15　例程2-15执行效果

2、饼图

3、散点图

第四节　小结