

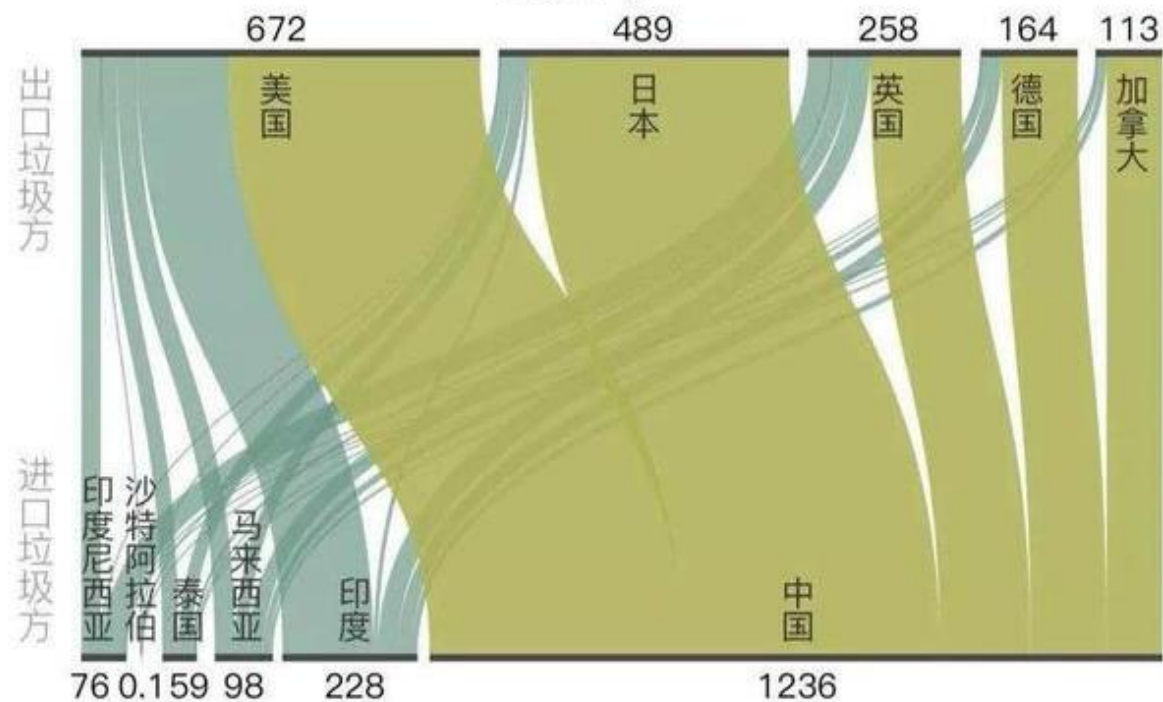
第14章

数据可视化

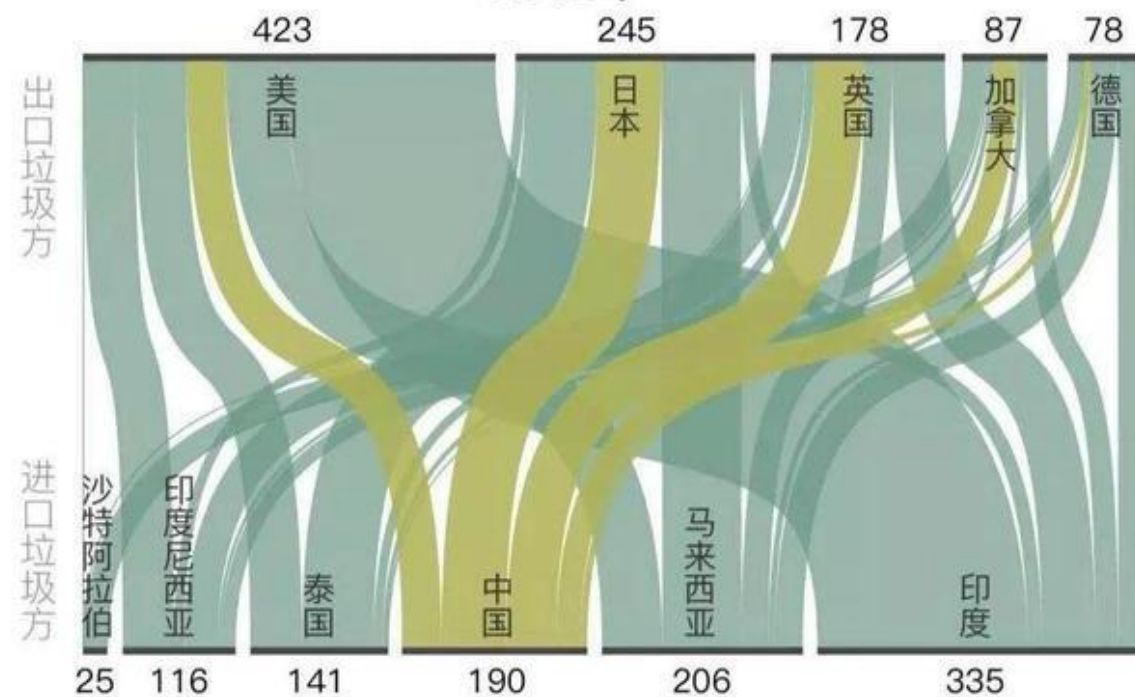
亚洲各国垃圾进口量变化

(单位: 百万美元)

2017年



2018年



Matplotlib 是第一个 Python 可视化程序库。

Matplotlib对于入门者而言还是比较容易上手的，只需几行代码即可生成绘图，包括我们常见的直方图，条形图，折线图，散点图等。

```
import matplotlib
```

```
import matplotlib.pyplot as plt #pyplot是最重要的子包
```



14.1概述



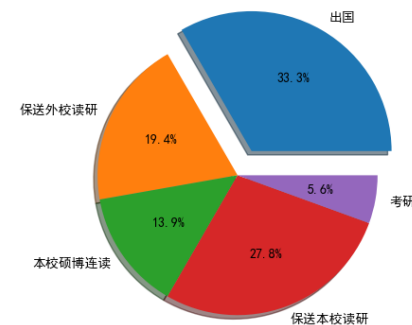
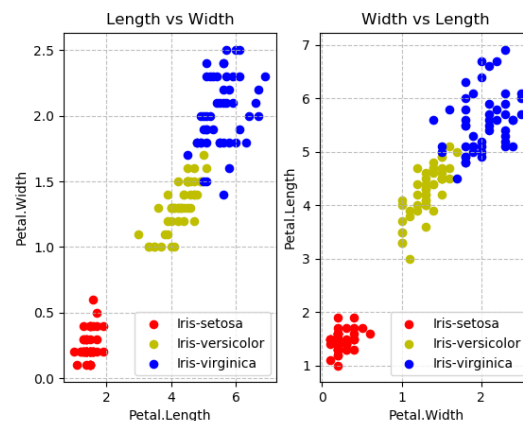
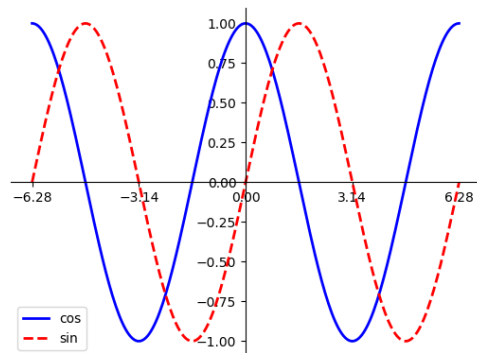
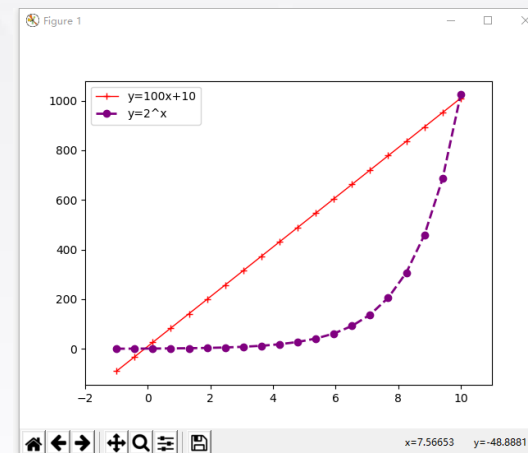
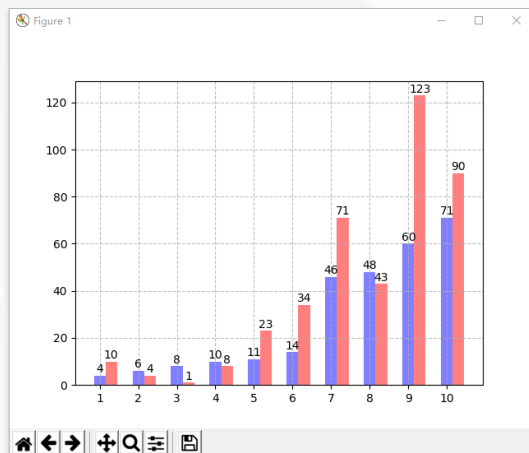
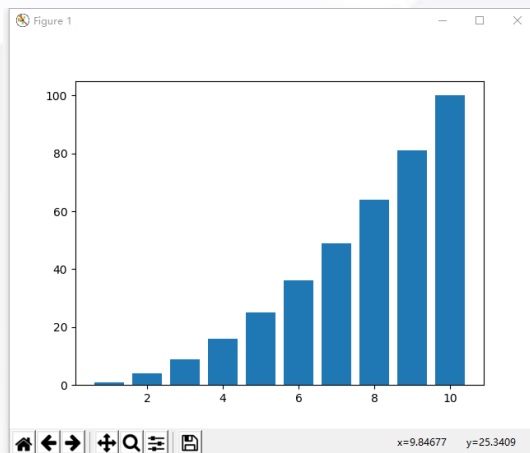
脚本层pyplot

艺术家层artist

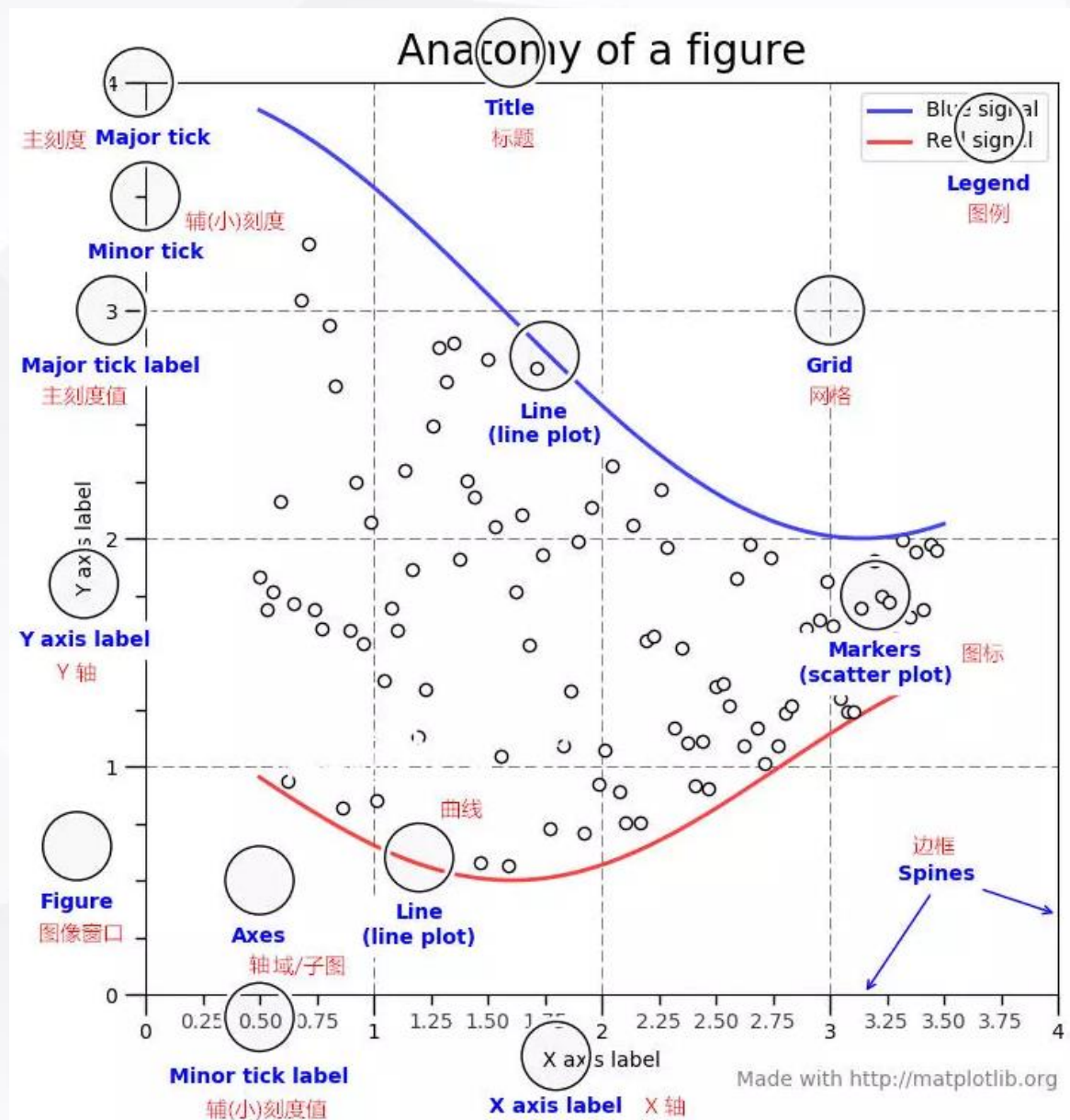
后端层backend



绘制直方图、线图、散点图、饼图等



14.1 Matplotlib的基本框架



序号	对象名称	中文名称	说明
1	Figure	图像窗口	用于展示图形的最外层窗口
2	Axes	子图	带有数据的图像区域。一个 Figure 中可以有多 个子图，但至少要有有一个能够显示内容的子图。
3	Title	标题	子图中的标题(Figure 也可以有标题)
4	Legend	图例	各种符号和颜色所代表内容与指标的说明，一般 位于边角上
5	Text	注释文本	在图内用文字对图像进行注解
6	Grid	网格线	在子图中用于指示刻度或数值的辅助线
7	Axis	坐标轴	一般包含X轴和Y轴
8	Lable	标签	一般是在轴、刻度等对象之上的文字说明
9	Tick	刻度	轴上的刻度



一、 设置绘图风格

```
plt.style.use('classic') #设置图像的风格为经典风格，也是默认的风格
```

使用此命令后，同一个python进程中的其它图像的绘制也都会使用这种风格。

可以用`plt.style.available`命令查看所有可用的风格。例如有'`bmh`'、'`dark_background`'、'`seaborn-dark`'等风格。

二、 创建图像和坐标轴

```
fig=plt.figure() #创建图像
```

```
ax=plt.axes() #创建坐标轴对象
```

```
fig,ax=plt.subplots() #同时创建出图像和坐标轴的实例
```



三、 设置坐标轴的上下限

```
ax.set_xlim(min,max) #设置x轴上下限
```

```
ax.set_ylim(min,max) #设置y轴上下限
```



四、设置图像标题

```
plt.title("title_name") 或 ax.set_title("title_name")
```

#设置图像的标题为双引号中的字符串



五、设置坐标轴标签

```
ax.set_xlabel("label_name") #设置x轴上的标签
```

```
ax.set_ylabel("label_name") #设置y轴上的标签
```



六、设置图例

```
plt.legend(["y=100x+10","y=2^x"],loc='upper left')
```

#在左上角显示图例

```
ax.legend([line1,line2,line3],["label1","label2","label3"],loc='lower right')
```

#给三条线分别设置图例，位置在右下角



七、添加文字

```
ax.text(x,y,s) #在(x,y)坐标处添加文字串s
```



八、添加注释

```
ax.annotate("annotation",xy=(1,2),xycoords='axes fraction',xytext=(2,3),  
textcoords='axes fraction',arrowprops=dict(arrowstyle="->"))
```

#(x,y)坐标处为箭头的位置, xycoords为箭头的坐标体系, xytext为注释文字起始的坐标, textcoords为注释文字的坐标体系, arrowstyle为箭头样式



九、隐藏边框

```
ax.spines["top"].set_visible(False) #隐藏上边框  
#同理，隐藏下边框、左边框和右边框的参数分别为"bottom"、"left"  
#、"right"。
```



十、隐藏坐标轴（刻度和刻度值）

```
ax.set_xticks([ ]) #隐藏x轴刻度和刻度值
```

```
ax.xaxis.set_major_formatter( plt.NullFormatter() ) #只需隐藏刻度值,  
同时保留刻度
```



十一、设置坐标轴刻度和刻度标签

```
ax.set_xticks([1,2,3]) #设置x轴刻度为1,2,3
```

```
ax.set_yticks([1,2,3]) #设置y轴刻度为1,2,3
```

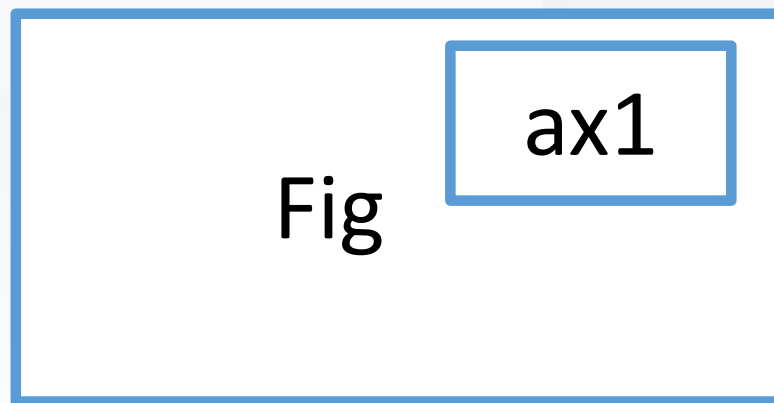
```
ax.set_xticklabels(["one","two","three"]) #设置x轴刻度标签为  
one,two,three
```

```
ax.set_yticklabels(["one","two","three"]) #设置y轴刻度标签为  
one,two,three
```



十二、创建多个图像

大图套小图

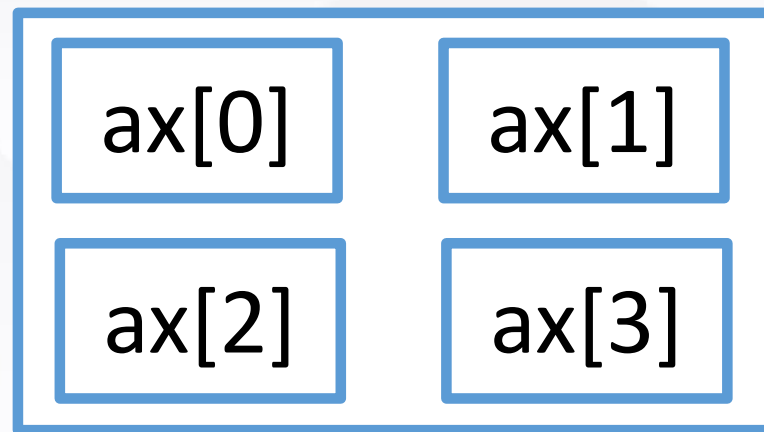


```
ax1=fig.add_axes([left, bottom, width, height])
```

[left, bottom, width, height]用来设置新坐标轴的位置和大小。

十二、创建多个图像

规则网格图



```
fig,ax=plt.subplots(num_of_rows,num_of_columns,sharex=True,sharey=True)
```

#创建出几行几列的网格图，可以用`ax[0].plot()`，`ax[1].plot()`等相继在各网格上画图。

```
ax1 = fig.add_subplot(121)
```

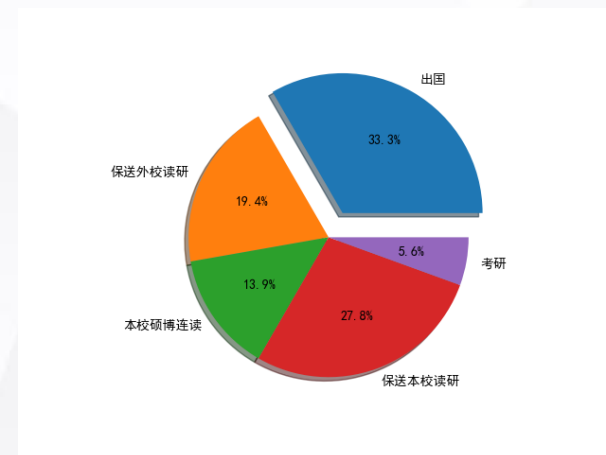
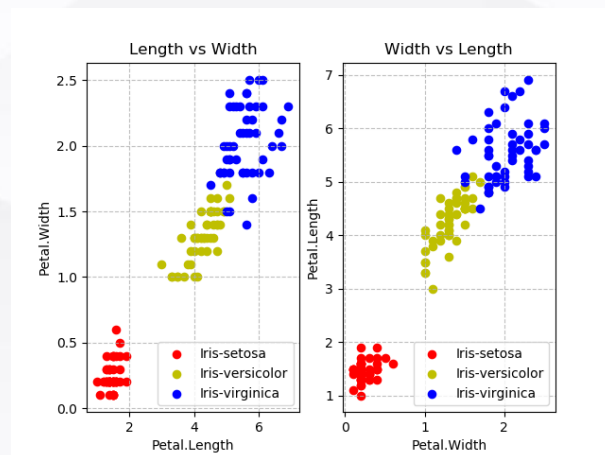
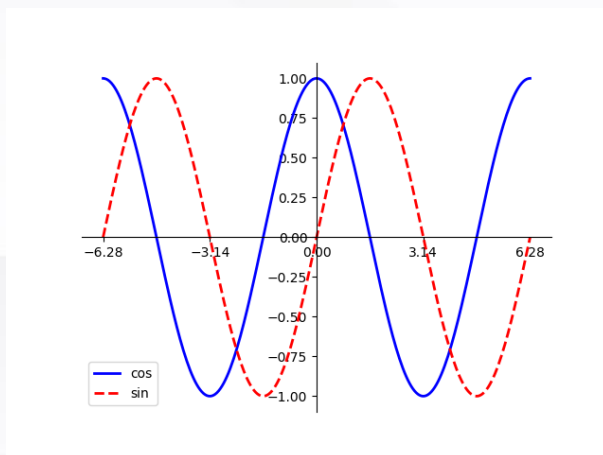
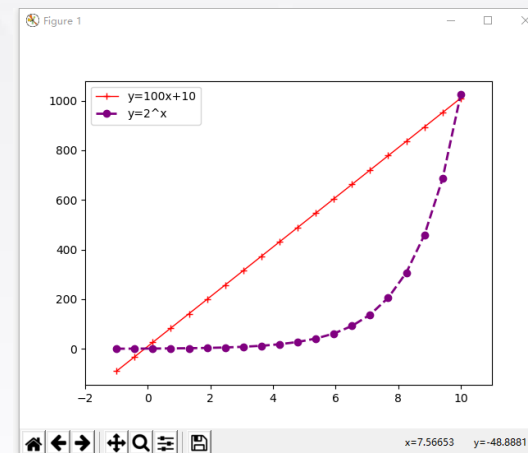
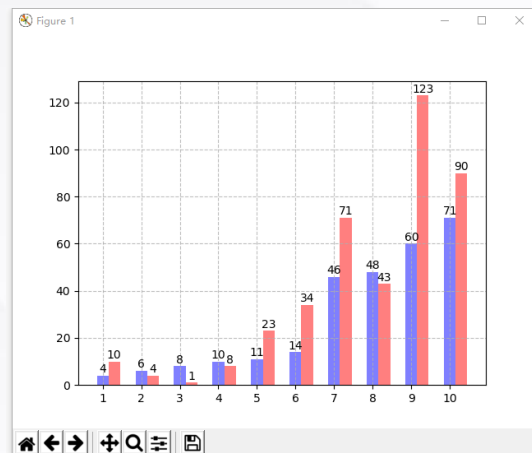
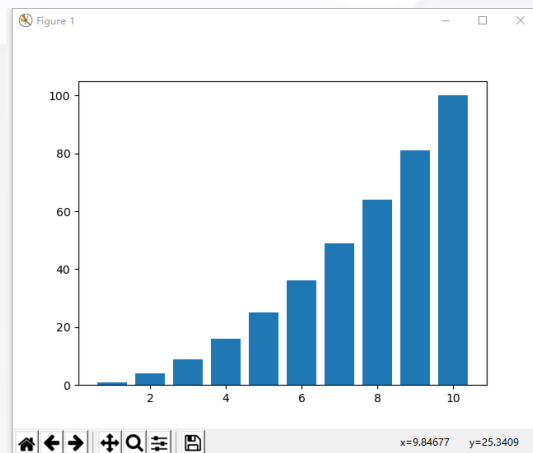
```
ax2 = fig.add_subplot(122)
```

十三、设置对中文的支持

```
matplotlib.rcParams['font.sans-serif'] = ['SimHei']
```



绘制直方图、线图、散点图、饼图等



十四、保存图像

```
fig.savefig('file_name.png') #将图像保存到当前文件夹的file_name.png  
文件之中。
```

注意：此命令必须放在plt.show()之前，否则保存的将会是空白图像。

可以通过fig.canvas.get_supported_filetypes()查看系统支持的文件格式，常用的包括pdf、png、svg等文件格式。



十五、显示图像

```
plt.show()
```

在一个python进程中只能使用一次plt.show(), 因此通常把此命令放在程序的最后, 等所有参数全部设置完毕后再显示图像。



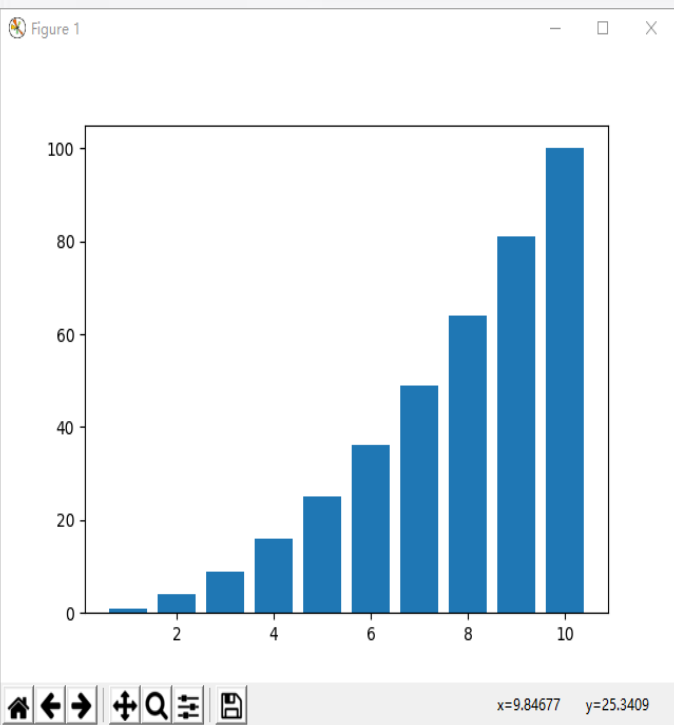
第14章 数据可视化

4.直方图

14.1.4 直方图

直方图，也被称为柱形图或条形图，是使用得最为广泛的图形。

```
pyplot.bar(x, height, width=0.8, bottom=None, *, align='center')
```



- x表示在横轴上的数据，height是直方图的高度，这两个是必须需要的参数，一般情况下这两个参数都是一个列表或者序列的形式。
- width是指的直方图的宽度，默认情况下是0.8，即最大宽度的80%。
- bottom参数指的是直方图的底部在y坐标上的起点值，默认情况下是0。
- *表示其它参数，例如颜色、边缘颜色、线宽、刻度标签等。
- align参数是表示直方图与数据的对齐关系，默认为居中对齐方式。可以使用'edge'参数表示从x数据的位置左对齐。

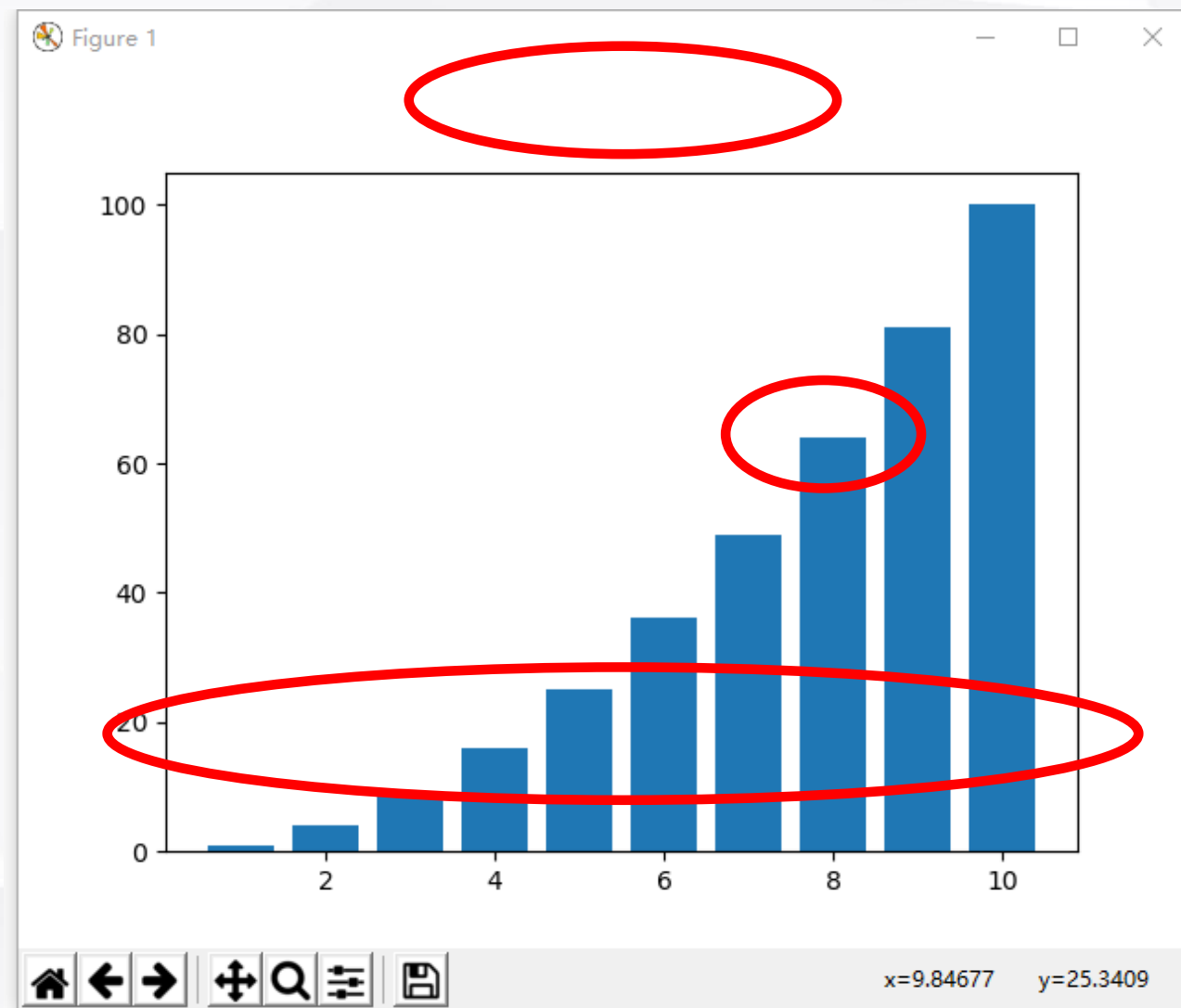
例1：简单的直方图示例

```
#demo1401:
import numpy as np
import matplotlib.pyplot as plt

x=np.array([1,2,3,4,5,6,7,8,9,10]) #创建一个numpy数组x
y=x*x #创建一个numpy数组y, 内容为x中数据的平方值
plt.bar(x,y) #调用bar函数画直方图
plt.show() #显示图像
```



14.1.4 直方图

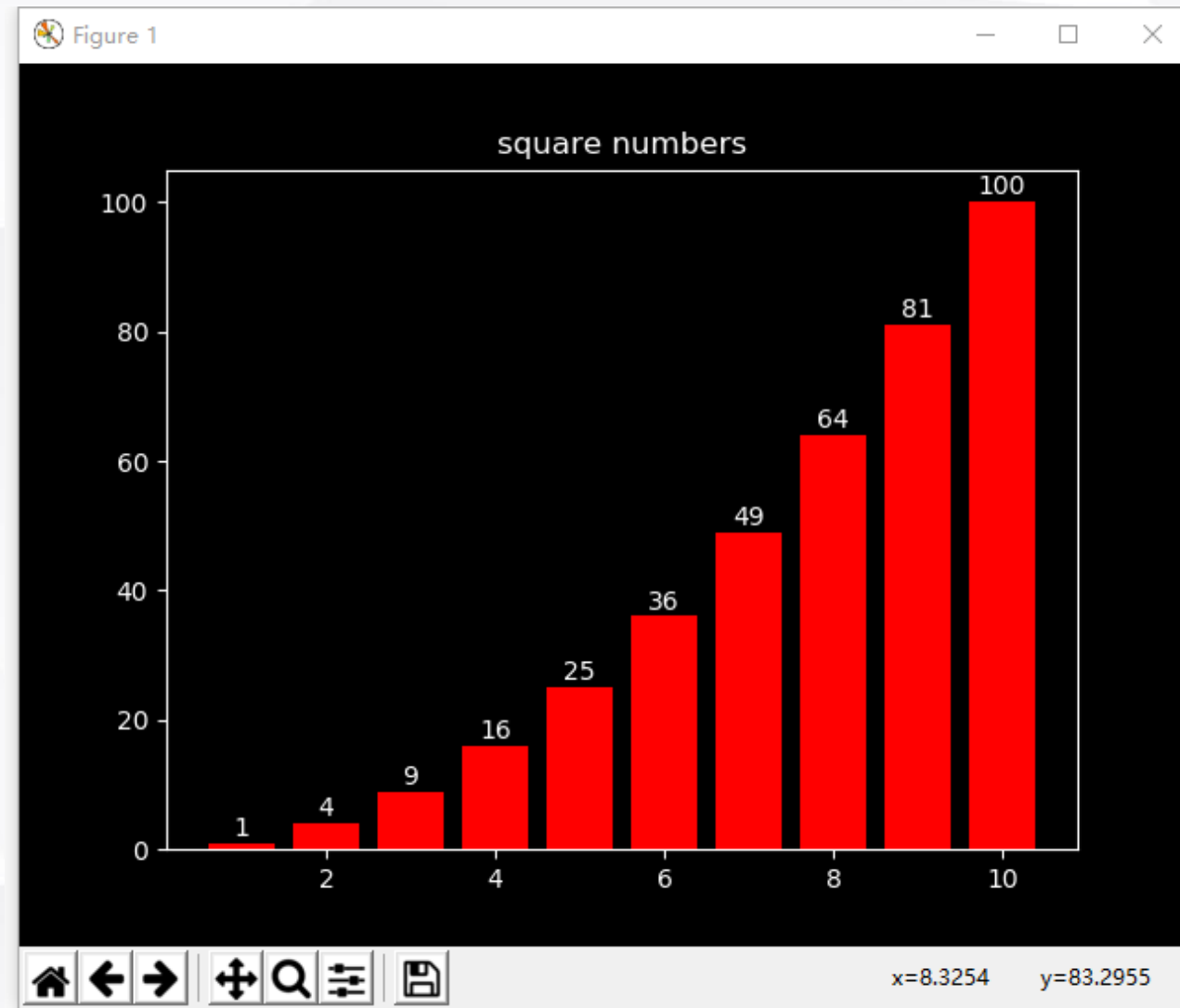


14.1.4 直方图

```
#demo1401-1:
plt.style.use('dark_background') #设置图像风格
fig,ax=plt.subplots()
ax.set_title("square numbers")
x=np.array([1,2,3,4,5,6,7,8,9,10]) #创建一个numpy数组x
y=x*x #创建一个numpy数组y, 内容为x中数据的平方值
plt.bar(x,y,color='r') #bar的颜色改为红色
for a,b in zip(x,y): #在直方图上显示数字
    plt.text(a,b+0.2,'%d'%b,ha = 'center',va = 'bottom',fontsize=10)
plt.show()
```



14.1.4 直方图



作业1

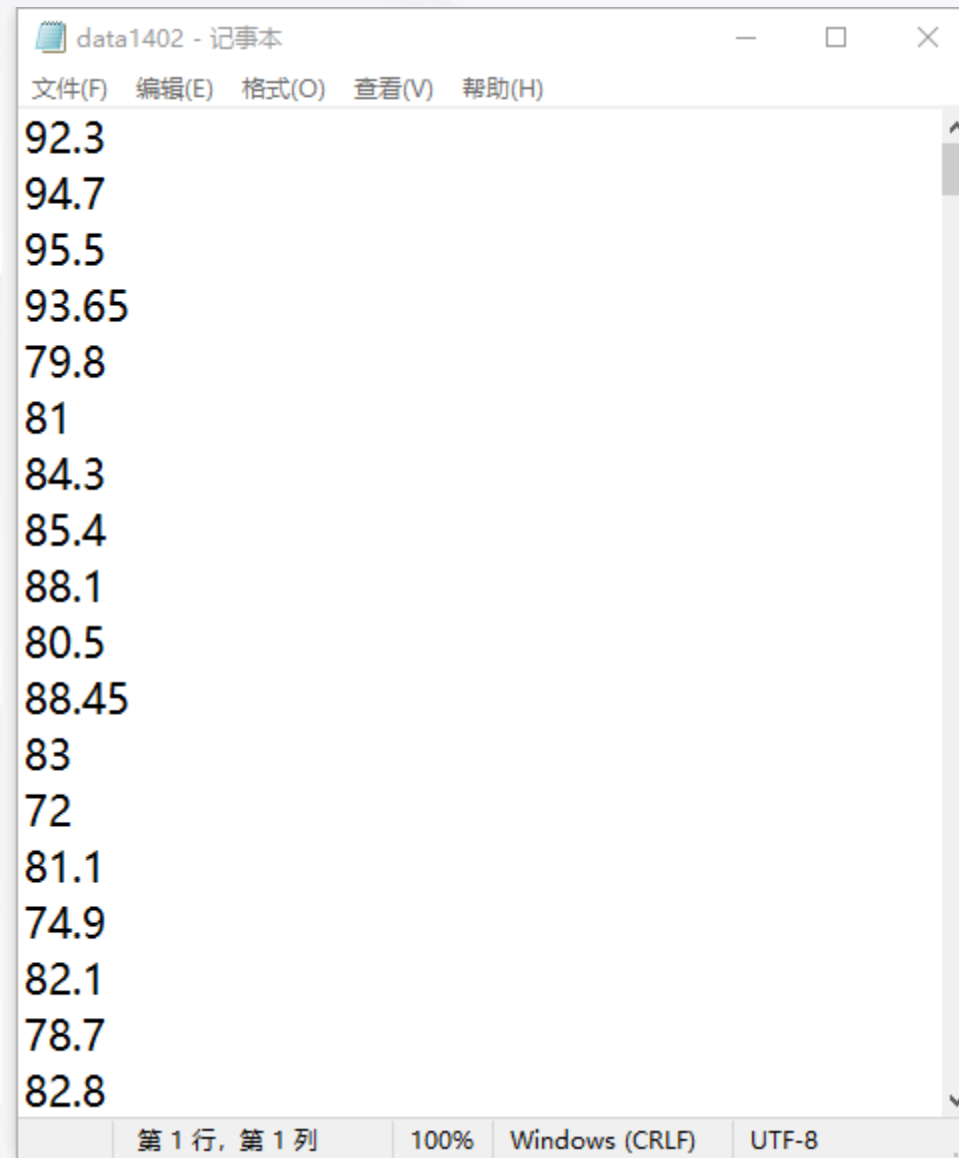
1. 请更换图形的风格
2. 请将x轴的数据改为-10到10
3. 请将y函数改为x的立方
4. 请将直方图上的数字，位置改到柱形图的内部



例2：数据的分布

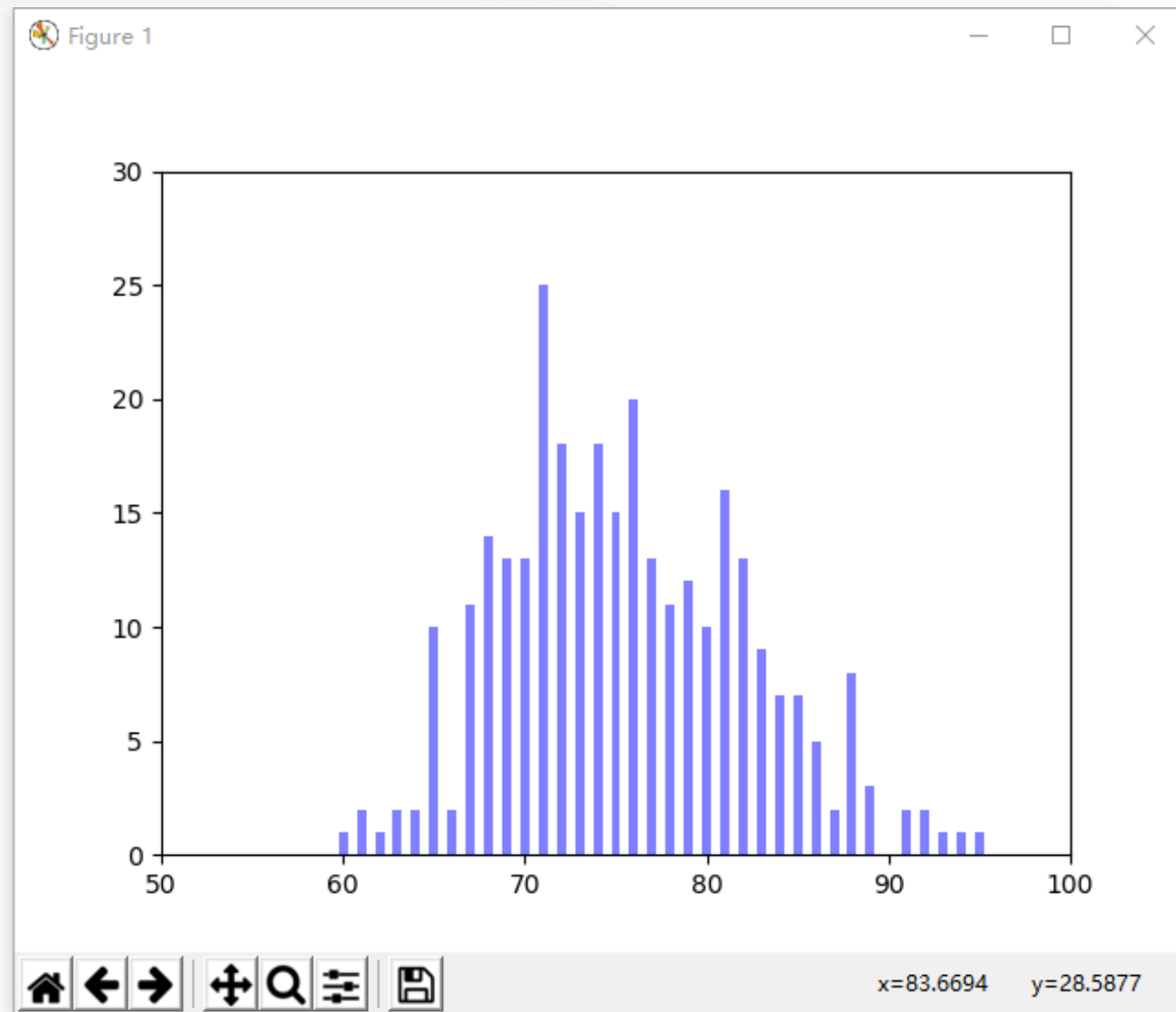
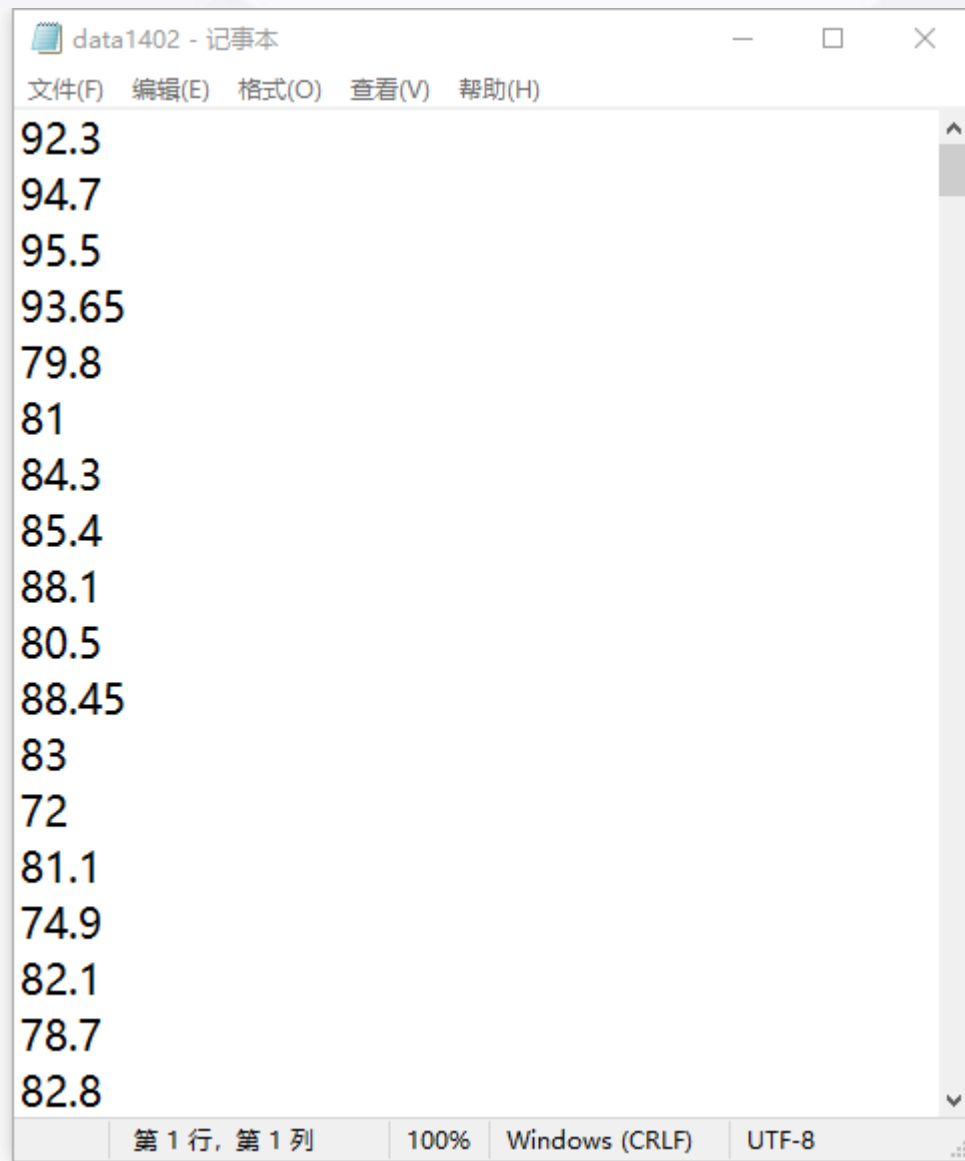
我们需要查看某一门课程的分数分布情况，给你一门课所有同学的分数，要求展示出每个整数分数对应的学生人数。

我们事先把分数保存到了一个data1402.csv文件中，每一行是一个同学这门课的成绩。我们的数据大概有300行左右，分数的范围集中在60-100分之间，每一行是一个小数。



```
data1402 - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
92.3
94.7
95.5
93.65
79.8
81
84.3
85.4
88.1
80.5
88.45
83
72
81.1
74.9
82.1
78.7
82.8
第 1 行, 第 1 列 100% Windows (CRLF) UTF-8
```

14.1.4 直方图



例2：数据的分布

1. 读入文件，每次读取一行，将每一行数据保存到一个数组scores中；
2. 创建一个新的数组，通过汇总计算scores中的数据，在新数组中存放每个分数对应的人数，第N个元素就是该分数对应的人数。
3. 利用直方图进行展示，其中每个直方柱的x值就是分数，y值就是人数。



例2：数据的分布

1.读入文件，每次读取一行，将每一行数据保存到一个数组scores中；

```
#demo1402:
BasePath='c:\\code' #csv文件的保存路径
import csv
scores=[] #创建一个列表对象
with open(BasePath+'\\data1402.csv', 'r') as csvfile:
    f_csv = csv.reader(csvfile) #读入文件
    for row in f_csv: #将每一行的数据保存到scores中
        scores.append(float(row[0]))
```



例2：数据的分布

1. 读入文件，每次读取一行，将每一行数据保存到一个数组scores中；
2. 创建一个新的数组，通过汇总计算scores中的数据，在新数组中存放每个分数对应的人数，第N个元素就是该分数对应的人数。
3. 利用直方图进行展示，其中每个直方柱的x值就是分数，y值就是人数。

问题：分数集中在60-100之间，如果使用数组，需要开100个元素的空间，但实际又用不到，会造成空间的浪费。如果只开40个空间，又涉及到比较麻烦的转换。

字典对象

{[键值1:数值1, 键值2:数值2, ..., 键值n:数值n]}

例如:

```
d1={1001:"张三", 1002:"李四",1003:"张三"}
```

get(k,[v]) #参数v为可选项, 为查找不到时的返回值

```
d1.get(1002)
```

```
d1.get(1004,"查无此人")
```



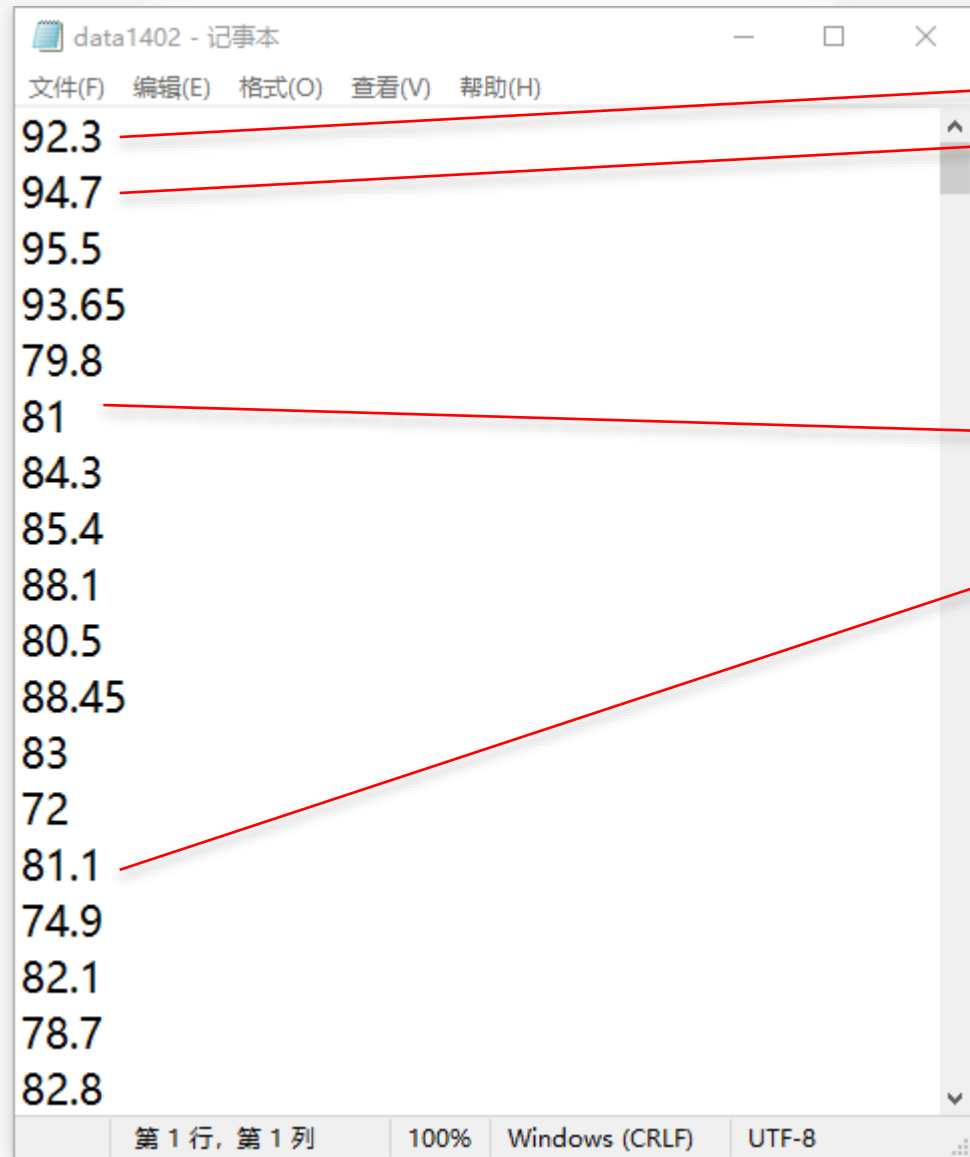
例2：数据的分布

2.创建一个字典对象scorescount，用于记录每个整数分数对应的人数。对scores中的每一个元素（小数），先转换为对应的整数。这个整数即是scorescount中的键值，数值是在原有的基础上加1；

```
def count_elements(scores): #定义转换函数，统计每个分数值对应多少人数
    scorescount = {} #定义一个字典对象
    for i in scores:
        scorescount[int(i)] = scorescount.get(int(i), 0) + 1 #累加每个分数值的人数
    return scorescount
```



14.1.4 直方图



92: 1,
94: 1,
95: 1,
93: 1,
79: 1,
81: 2,
84: 1,
85: 1,
88: 2,
80: 1,
83: 1,
72: 1,
74: 1,
82: 2,
78: 1



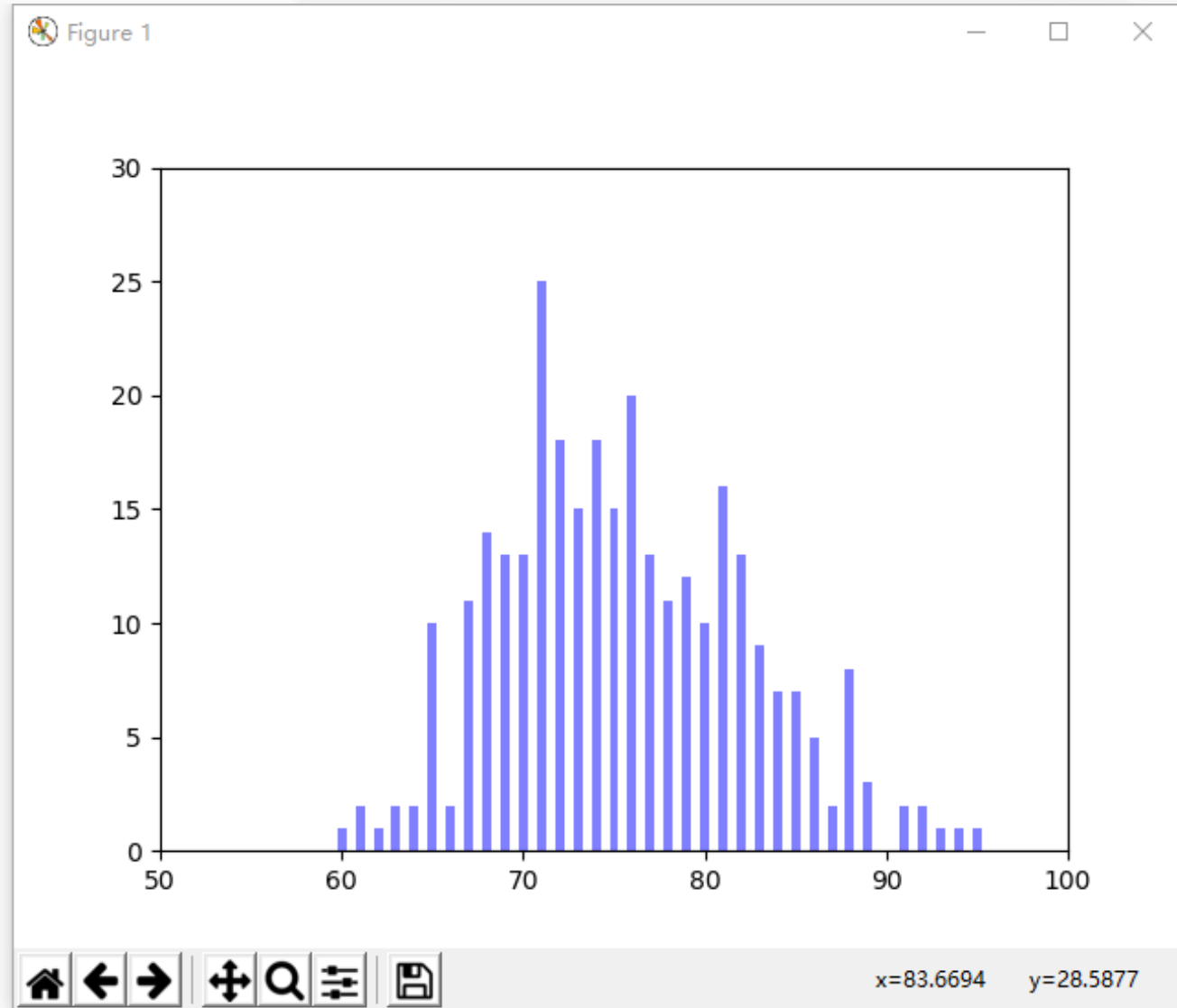
例2：数据的分布

3.利用直方图进行展示，其中每个直方图的x值就是scorescount元素的键值，Y值就是scorescount元素的数值。

```
counted = count_elements(scores)
plt.axis([50,100,0,30]) # 设置x轴和y轴的最小和最大值
plt.bar(counted.keys(),counted.values(),0.5,alpha=0.5,color='b')
# 绘制直方图，第三个参数表示直方图的宽度，alpha为透明度,color为颜色
plt.show()
```



14.1.4 直方图



作业2

1. 请自己创建出一些数据（范围在0-1000，数据量不少于1万），并通过直方图看看这些数据的分布情况。
2. 如果需要将每5分作为一个分数段，展示出每个分数段的人数，如何修改程序？



例3：排名的变化

每个学期结束之后，我们都要对学生的排名进行统计。现在希望看看某个班的10名学生，在全年级200人当中的两个不同学期的排名变化情况。数据保存在data1403.csv文件之中，逗号之前的第一列表示本学期的排名，逗号之后的第二列表示上学期的排名。

```
4,10  
6,4  
8,1  
10,8  
11,23  
14,34  
46,71  
48,43  
60,123  
71,90
```


例3：排名的变化

1. 读入文件，每次读取一行，将每一行的2个数据分别保存到数组Semester1和数组Semester2中；
2. 创建2组直方图，但要注意它们x轴上的位置，不能互相重叠
3. 在每个直方柱上标注数字



1.读入文件，每次读取一行，将每一行的2个数据分别保存到数组Semester1和数组Semester2中；

```
#demo1403:
fig,ax=plt.subplots()
plt.rcParams['font.sans-serif'] = ['SimHei'] #添加对中文字体的支持
BasePath='c:\\code' #csv文件的保存路径
Semester1=[] #创建一个列表对象保存本学期的排名
Semester2=[] #创建一个列表对象保存上学期的排名
with open(BasePath+'\\data1403.csv', 'r') as csvfile:
    f_csv = csv.reader(csvfile) #读入文件
    for row in f_csv: #将每一行的数据分别保存到Semester1和Semester2中
        Semester1.append(int(row[0]))
        Semester2.append(int(row[1]))
```



14.1.4 直方图

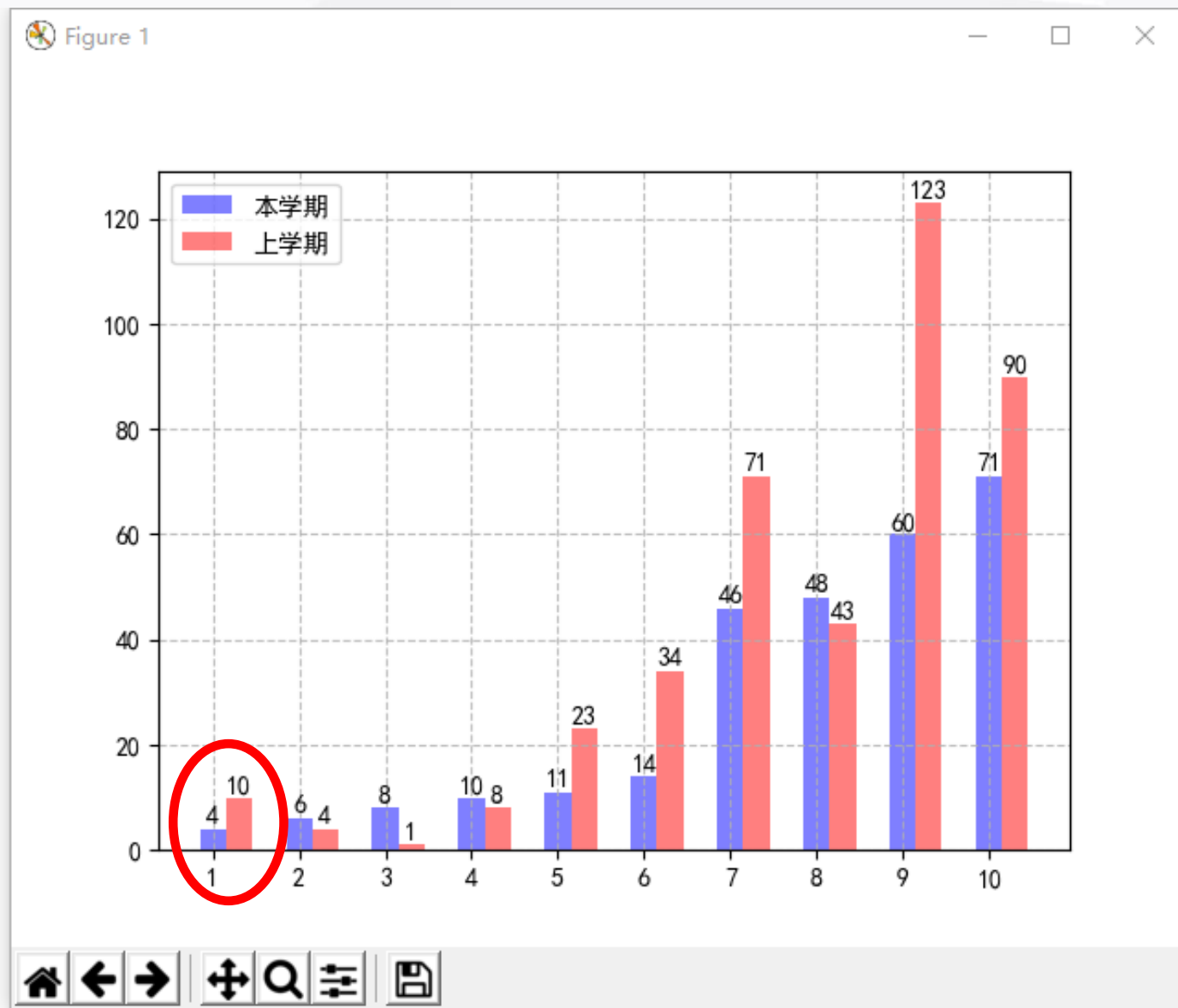
2. 创建2个直方图，但要注意它们x轴上的位置，不能互相重叠
3. 在每个直方图上标注排名的数字

```
x=np.arange(1,11) #生成横轴数据
plt.bar(x,Semester1,0.3,alpha=0.5,color='b') #生成本学期的排名直方图，宽度为0.3
plt.bar(x+0.3,Semester2,0.3,alpha=0.5,color='r')
#生成上学期的排名直方图，在上一个直方图的右侧0.3的距离显示

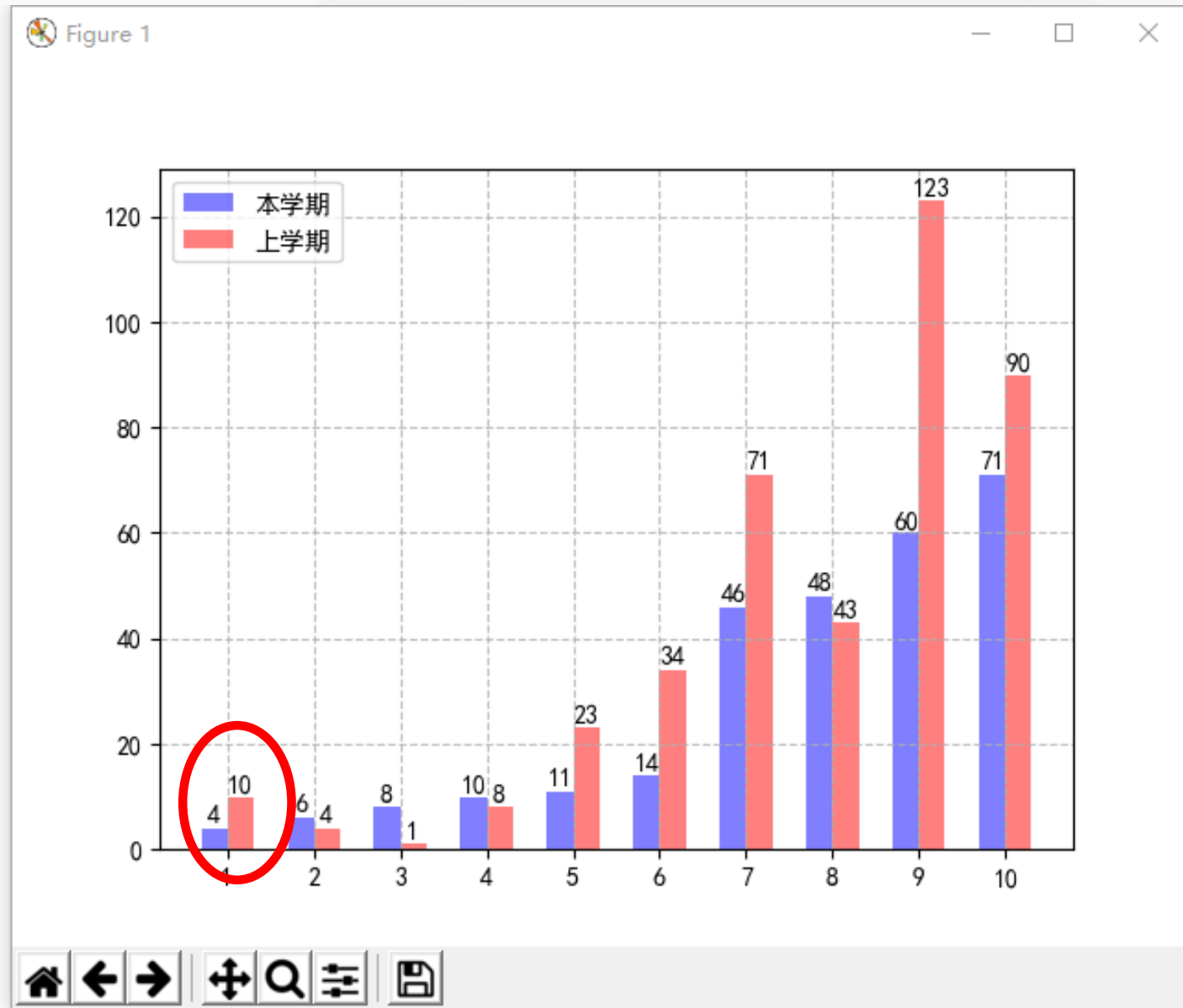
for a,b in zip(x,Semester1): #在直方图上显示本学期的排名数字
    plt.text(a,b+0.2,'%d'%b,ha = 'center',va = 'bottom',fontsize=10)
for a,b in zip(x,Semester2): #在直方图上显示上学期的排名数字
    plt.text(a+0.3,b+0.2,'%d'%b,ha = 'center',va = 'bottom',fontsize=10)
plt.legend(["本学期","上学期"],loc='upper left')
plt.grid(True, linestyle='--', alpha=0.8) #设置网格线
plt.show()
```



14.1.4 直方图



14.1.4 直方图

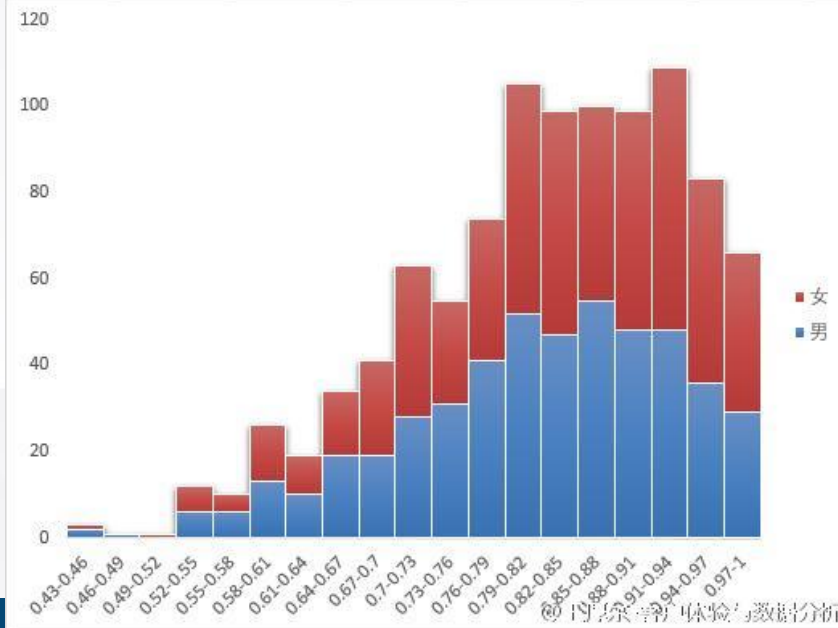
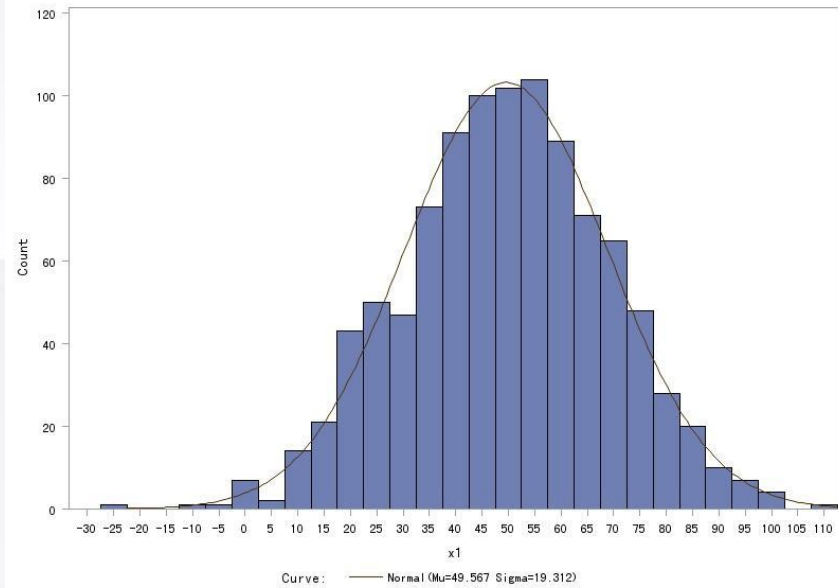


作业3

再创建一组新的数据，使得每个学生的数据有3个学期的排名，并通过直方图进行展示。



14.1.4 直方图



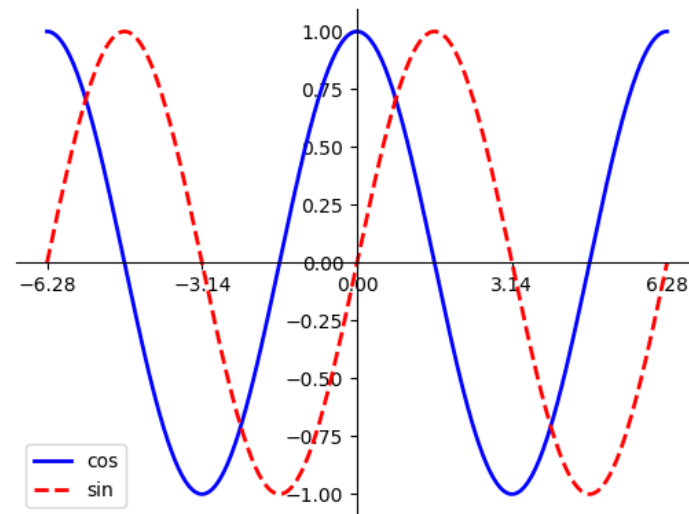
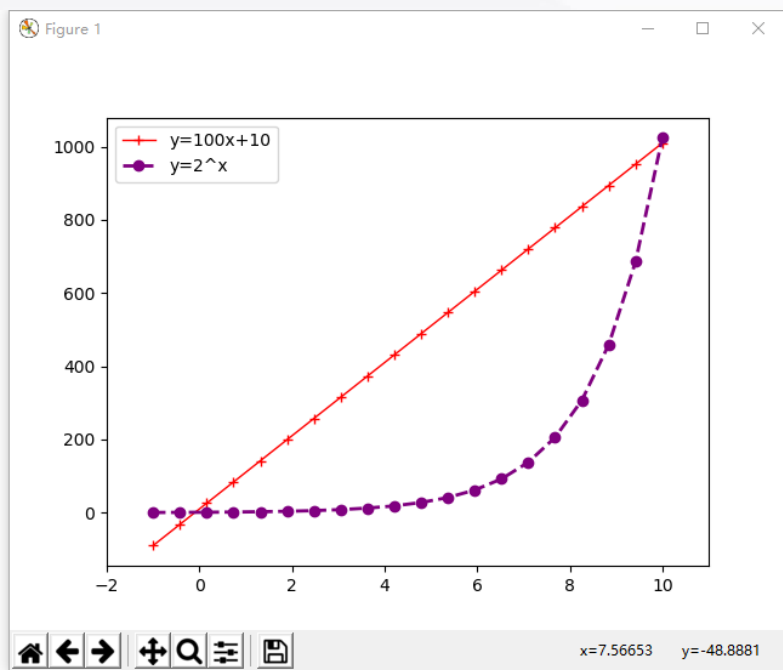
第14章 数据可视化

5.线图

14.1.5 线图

线图，也称为折线图，一般用来反映数据的变化趋势。

例如线性函数和指数函数的对比，正弦函数和余弦函数的图形对比，某个商品在一年之中销售量的变化情况等。



使用plot函数来绘制线图：

```
pyplot.plot(x, y, marker='o', color=blue, linestyle='-', linewidth=2.5,  
markersize=10)
```

- x,y为数据点所在的坐标位置，不可缺少。
- marker是数据点的样式，color是线条的颜色，linestyle是线条的形式，linewidth是线宽，markersize是数据点的大小。

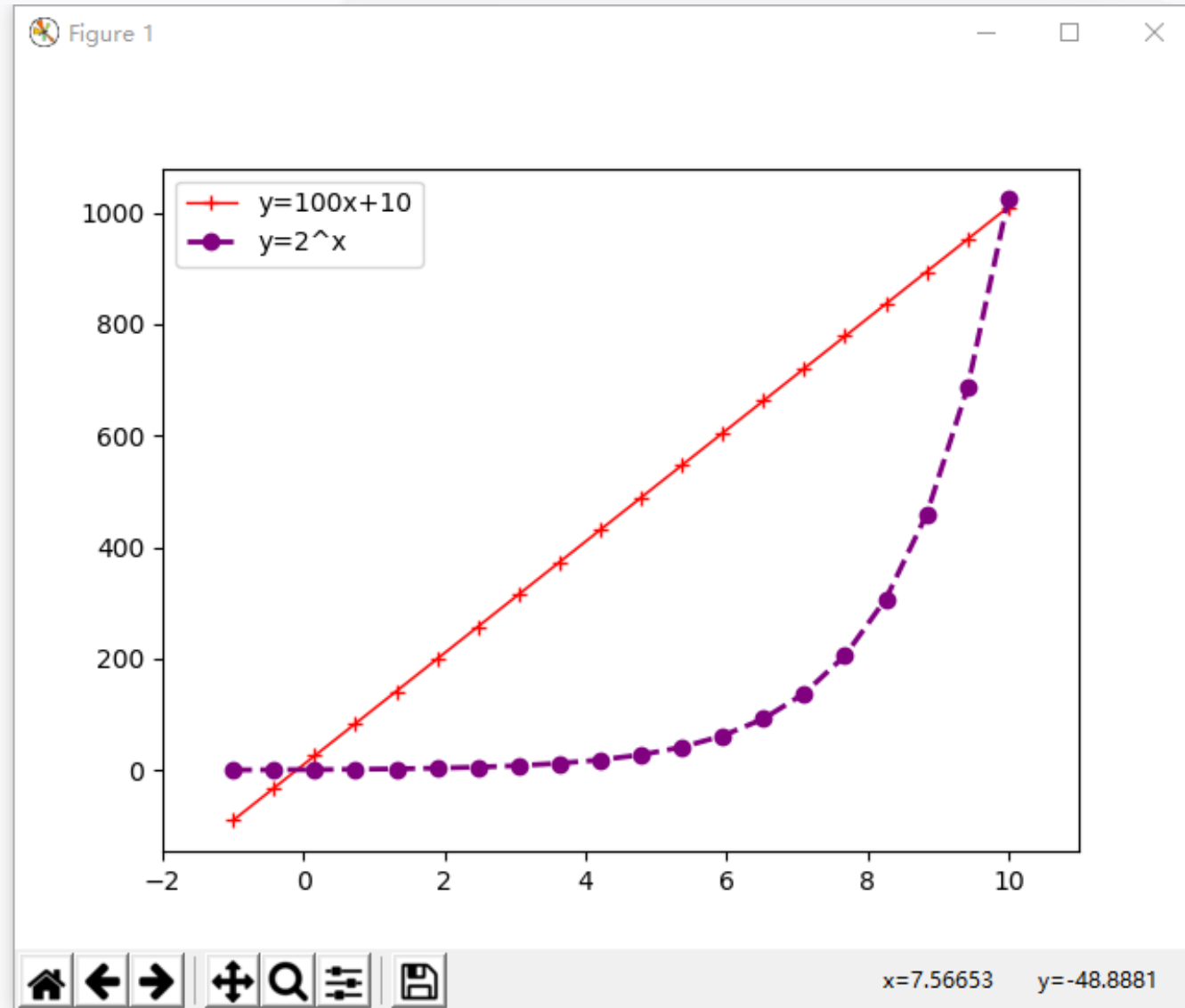


例1：线性函数和指数函数的对比

```
#demo1405:
x = np.linspace(-1, 10, 20) #在-1到10的区间内生成20个数据
y1 = 100* x + 10 #直线 y1
y2 = 2**x        #曲线 y2
plt.plot(x, y1, 'r+',color="red", linewidth=1.0, linestyle="-", label='line1')
# 绘制颜色为蓝色、宽度为 1 像素的连续直线 y1，数据点为+号形式
plt.plot(x, y2, 'bo', color="#800080", linewidth=2.0, linestyle="--", label='line2')
# 绘制颜色为紫色、宽度为 2 像素的不连续曲线 y2,数据点为圆点形式
plt.xlim(-2,11) # 设置横轴的最大最小值
ax.legend(["y=100x+10","y=2^x"],loc='upper left') #在左上角显示图例
plt.show()
```



14.1.5 线图



课堂小练习4

1. 增大 x 的取值范围
2. 调整线性函数和指数函数的参数

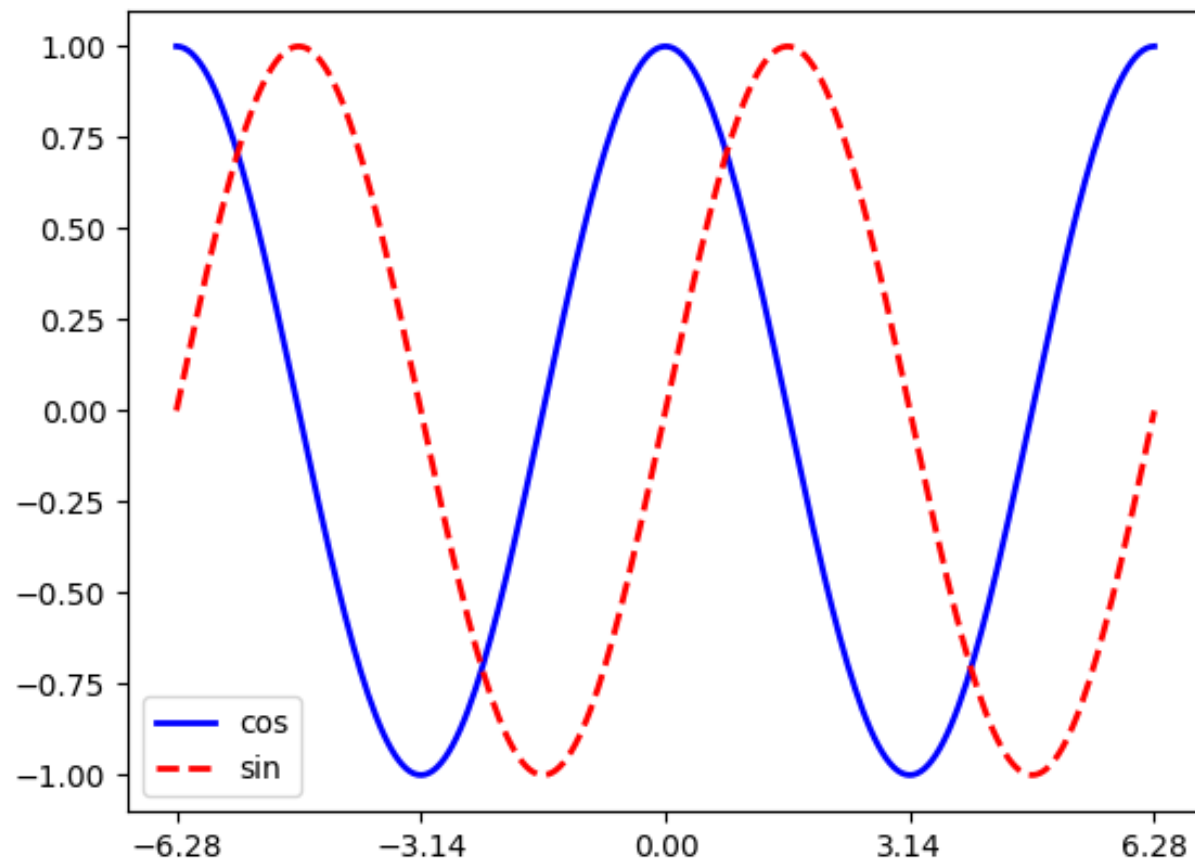


例2: sin函数和cos函数的对比

```
demo1406:
import numpy as np
import matplotlib.pyplot as plt
fig,ax=plt.subplots()
x = np.linspace(-2*np.pi, 2*np.pi, 256) #生成从-2π到2π的256个数据
cos,sin = np.cos(x), np.sin(x) #分别计算x的cos和sin函数值
ax.set_xticks( [-2*np.pi, -1*np.pi, 0, np.pi, 2*np.pi]) #设置x轴的刻度
plt.plot(x, cos, color="blue", linewidth=2, linestyle="-", label="cos")
#画出cos曲线, 颜色为蓝色, 线宽为2, 连续线
plt.plot(x, sin, color="red", linewidth=2, linestyle="--", label="sin")
#画出sin曲线, 颜色为红色, 线宽为2, 间断线
```



例2：线性函数和cos函数的对比



例2：线性函数和cos函数的对比

#画出十字形的坐标轴

ax.spines["right"].set_visible(False)

#隐藏右边框

ax.spines["top"].set_visible(False)

#隐藏上边框

ax.spines['bottom'].set_position(('data',0)) #设置下边框到y轴0的位置

ax.xaxis.set_ticks_position('bottom') #刻度值设置在下方

ax.spines['left'].set_position(('data',0)) #设置左边框到x轴0的位置

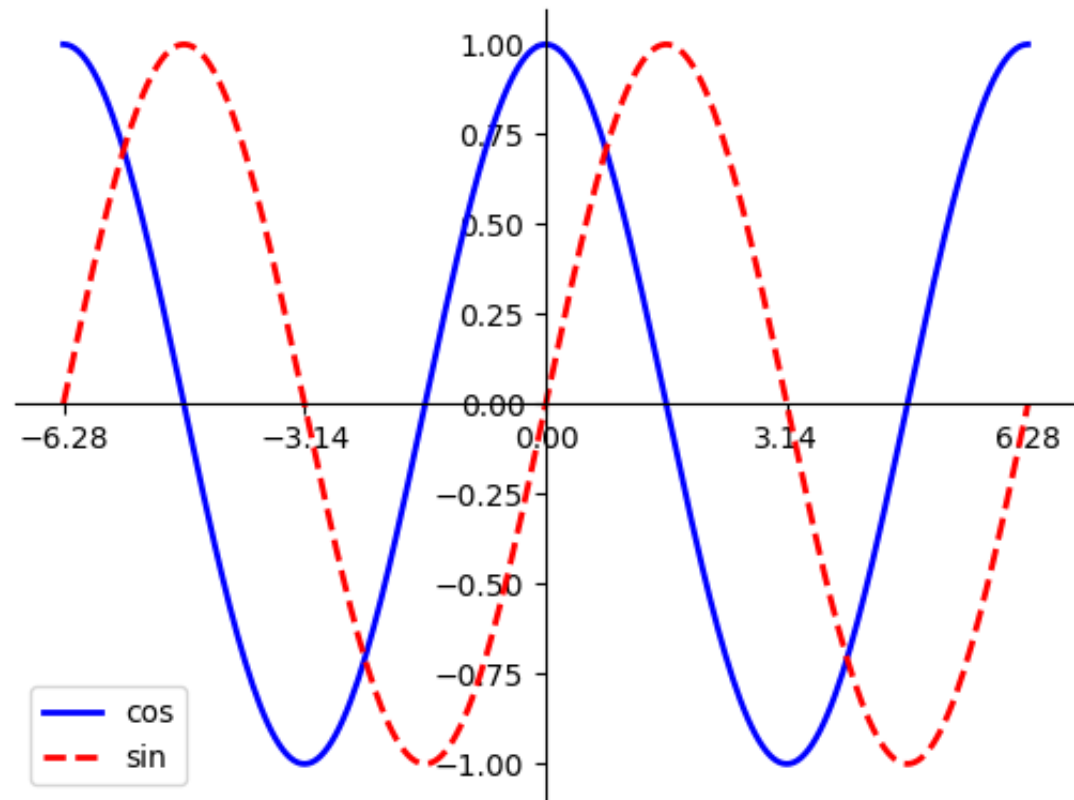
ax.yaxis.set_ticks_position('left') #刻度值设置在左侧

plt.legend(loc= 'lower left')

plt.show()

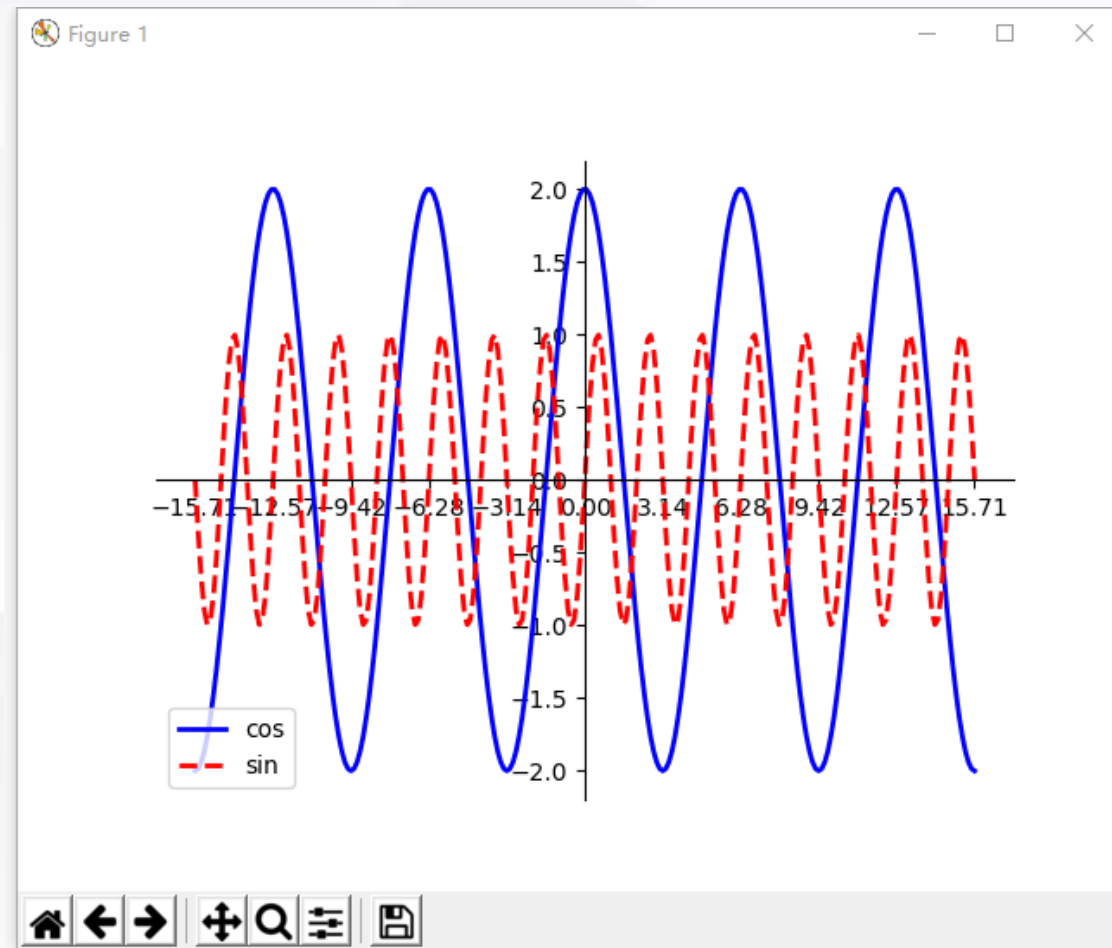


14.1.5 线图

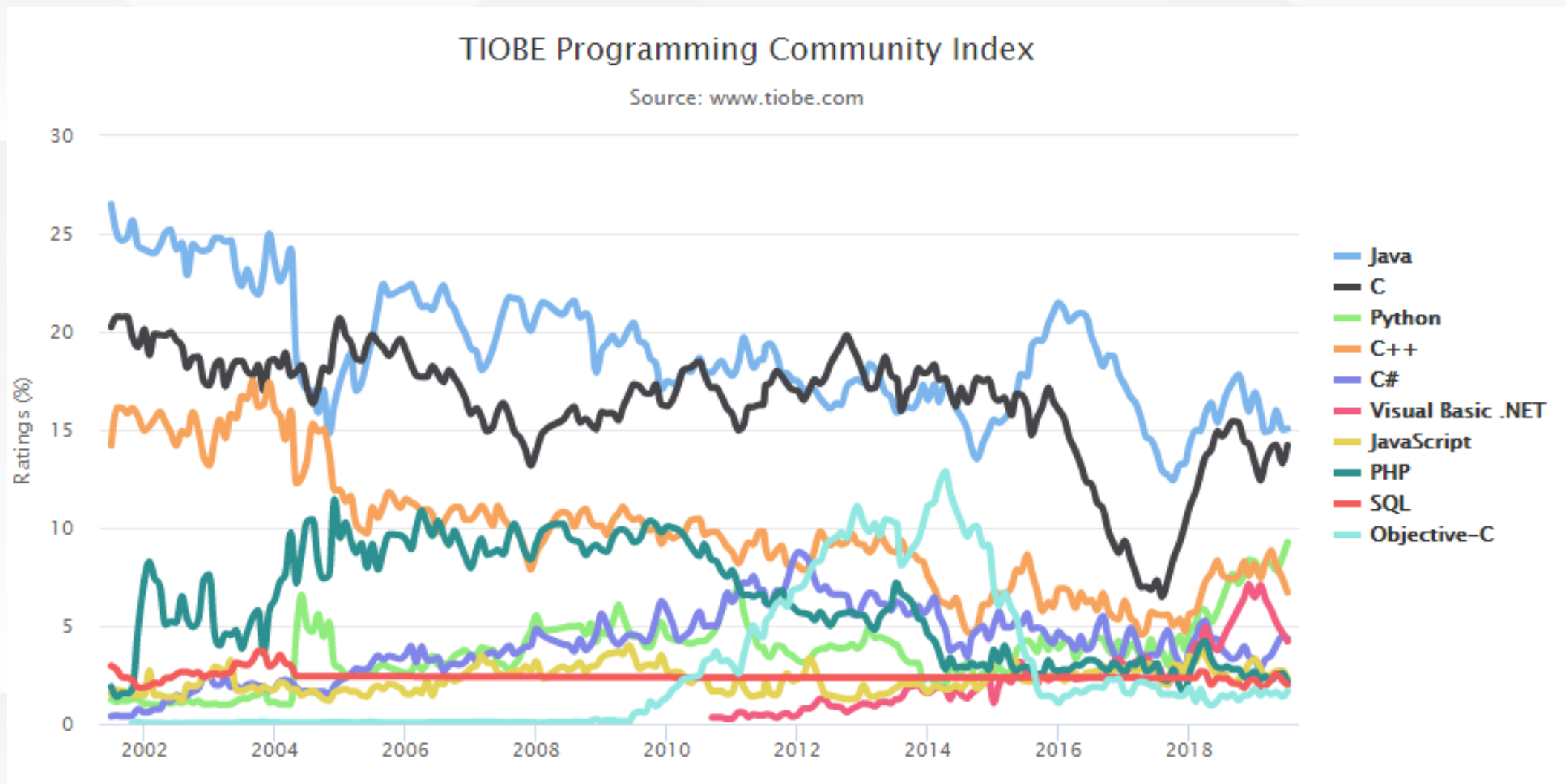


作业5

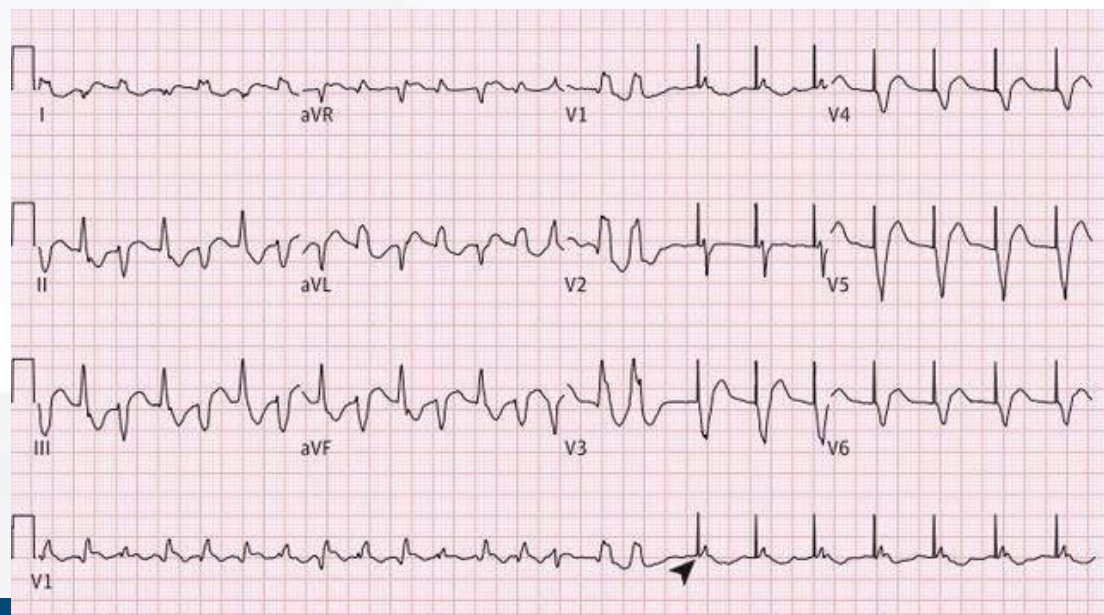
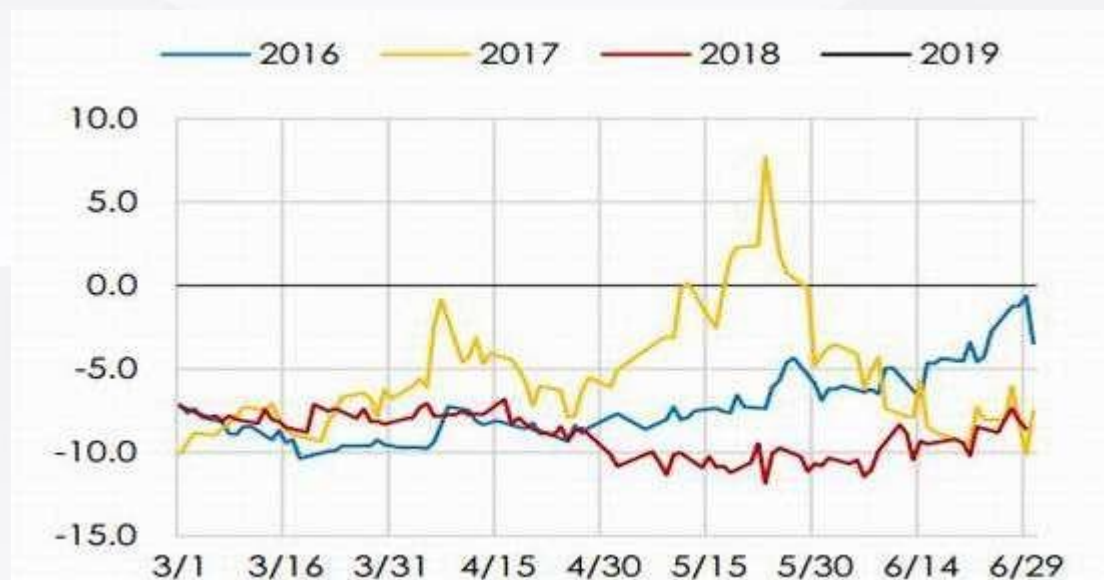
- 1.把这个图像做一些调整,
要求出现5个完整的波峰。
- 2.调大cos波形的幅度
- 3.调大sin波形的频率



14.1.5 线图



14.1.5 线图



作业6

将北京、广州、沈阳、上海和成都的雾霾数据，绘制出以天为单位的线图，数据取天的平均值，每个城市的数据用一条曲线表示。



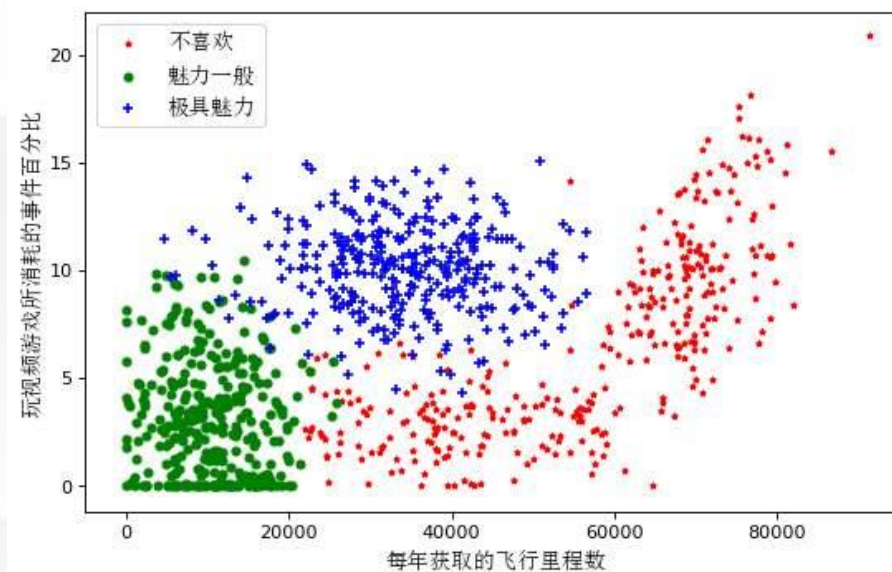
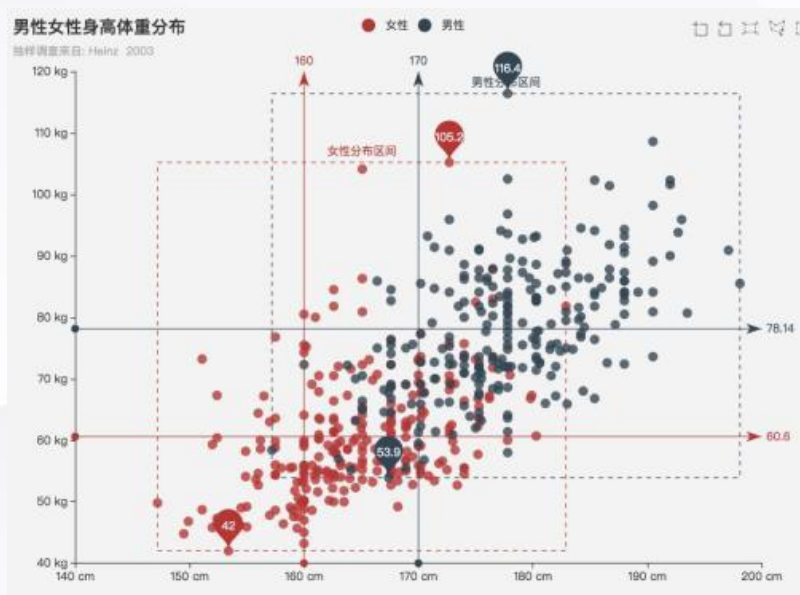
第14章 数据可视化

6. 散点图

14.1.6 散点图

散点图一般用在两组（多组）数据对比时，用这两组数据分别构成多个坐标点，考察坐标点的分布，便可以判断两组变量之间是否存在某种关联或者是属于不同的分类，或是根据坐标点的分布模式总结出其它规律。

散点图的值由点在图中的位置表示。类别由图中的不同标记表示。散点图通常用于比较跨类别的聚合数据。



使用scatter函数来绘制散点图，常用的参数如下：

```
plt.scatter(x, y, s= 20, c= None, marker= 'o', alpha= None, linewidths= None,  
edgecolors= None)
```

- x,y为散点的坐标位置，不可缺少；
- s：指定散点图点的大小，默认为20。通过传入新的变量，可以实现气泡图的绘制；
- c：指定散点图点的颜色，默认为蓝色；
- marker：指定散点图点的形状，默认为圆形；
- alpha：设置散点的透明度；
- linewidths：设置散点边界线的宽度；
- edgecolors：设置散点边界线的颜色。



14.1.6 散点图

示例：鸢尾花的分类问题。



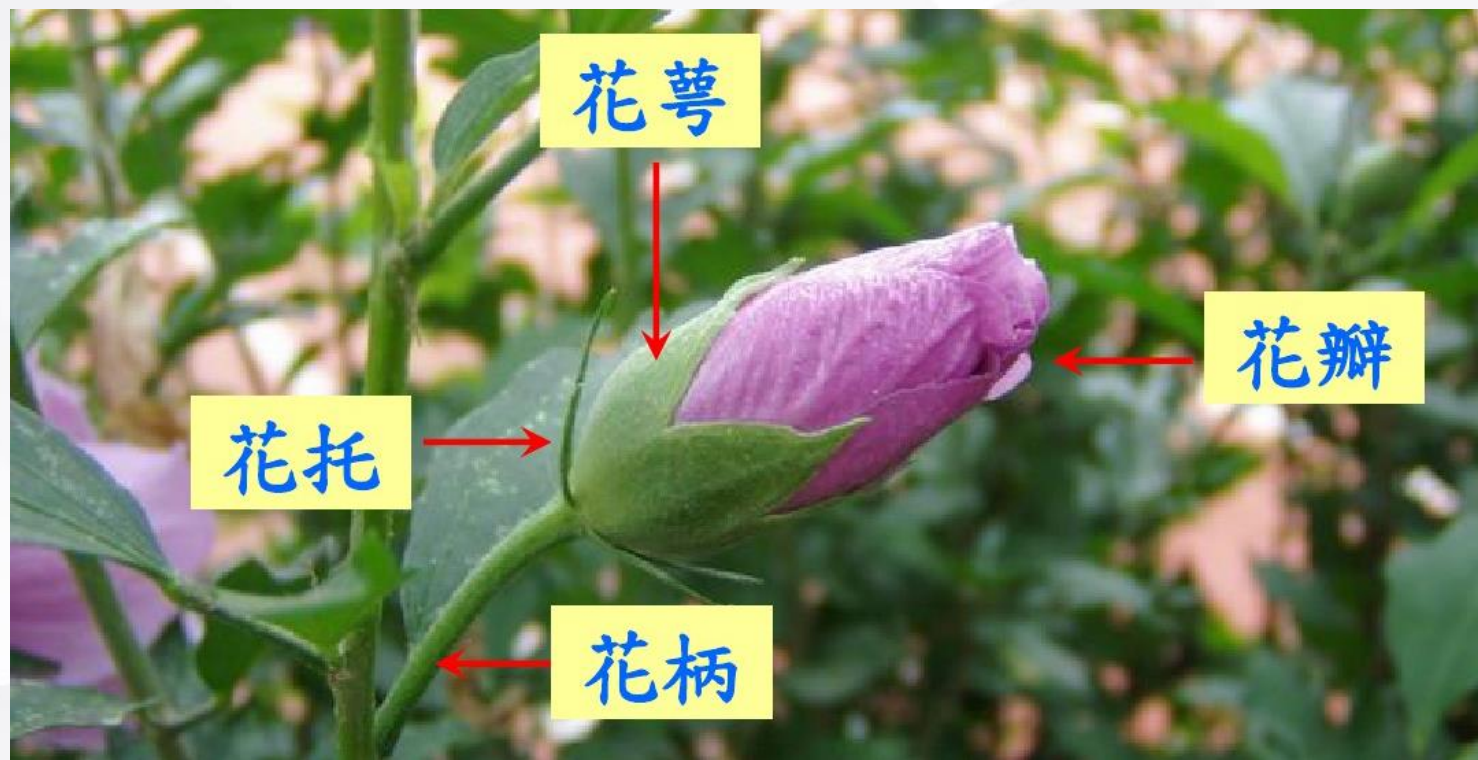
14.1.6 散点图

示例：鸢尾花的分类问题。



14.1.6 散点图

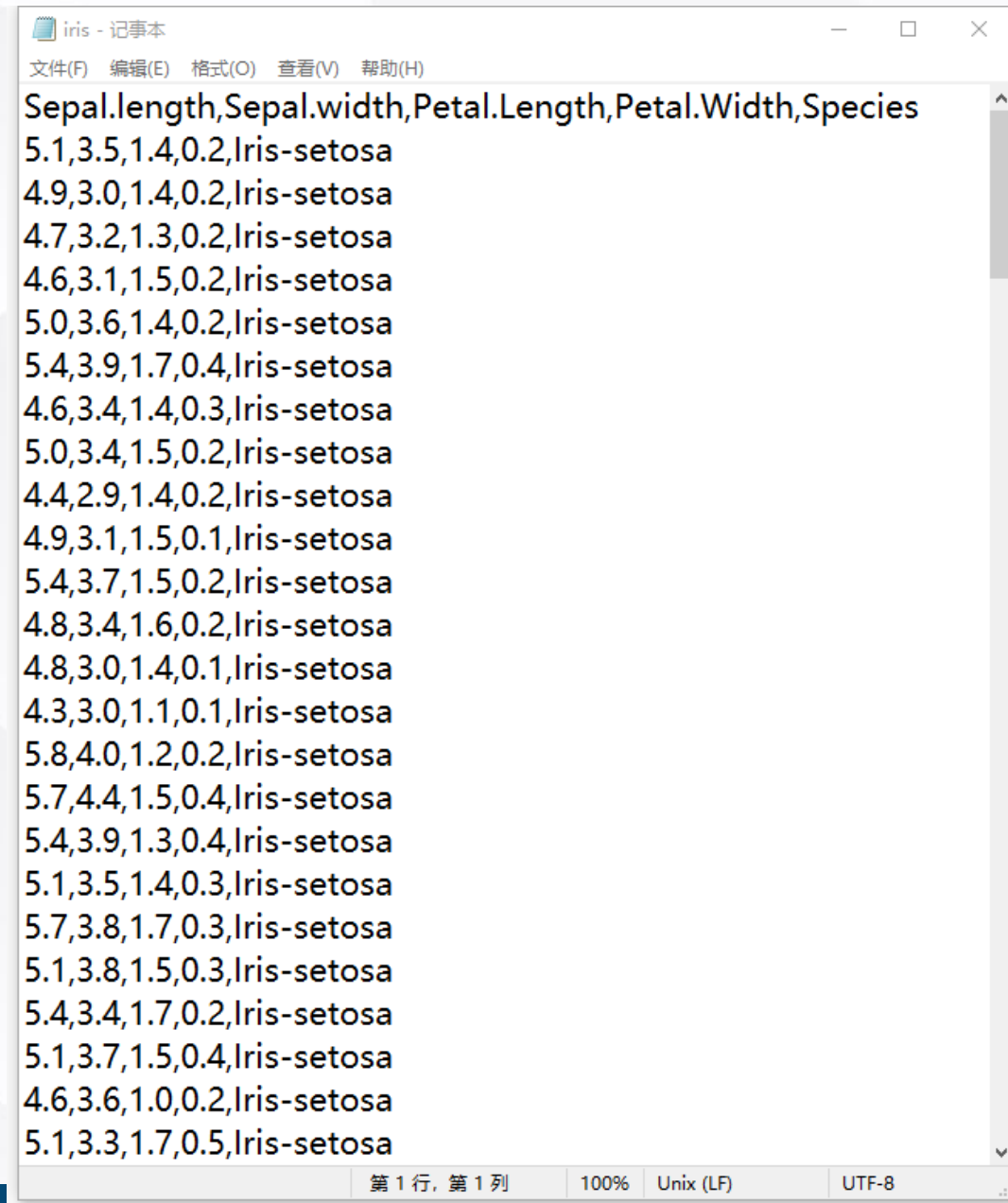
示例：鸢尾花的分类问题。



14.1.6 散点图

示例：鸢尾花的分类问题。

数据来自于IRIS数据集，该数据有150行，每一行有5个数据，分别代表花萼长度、花萼宽度、花瓣长度、花瓣宽度和种类。种类共有三种，分别是Iris-Setosa（山鸢尾）、Iris-Versicolour（杂色鸢尾），以及Iris-Virginica（维吉尼亚鸢尾）。



```
iris - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
Sepal.length,Sepal.width,Petal.Length,Petal.Width,Species
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
4.6,3.1,1.5,0.2,Iris-setosa
5.0,3.6,1.4,0.2,Iris-setosa
5.4,3.9,1.7,0.4,Iris-setosa
4.6,3.4,1.4,0.3,Iris-setosa
5.0,3.4,1.5,0.2,Iris-setosa
4.4,2.9,1.4,0.2,Iris-setosa
4.9,3.1,1.5,0.1,Iris-setosa
5.4,3.7,1.5,0.2,Iris-setosa
4.8,3.4,1.6,0.2,Iris-setosa
4.8,3.0,1.4,0.1,Iris-setosa
4.3,3.0,1.1,0.1,Iris-setosa
5.8,4.0,1.2,0.2,Iris-setosa
5.7,4.4,1.5,0.4,Iris-setosa
5.4,3.9,1.3,0.4,Iris-setosa
5.1,3.5,1.4,0.3,Iris-setosa
5.7,3.8,1.7,0.3,Iris-setosa
5.1,3.8,1.5,0.3,Iris-setosa
5.4,3.4,1.7,0.2,Iris-setosa
5.1,3.7,1.5,0.4,Iris-setosa
4.6,3.6,1.0,0.2,Iris-setosa
5.1,3.3,1.7,0.5,Iris-setosa
```

第 1 行, 第 1 列 100% Unix (LF) UTF-8

1.从文件读入数据

```
#demo1406:
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
fig,ax=plt.subplots()
# 读取数据
BasePath='c:\\code' #csv文件的保存路径
iris = pd.read_csv( BasePath+'\\iris.csv')
colors = [ 'r', 'y', 'b'] # 定义三种散点的颜色
Species = iris.Species.unique() #对类别去重
```



2. 绘制散点图，子图1

```
plt.subplot(1,2,1) #设置子图有2个，即1行2列，先画左边的第一个
for i in range(len(Species)):
    plt.scatter(iris.loc[iris.Species == Species[i], 'Petal.Length'], iris.loc[iris.Species
== Species[i], 'Petal.Width'], s = 35, c = colors[i], label = Species[i])

# 添加轴标签和标题
plt.title( 'Length vs Width')
plt.xlabel( 'Petal.Length')
plt.ylabel( 'Petal.Width')
plt.grid(True, linestyle='--', alpha=0.8) #设置网格线
plt.legend(loc = 'lower right') # 添加图例
```

```
plt.scatter(
iris.loc[iris.Species == "Iris-Setosa", 'Petal.Length'],
iris.loc[iris.Species == "Iris-Setosa", 'Petal.Width'],
s = 35, c = colors[0], label = Species[0])
```

2.绘制散点图，子图2

```
plt.subplot(1,2,2) #设置子图有2个，即1行2列，再画右边的第二个
for i in range(len(Species)): #x和y轴交换一下位置
    plt.scatter(iris.loc[iris.Species == Species[i], 'Petal.Width'], iris.loc[iris.Species
== Species[i], 'Petal.Length'], s = 35, c = colors[i], label = Species[i])
```

添加轴标签和标题

```
plt.title( 'Width vs Length')
```

```
plt.xlabel( 'Petal.Width')
```

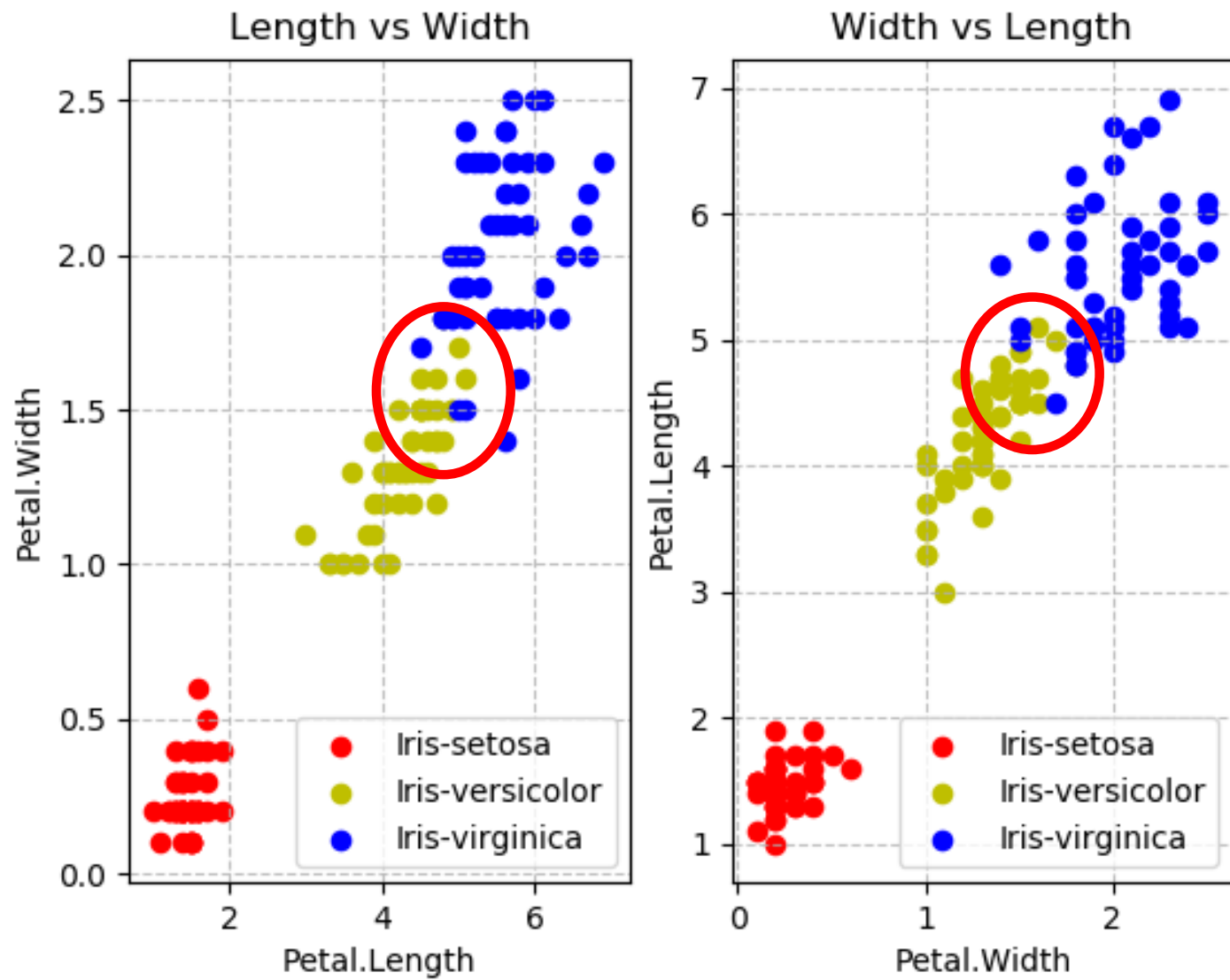
```
plt.ylabel( 'Petal.Length')
```

```
plt.grid(True, linestyle='--', alpha=0.8) #设置网格线
```

```
plt.legend(loc = 'lower right') # 添加图例
```

```
plt.show()
```

14.1.6 散点图

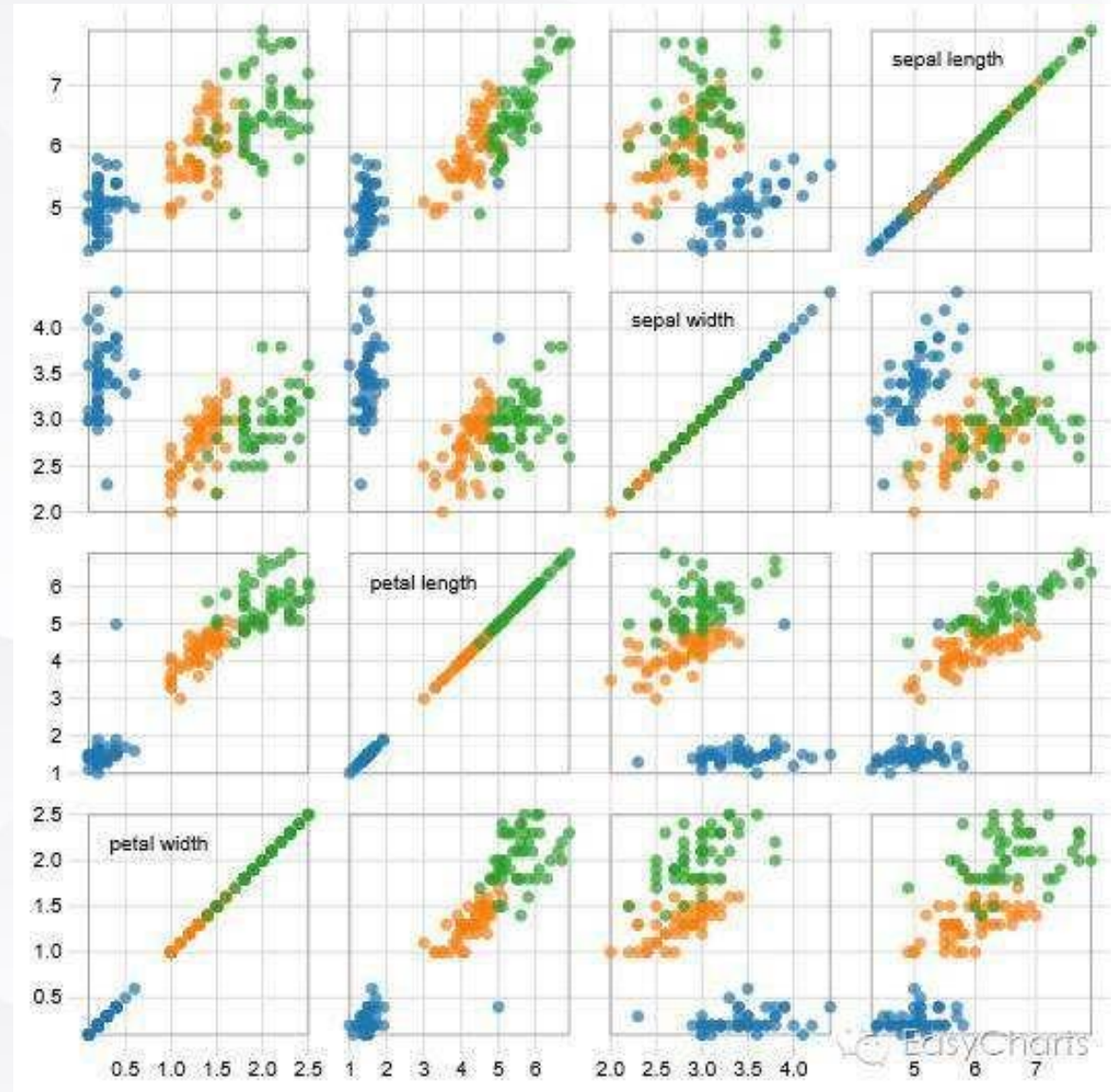


14.1.6 散点图

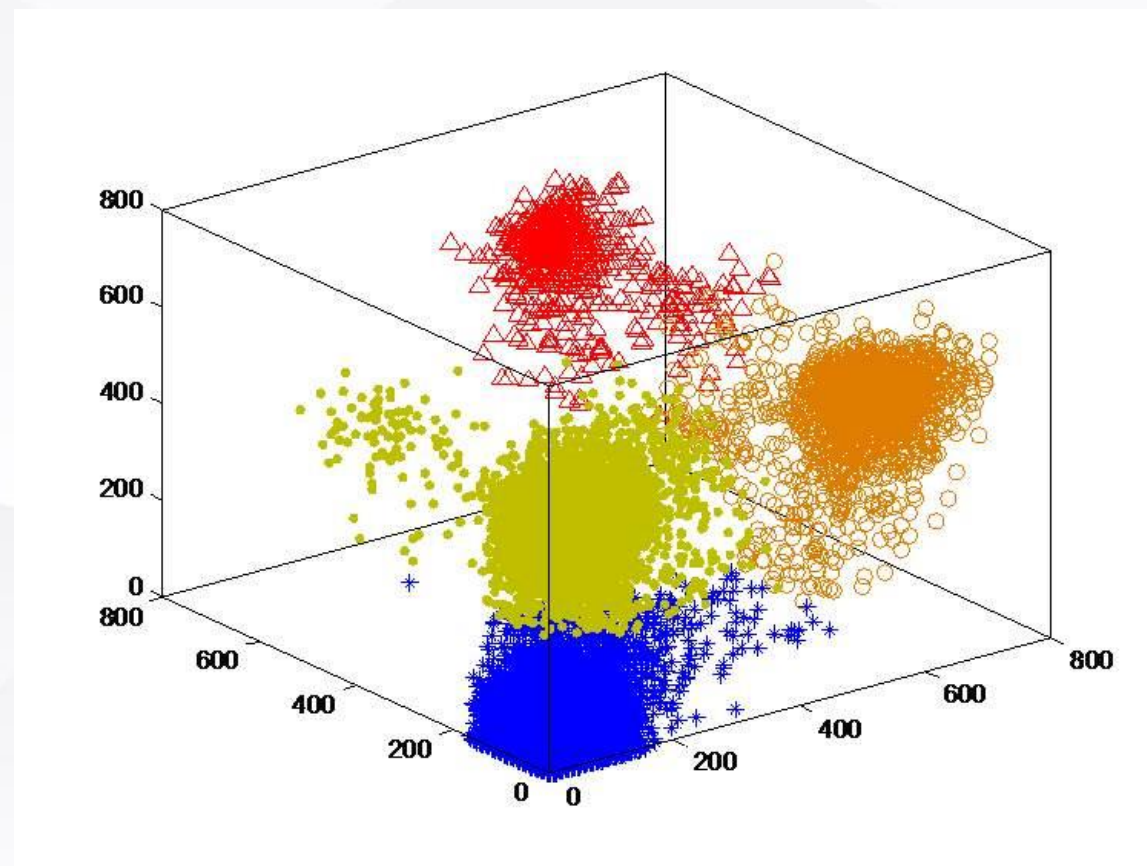
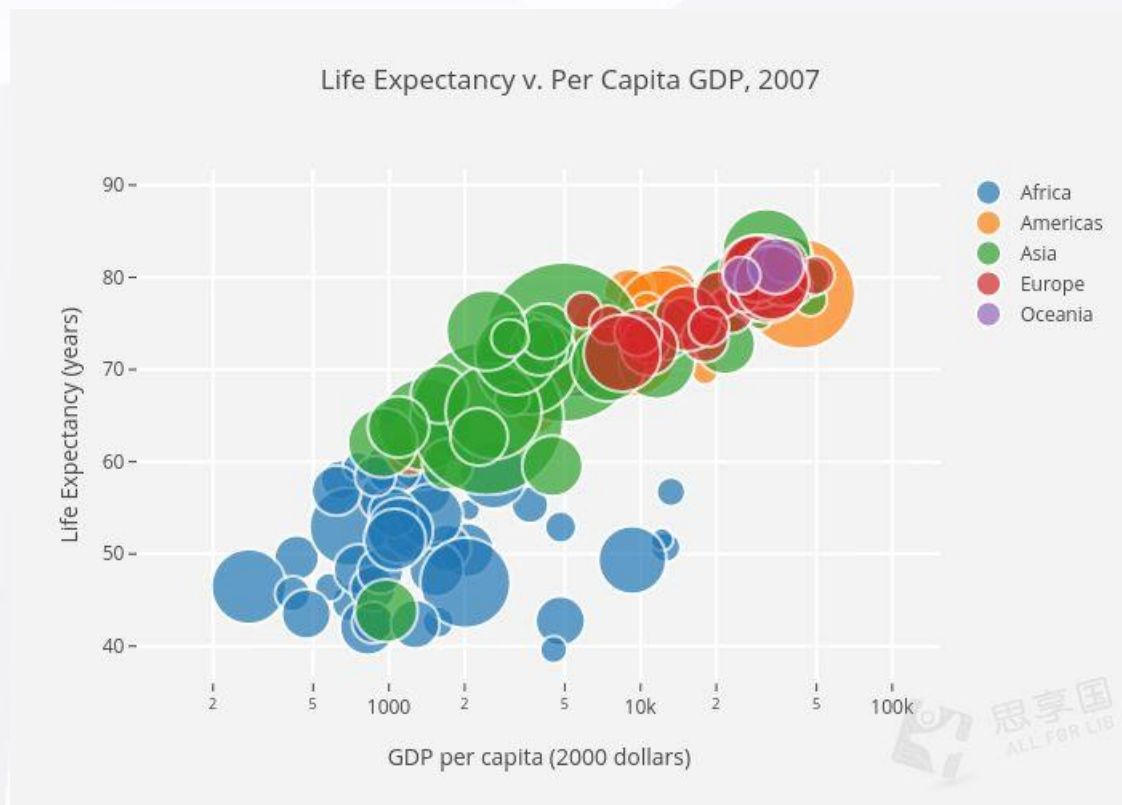
作业7:

使用IRIS数据集，在一个figure中绘制出右侧的16个子图。

分别使用花瓣长度、花瓣宽度、花萼长度和花萼宽度这四种数据，两两组合，形成散点。



14.1.6 散点图



第14章 数据可视化

7. 饼图

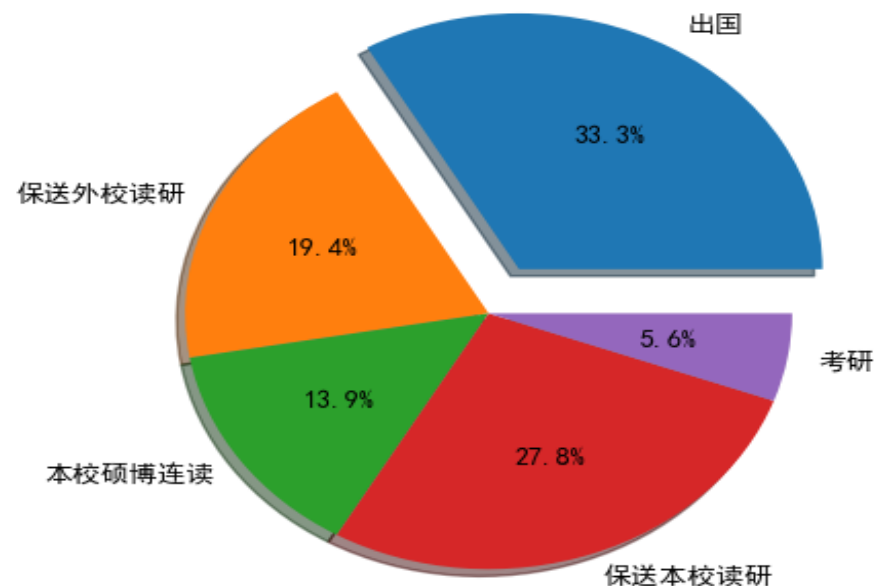
14.1.7 饼图

饼图一般用来表示各个组成部分的比例或者是构成关系。

我们使用pie函数来绘制饼图，常用的参数如下：

```
patches, texts, autotexts = plt.pie(size, explode=explode_list, labels=label_list,
labeldistance=1.1, autopct="%1.1f%%", shadow=False, startangle=90, pctdistance=0.6)
```

- size: 饼图中各部分的数值
- explode: 设置各部分突出显示的比例
- label: 设置各部分的标签
- labeldistance: 设置标签文本距圆心位置, 1.1表示1.1倍半径
- autopct: 设置饼图里面的文本格式
- shadow: 设置是否有阴影
- startangle: 起始角度, 默认从0开始逆时针转
- pctdistance: 设置圆内文本距圆心距离



14.1.7 饼图

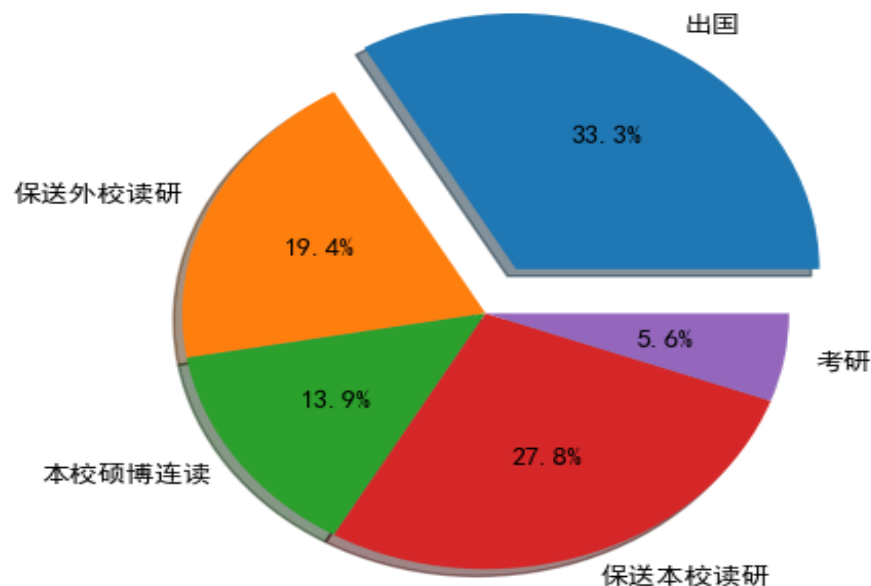
饼图一般用来表示各个组成部分的比例或者是构成关系。

我们使用pie函数来绘制饼图，常用的参数如下：

```
patches, texts, autotexts = plt.pie(size, explode=explode_list, labels=label_list,
labeldistance=1.1, autopct="%1.1f%%", shadow=False, startangle=90, pctdistance=0.6)
```

返回值

- patches: 返回一个列表，其中的元素为饼图的各个扇区
 - texts: 返回一个包含标签实例的列表，标签是指在饼图外部的文本
 - autotexts: 返回一个包含数据标签实例的列表，数据标签是指在饼图内部的文本
- 可以对返回值进行进一步的设置，实现更多的显示效果。

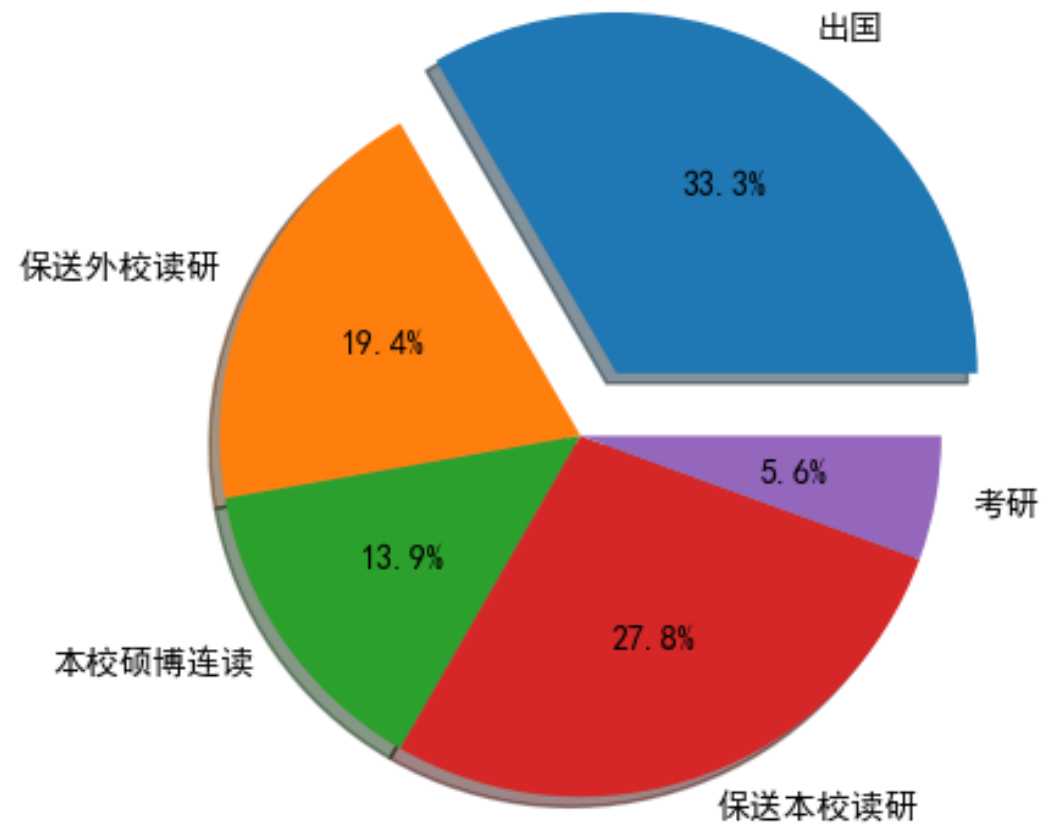


14.1.7 饼图

```
#demo1407:
import matplotlib.pyplot as plt
import matplotlib
from matplotlib import font_manager as fm

matplotlib.rcParams['font.sans-serif'] = ['SimHei'] #支持中文的细黑体
label_list = ["出国", "保送外校读研", "本校硕博连读", "保送本校读研", "考研"]
# 各部分标签
size = [12,7,5,10,2] # 各部分的人数
explode = [0.2, 0, 0,0,0] # 各部分的突出显示比例
patches, texts, autotexts= plt.pie(size, explode=explode, labels=label_list,
labeldistance=1.1, autopct="%1.1f%%", shadow=True, startangle=0,
pctdistance=0.6)
plt.show()
```

14.1.7 饼图



#调整字体大小

```
proptease = fm.FontProperties()
```

```
proptease.set_size('xx-large')
```

font size include: 'xx-small', 'x-small', 'small', 'medium', 'large', 'x-large', 'xx-large'
or number, e.g. '12'

```
plt.setp(texts, fontproperties=proptease)
```

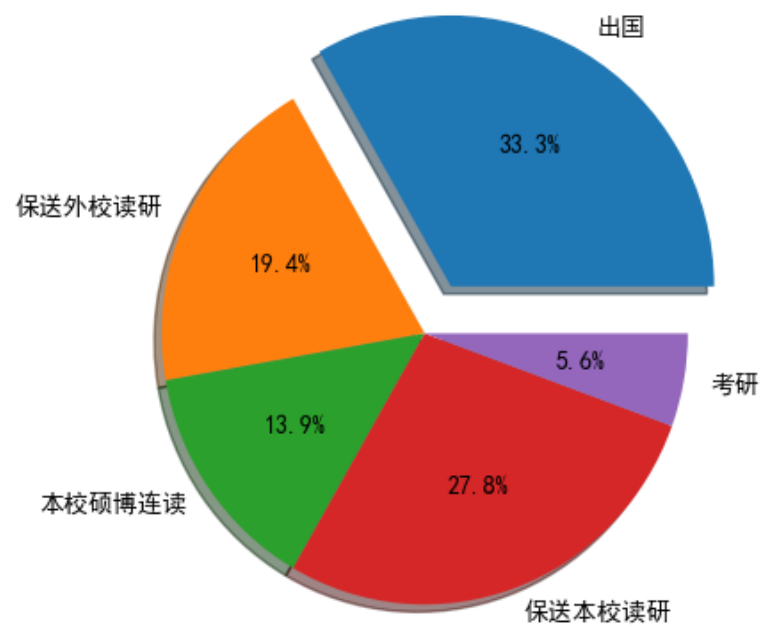
```
plt.setp(autotexts, fontproperties=proptease)
```

```
plt.show()
```

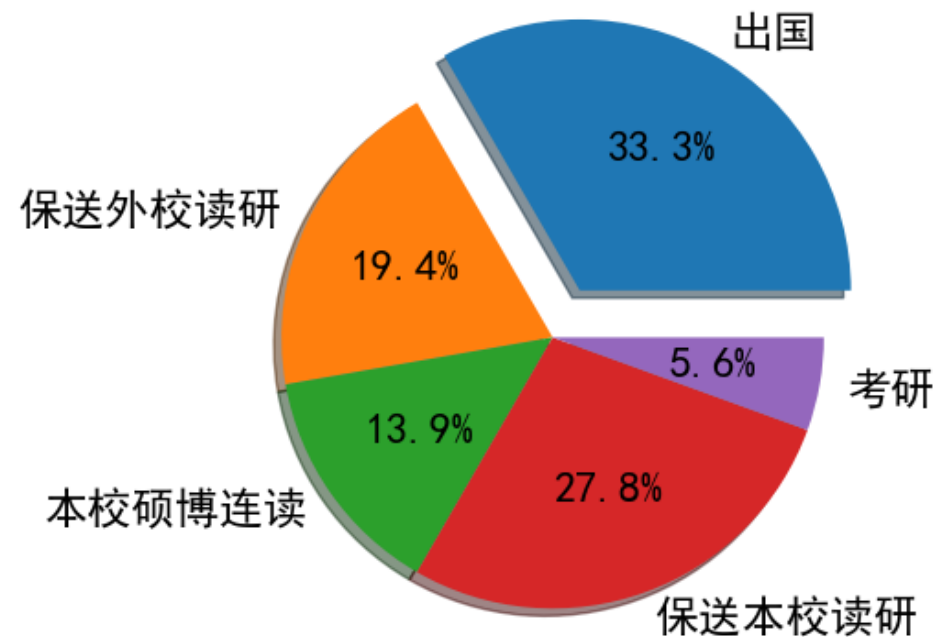


14.1.7 饼图

调整字体大小前



调整字体大小后

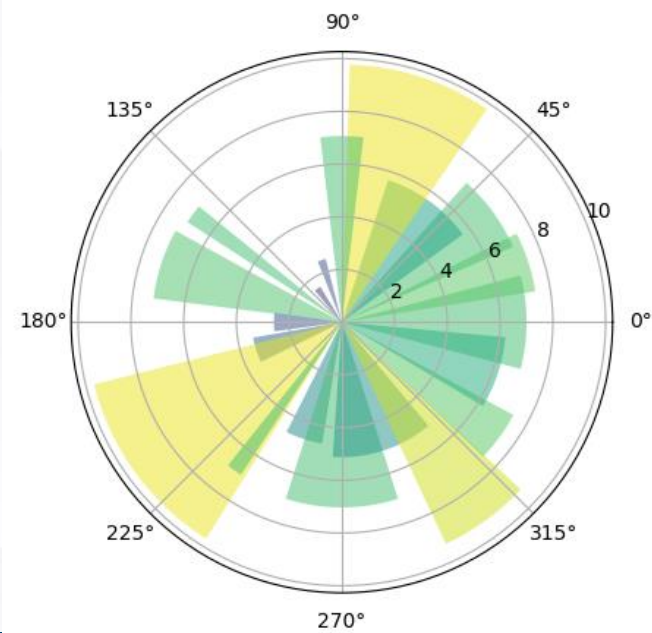
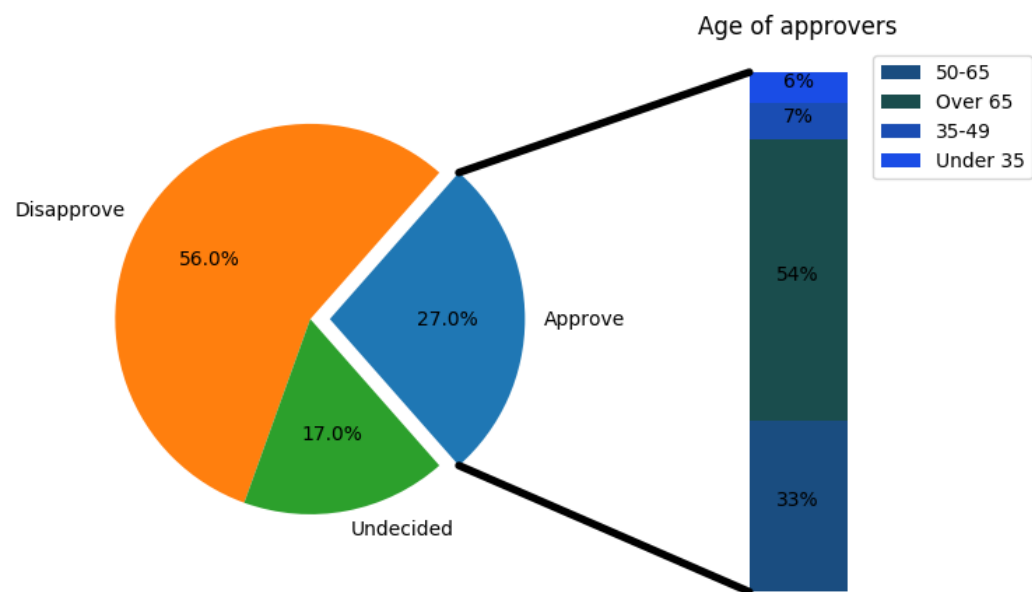
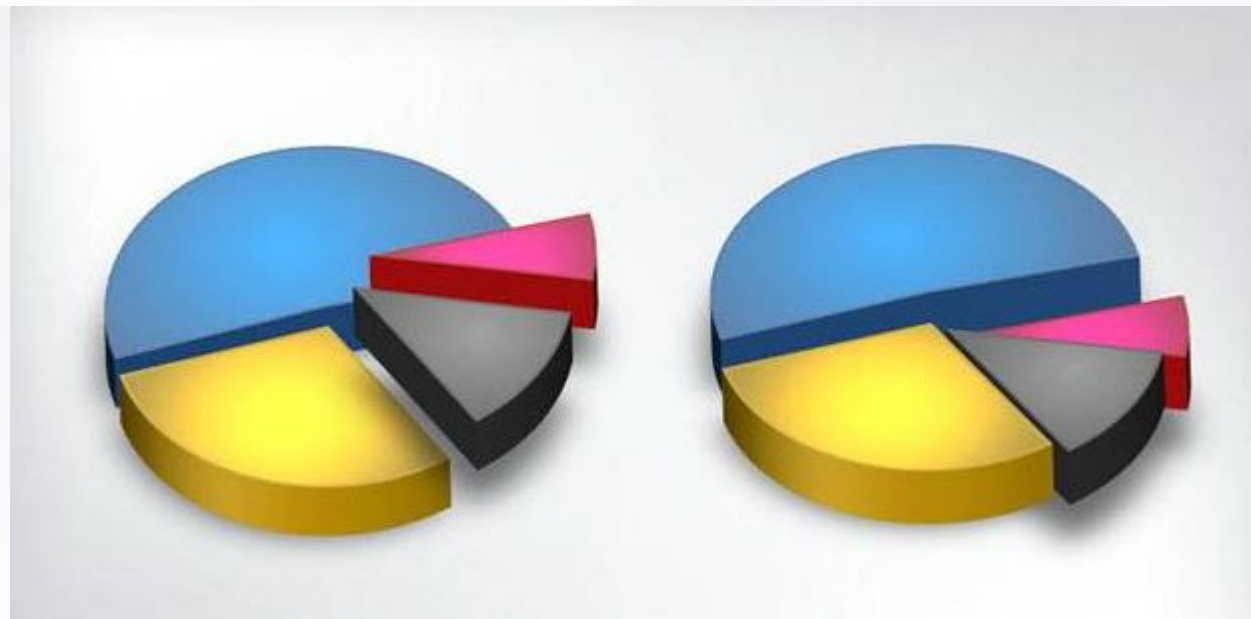


作业8

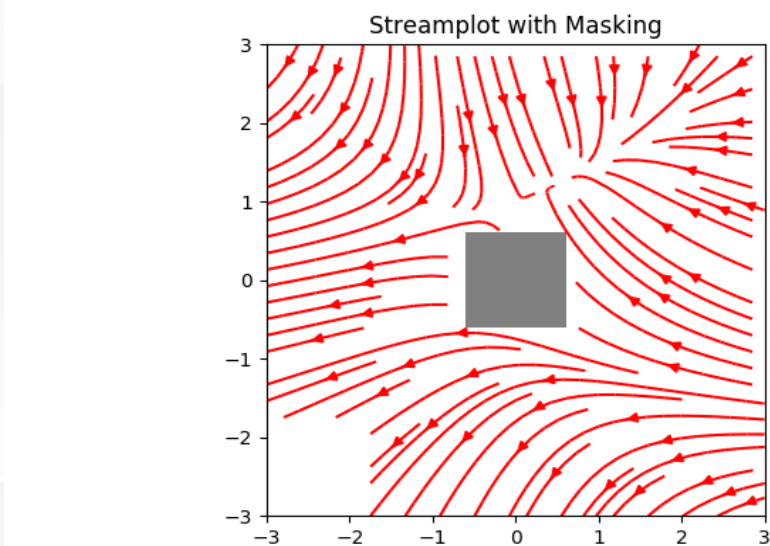
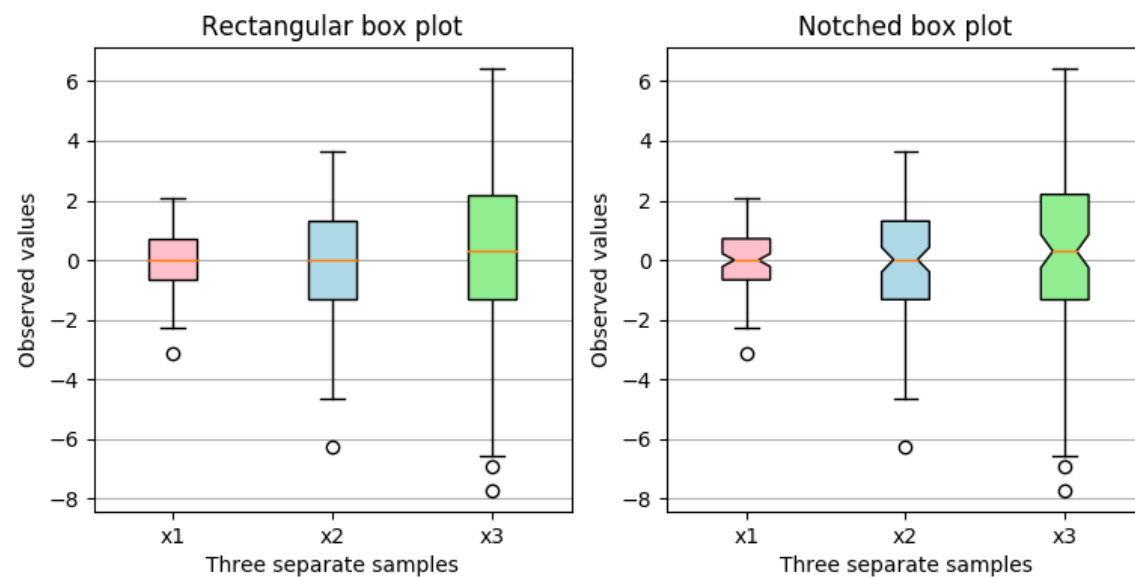
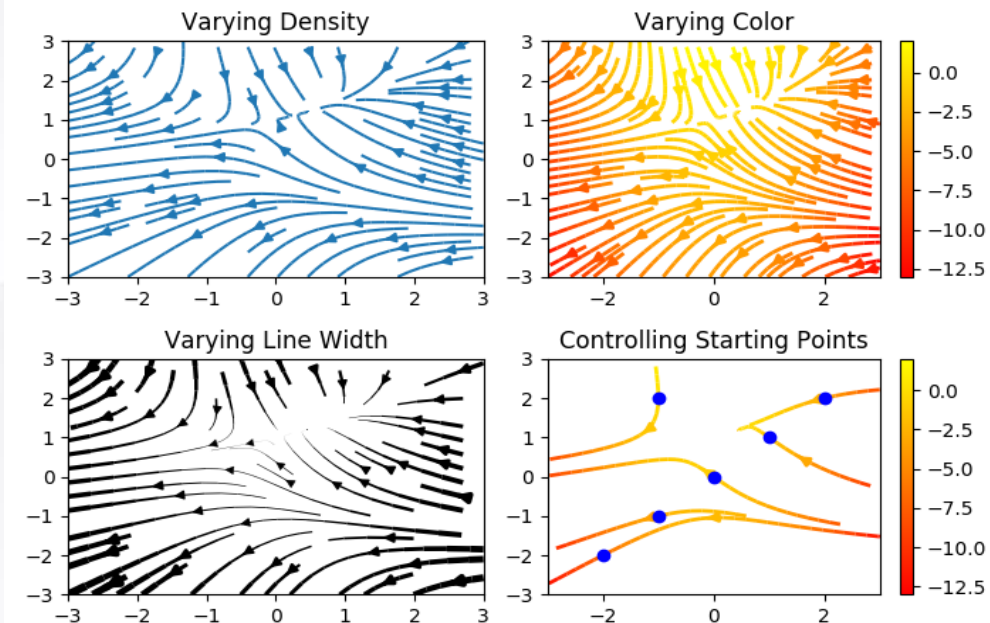
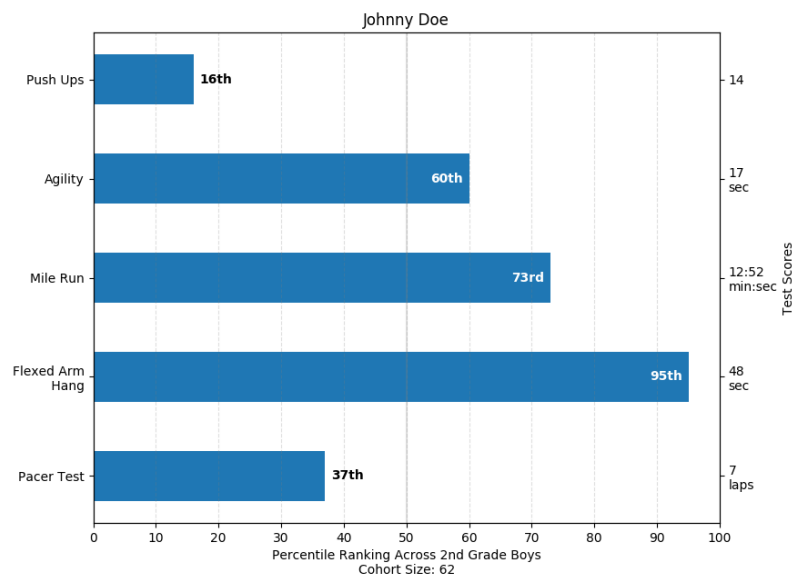
1. 增加数据量，看看数据的项数在什么范围比较合适在饼图中展示；
2. 调整数据的顺序或角度，使得“保送研究生”的扇区在12点方向开始；
3. 调整字体的大小、标签的位置等参数。



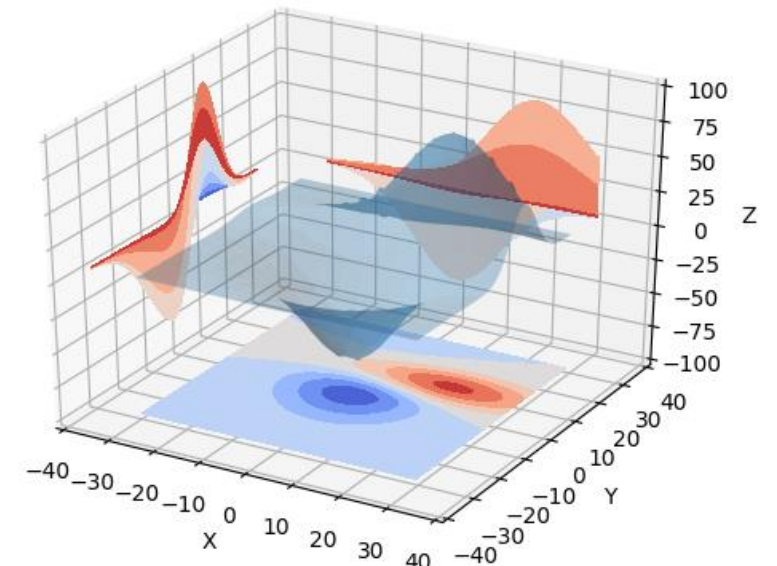
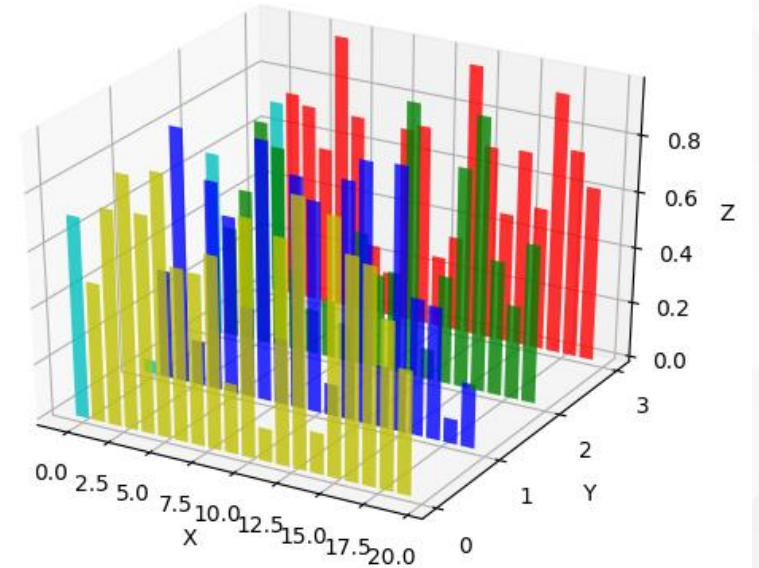
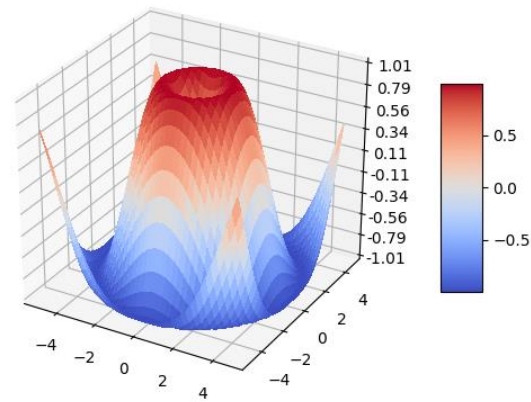
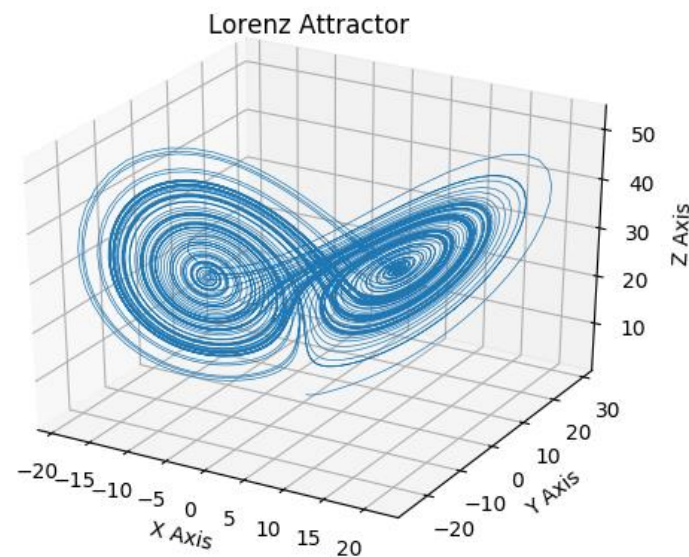
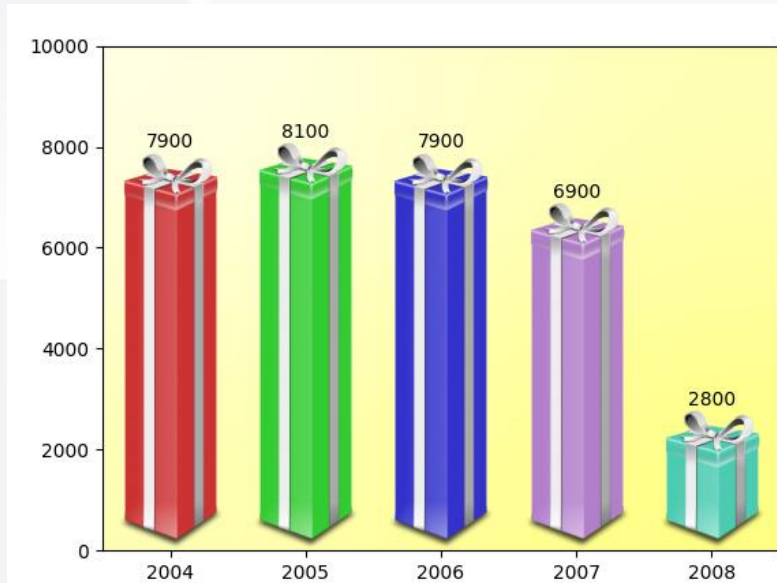
14.1.7 饼图



14.1.8 其它图形



14.1.8 其它图形



用户量排行

排名	区域	用户量
1	鼓楼、晋安	1,565
2	台江、仓山	995
3	福清	361
4	闽侯	251

用户量排行

排名	区域	用户量
1	鼓楼、晋安	1,565
2	台江、仓山	995
3	福清	361
4	闽侯	251

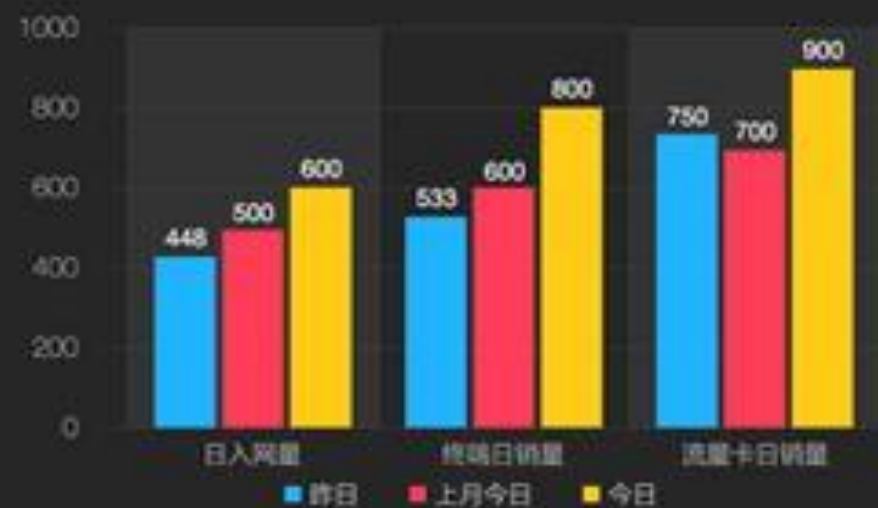
用户量排行

排名	区域	用户量
1	鼓楼、晋安	1,565
2	台江、仓山	995
3	福清	361
4	闽侯	251

客流量



变动情况



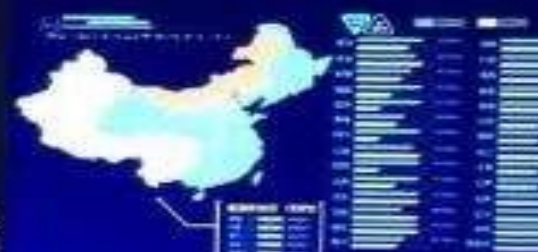
指标完成度

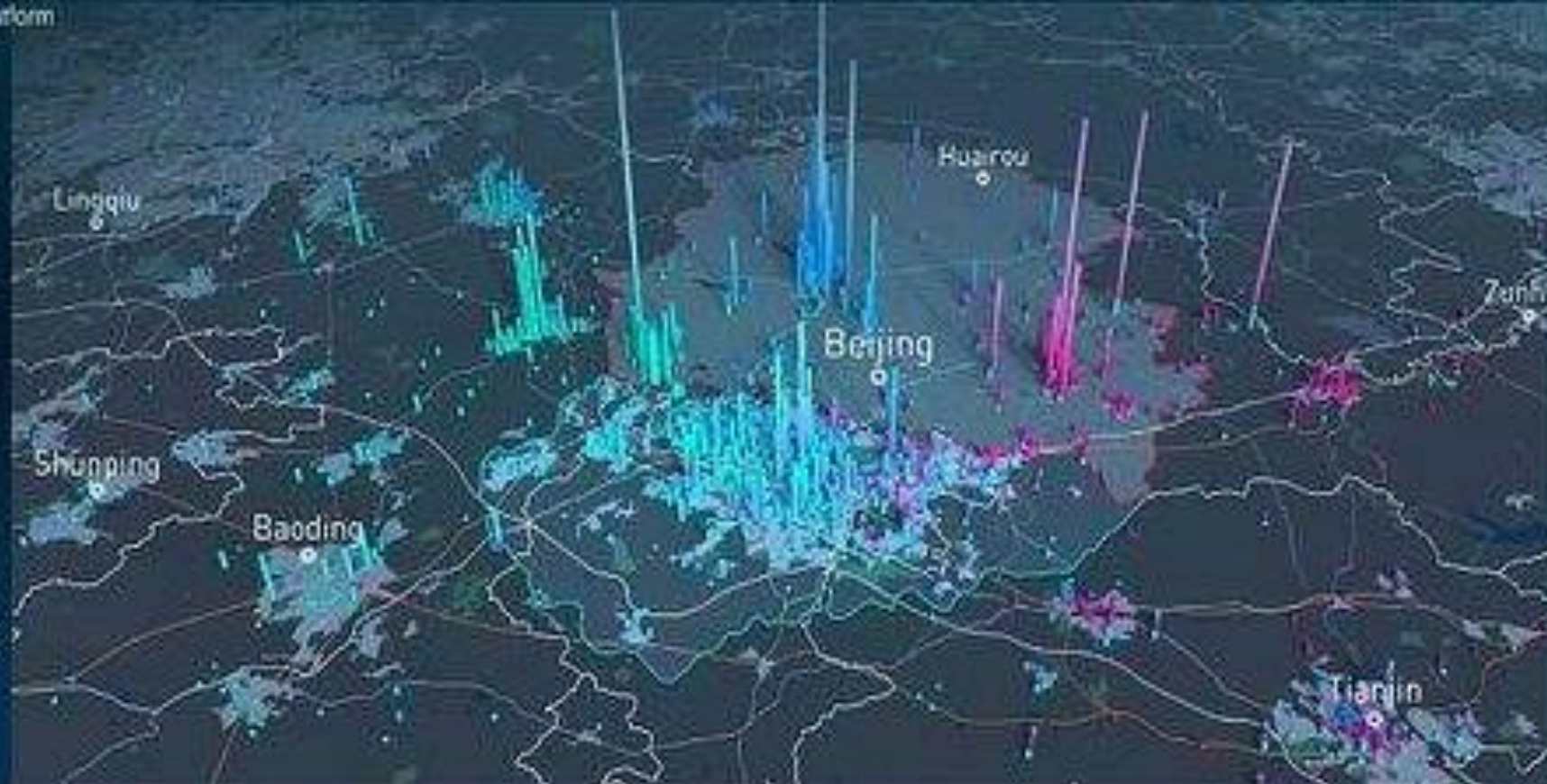


销售情况



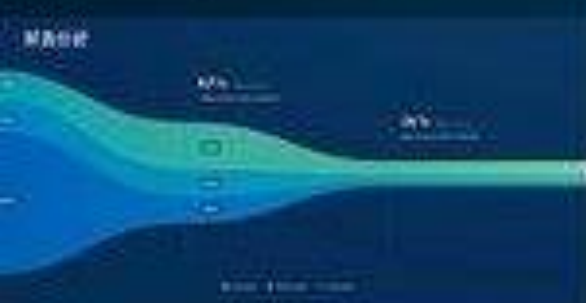
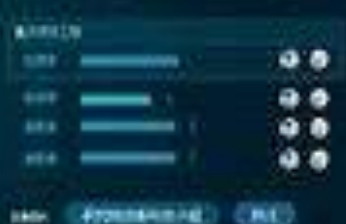
总销量	541
合约机销量	342
裸机销量	199
寄售机销量	100
4G手机销量	321





实时监控数据

实时监控数据

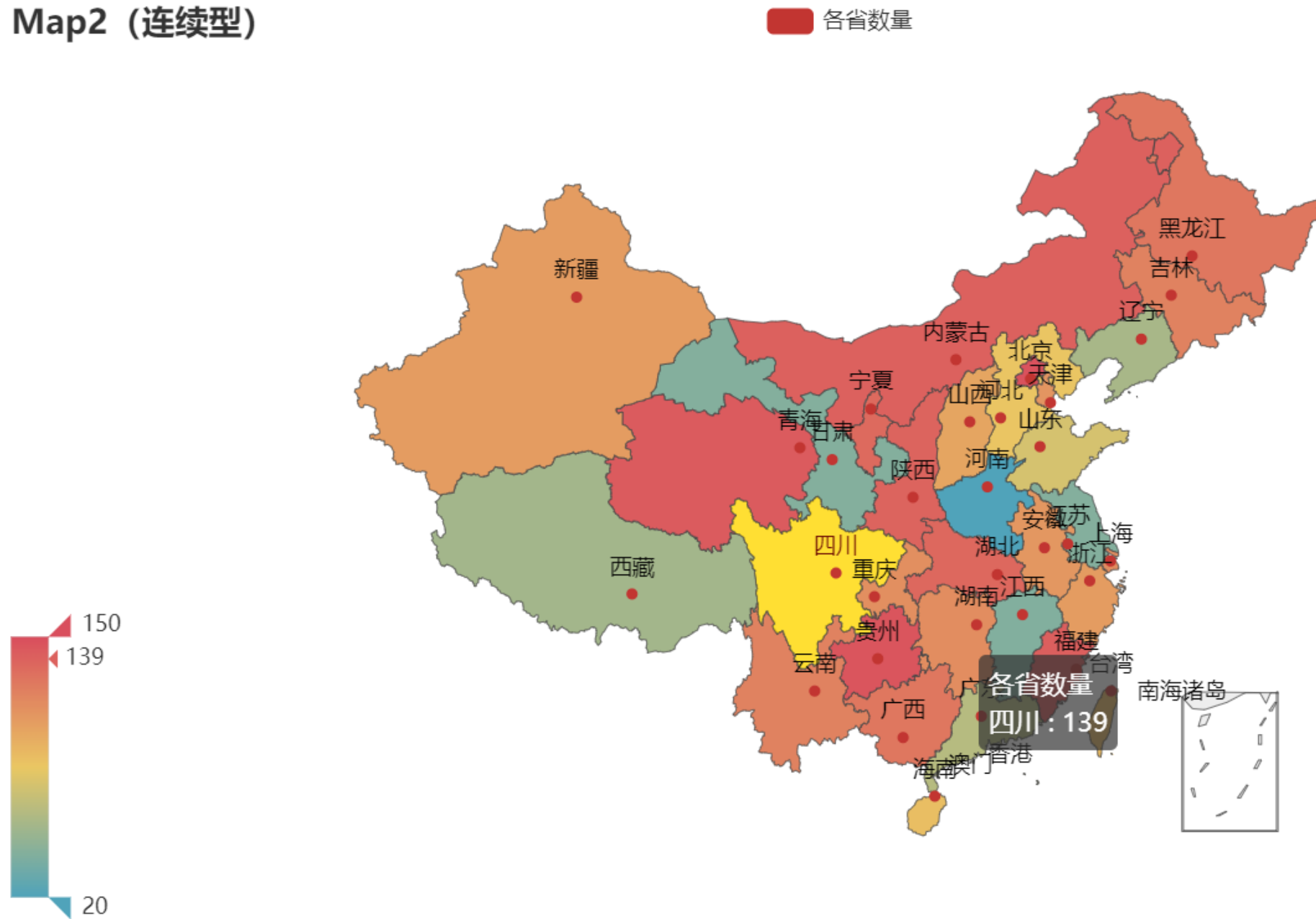


第14章 数据可视化

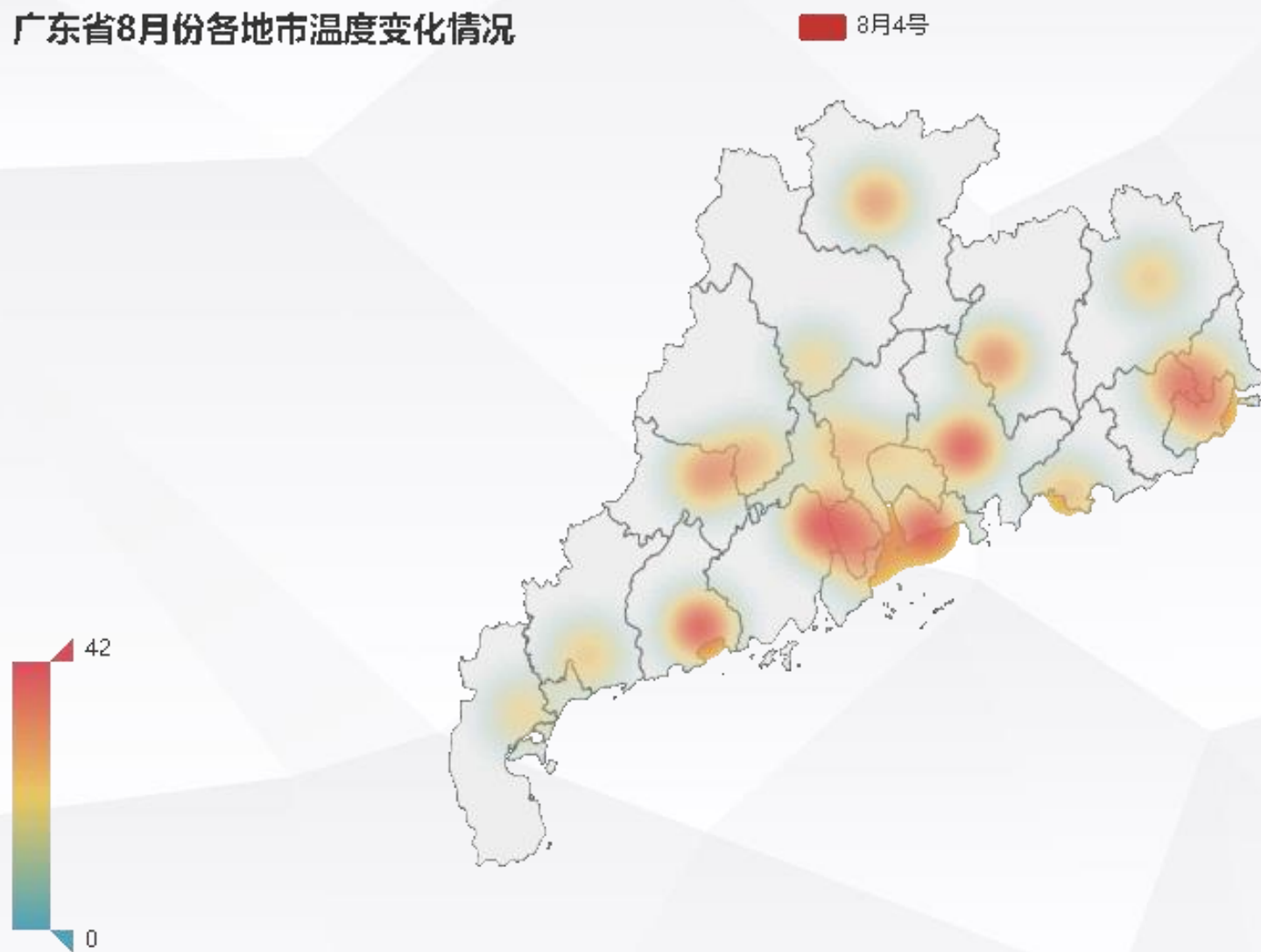
8. 地图

14.1.9 其它可视化工具

Map2 (连续型)



广东省8月份各地市温度变化情况



Echarts 是一个由百度开源的数据可视化工具，pyecharts 是支持Python语言的第三方库。

<https://pyecharts.org/#/zh-cn/intro>

- 简洁的 API 设计，使用如丝滑般流畅，支持链式调用
- 囊括了 30+ 种常见图表，应有尽有
- 可轻松集成至 Flask, Django 等主流 Web 框架
- 高度灵活的配置项，可轻松搭配出精美的图表
- 多达 400+ 地图文件以及原生的百度地图，为地理数据可视化提供强有力的支持
- 详细的文档和示例，帮助开发者更快的上手项目



```
from pyecharts import options as opts
from pyecharts.charts import Map
import random
```

```
class Data:
```

```
    provinces = ["湖北", "广东", "北京", "上海", "江西", "河南", "浙江", "江苏",
                  "湖南", "广西", "山东", "陕西", "山西", "河北", "福建", "黑龙江",
                  "新疆", "西藏", "云南", "贵州", "四川", "台湾", "宁夏", "吉林",
                  "青海", "甘肃", "内蒙古", "重庆", "安徽", "天津", "海南", "辽宁"]
```

```
@staticmethod
```

```
def values(start: int = 20, end: int = 150) -> list:
    return [random.randint(start, end) for _ in range(32)]
```

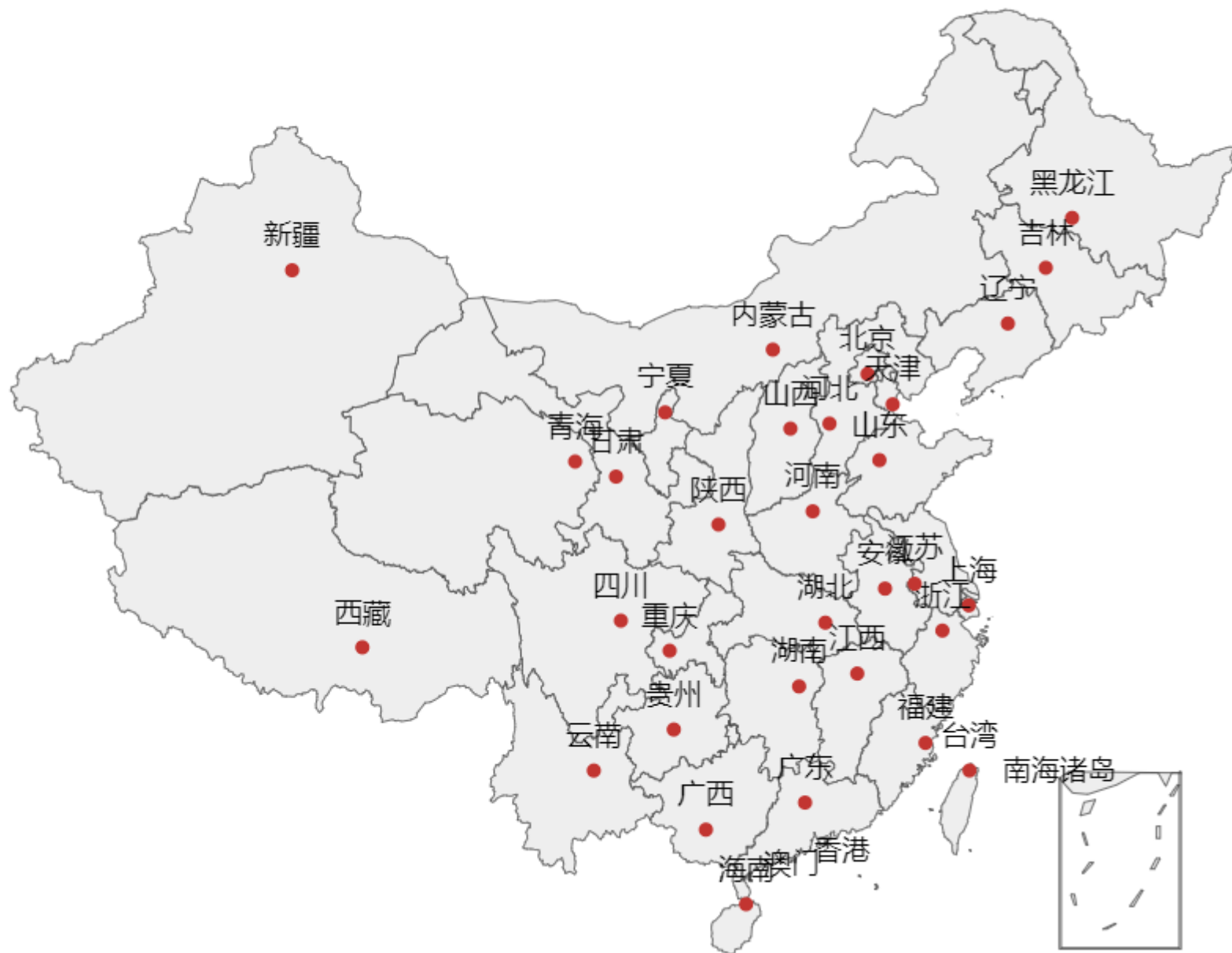


```
def map1() -> Map:  
    c = (  
        Map()  
        .add("各省数量", [list(z) for z in zip(Data.provinces, Data.values())], "china")  
        .set_global_opts(title_opts=opts.TitleOpts(title="Map1-基本示例"))  
    )  
    return c  
  
map1().render("map1.html")
```



Map1-基本示例

各省数量



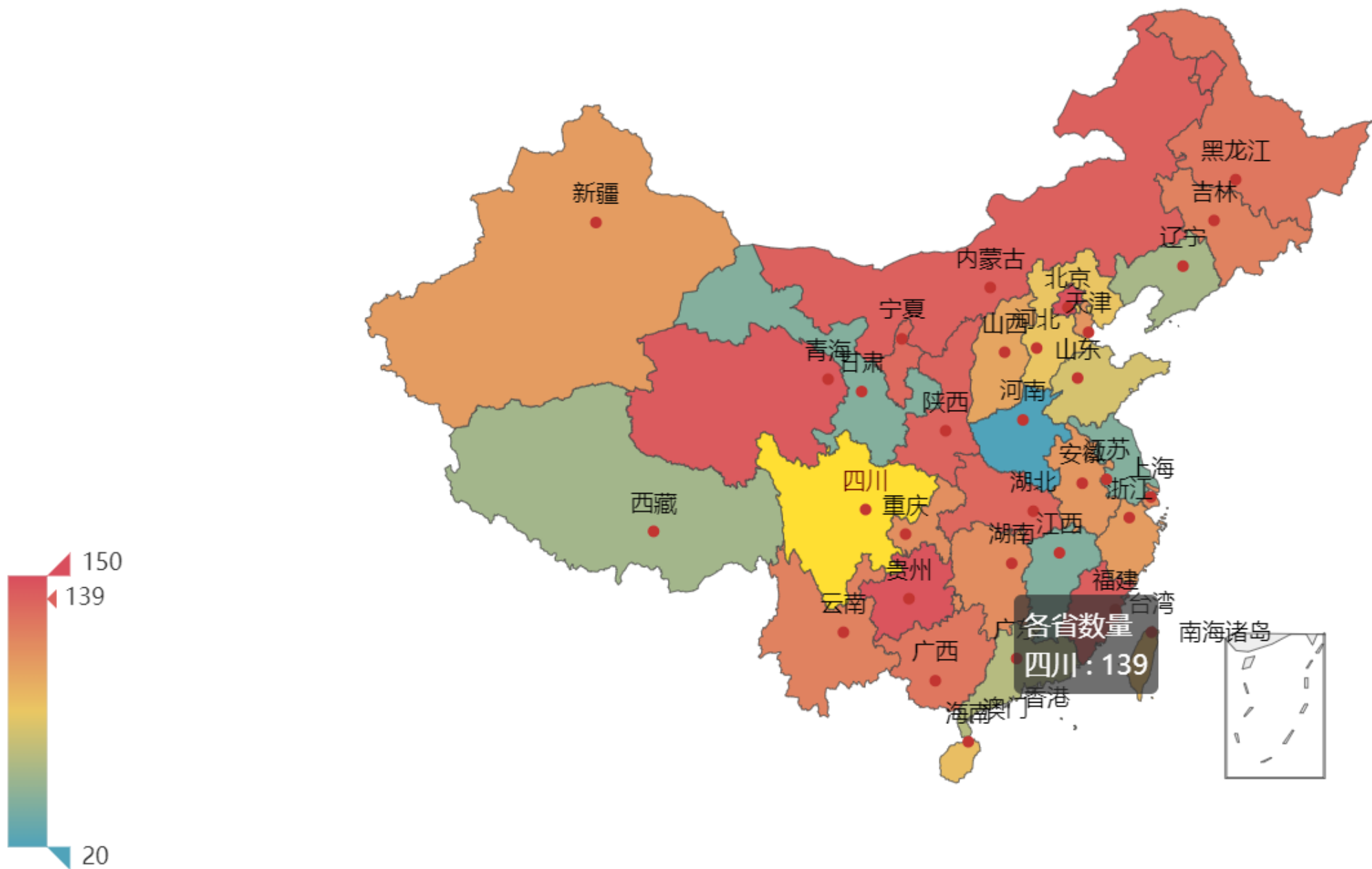
```
def map2() -> Map:
    c = (
        Map()
        .add("各省大学数量", [list(z) for z in zip(Data.provinces, Data.values())], "china")
        .set_global_opts(
            title_opts=opts.TitleOpts(title="Map2 (连续型) "),
            visualmap_opts=opts.VisualMapOpts(min_=20,max_=150))
        .set_series_opts(label_opts=opts.LabelOpts(is_show=True))
    )
    return c

map2().render("map2.html")
```



Map2 (连续型)

各省数量



14.1.9 其它可视化工具

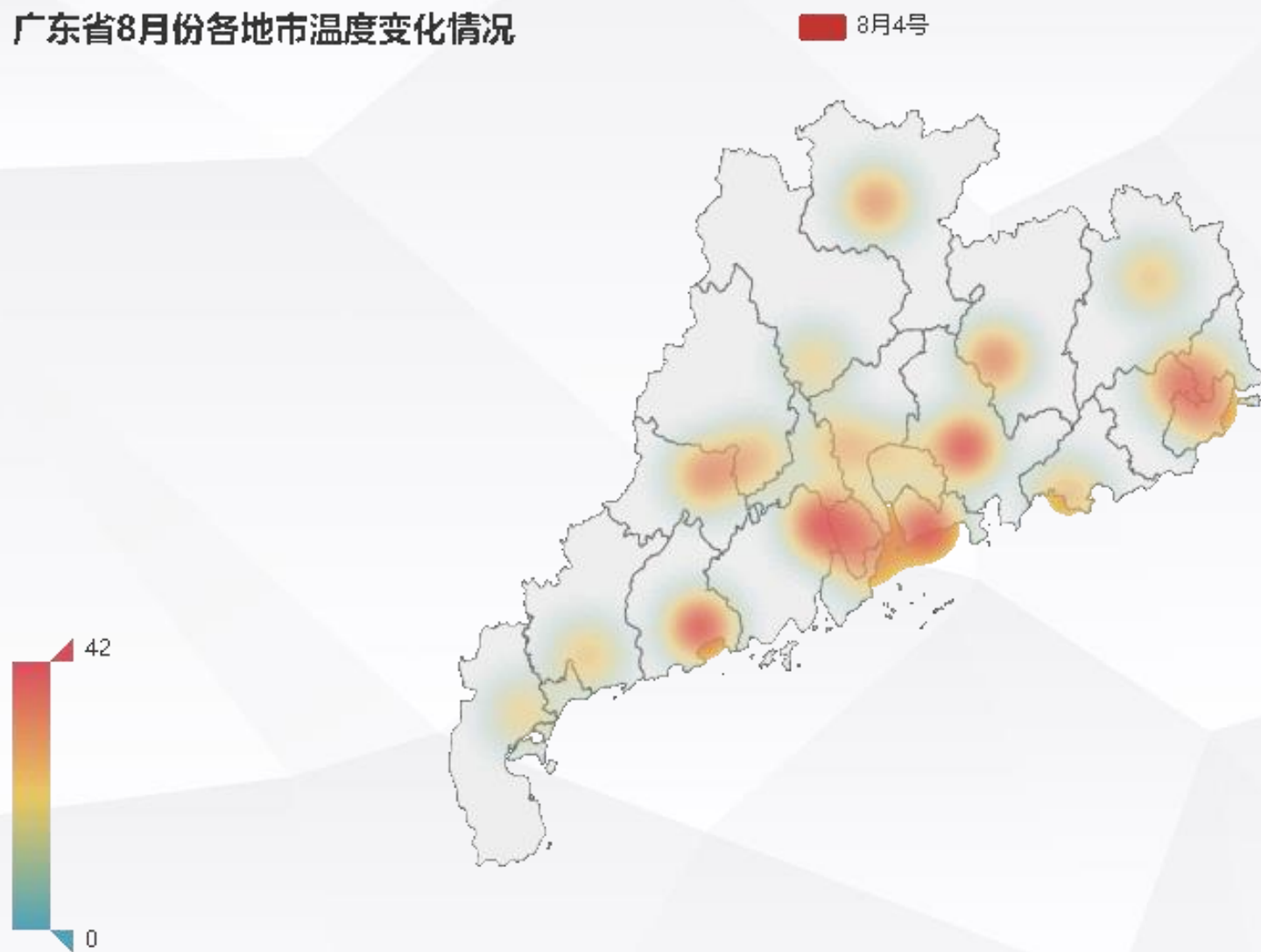
作业9

在中国地图上展示每个省
的高考人数或大学数量

A	B	C	D	E
	高考人数		大学数量	
省/市	2017年	2016年	211&985大学数量	公办本科大学数量
河南	86.3万	82万	1	37
广东	75.7万	73.3万	4	41
山东	58.3万	60.2万	3	44
四川	58.3万	57.13万	5	34
安徽	49.9万	50.99万	3	30
河北	43.6万	42.31万	0	36
贵州	41.2万	37.38万	1	19
湖南	41.1万	40.16万	3	31
广西	36.5万	33万	1	24
江西	36.5万	36.06万	1	23
湖北	36.2万	36.14万	1	36
江苏	33万	36.04万	11	45
陕西	31.9万	32.8万	1	34
山西	31.7万	33.9万	1	22
云南	29.3万	28万	1	22
浙江	29.1万	30.74万	1	32
甘肃	28.5万	29.6万	1	17
重庆	21.1万	24.88万	2	17
辽宁	20.8万	21.82万	4	4
内蒙古	19.8万	20.11万	1	14
福建	18.8万	17.5万	2	22
黑龙江	18.8万	19.7万	4	26
新疆	18.4万	16.61万	2	13
吉林	14.3万	14.85万	3	25
宁夏	6.9万	6.9万	1	4
北京	6万	6.12万	26	60
海南	5.7万	6.04万	1	5
天津	5.7万	6万	4	18
上海	5万	5.1万	9	31
青海	4.6万	4.5万	1	3
西藏	2.8万	2.4万	1	3



广东省8月份各地市温度变化情况



```
from pyecharts import options as opts
from pyecharts.charts import Geo
from pyecharts.globals import ChartType
import random
```

```
class Data:
```

```
    guangdong_city = ["佛山市","湛江市","潮州市","河源市","江门市","中山市","珠海市",  
"深圳市","东莞市","韶关市","清远市","云浮市","茂名市","汕头市","汕尾市","  
揭阳市","阳江市","肇庆市","广州市","惠州市","梅州市"]
```

```
    def values(start: int = 30, end: int = 40) -> list:
```

```
        return [random.randint(start, end) for _ in range(21)]
```



```
def geo_guangdong(title) -> Geo:
    c = (Geo()
        .add_schema(maptype="广东")
        .add(
            title,[list(z) for z in zip(Data.guangdong_city, Data.values())],
            type_=ChartType.HEATMAP)
        .set_global_opts(
            visualmap_opts=opts.VisualMapOpts(max_=42,is_pieewise=True),
            title_opts=opts.TitleOpts(title="广东省8月份各地市温度变化情况"),
        )
    )
    return c
```



```
for i in range(5):  
    str_date="8月" + str(i+1) + "日"  
    geo_guangdong(str_date).render(str_date+".html")
```



先安装一个snapshot_phantomjs包，然后还需要下载phantomjs.exe文件，保存到工程所在目录的Scripts 文件夹当中。例如我的工程目录是C:\Users\yy\PycharmProjects\mymatplotlib\venv\Scripts。

```
from pyecharts.render import make_snapshot
from snapshot_phantomjs import snapshot

for i in range(5):
    str_date="8月" + str(i+1) + "日"
    make_snapshot(snapshot, geo_guangdong(str_date).render(),
                  str(i+1)+".png",pixel_ratio=1)
```



作业10

展示自己家乡所在省的温度变化热力图，要求至少有10天的数据。



第14章 数据可视化

9. 三维图形

`mpl_toolkits.mplot3d`是Matplotlib里面专门用来画三维图的工具包

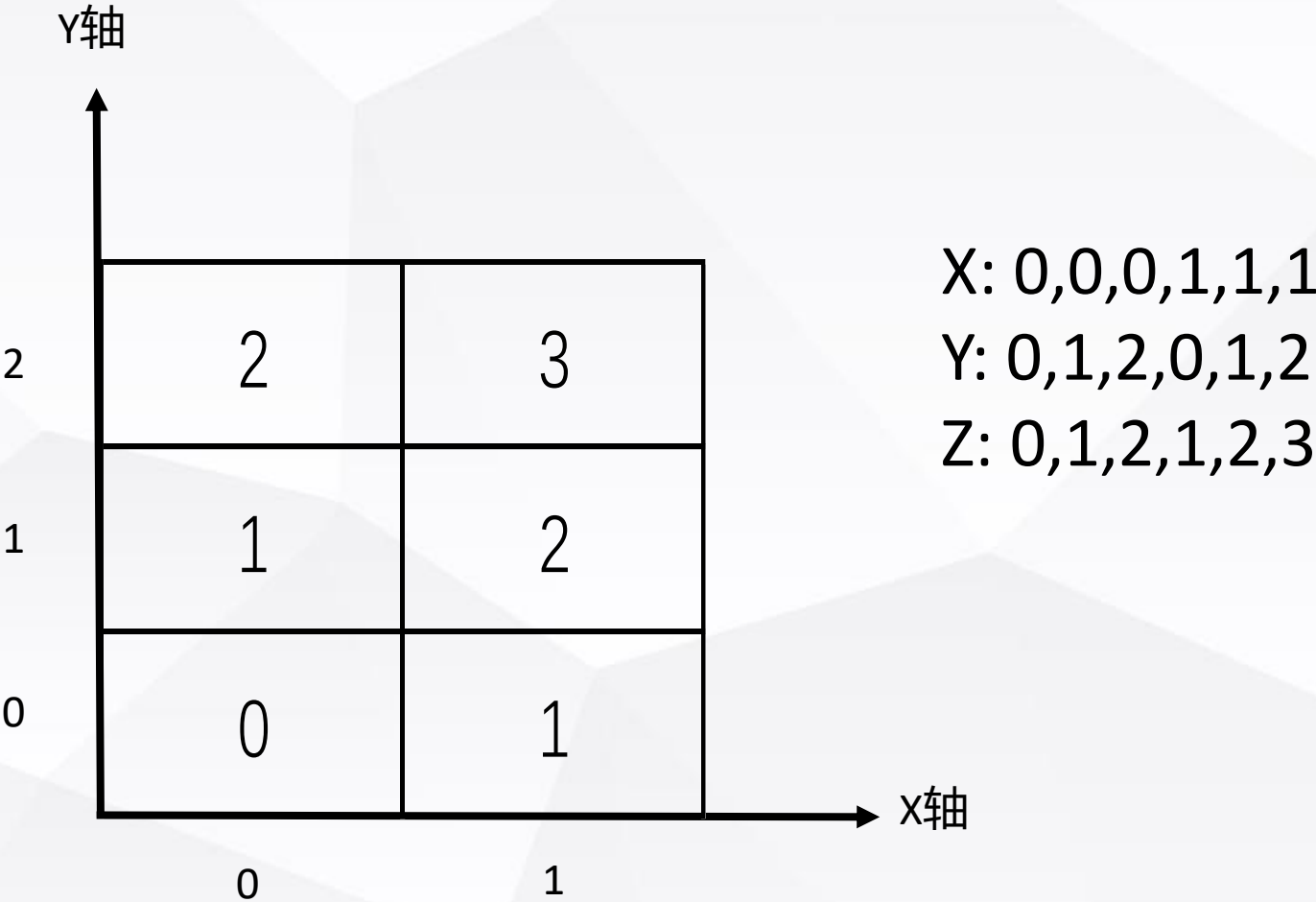
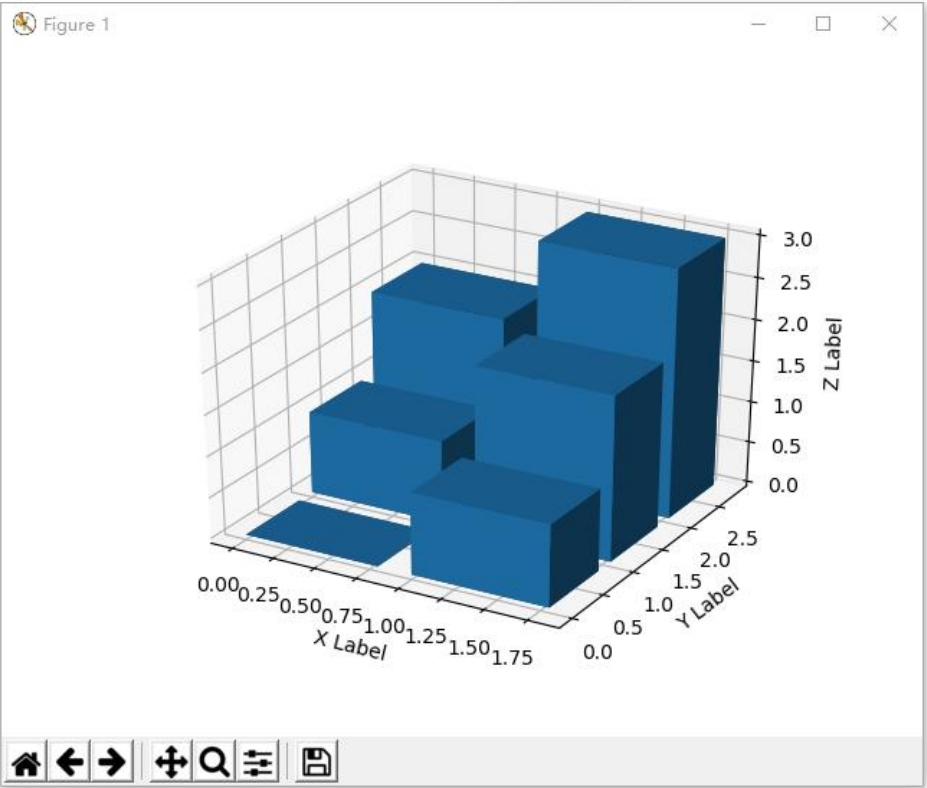
```
from mpl_toolkits.mplot3d import Axes3D
```

三维图形，需要提供x,y,z轴的数据。

- 直方图
- 线图
- 散点图



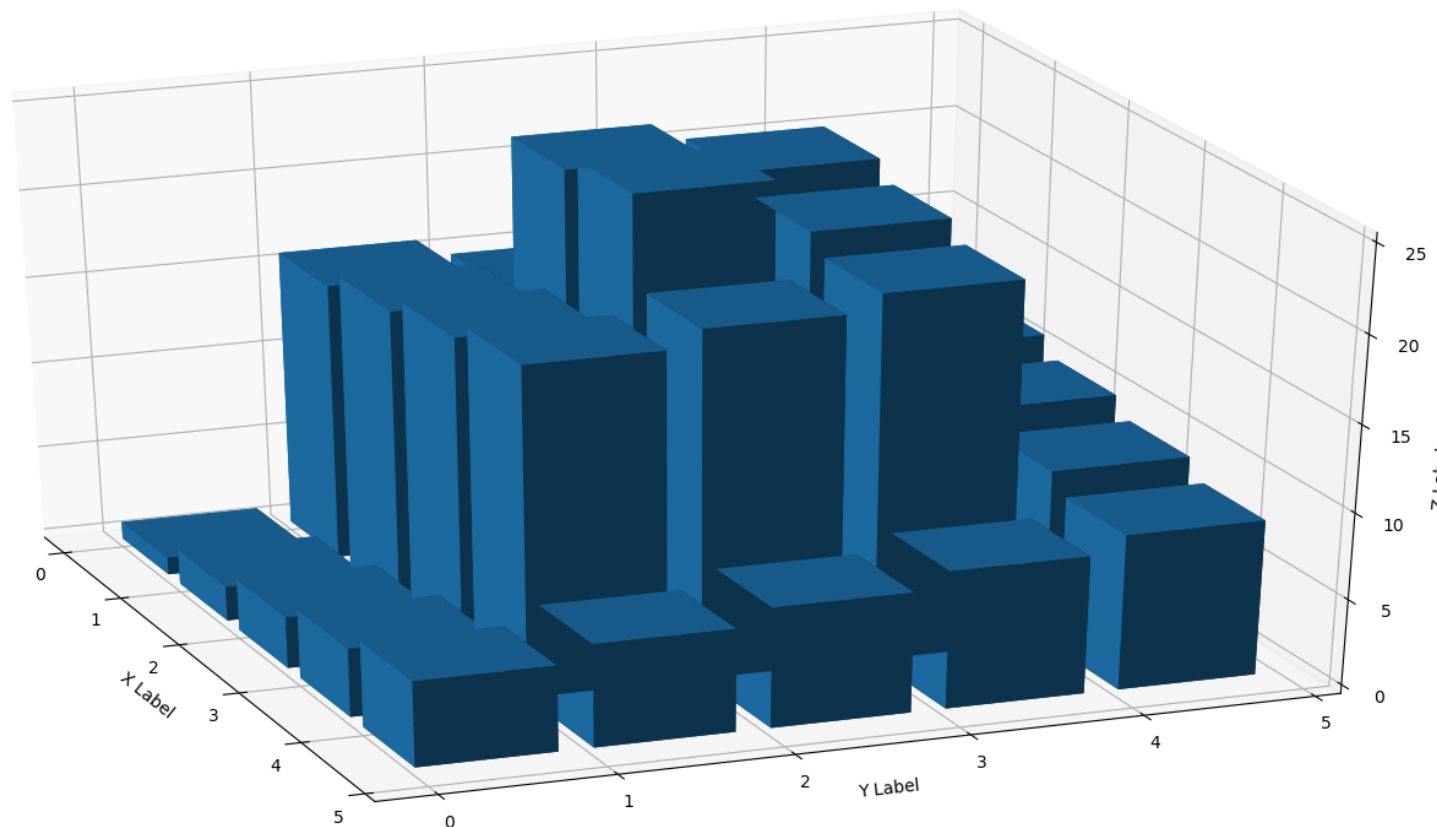
直方图



14.9 三维图形

作业

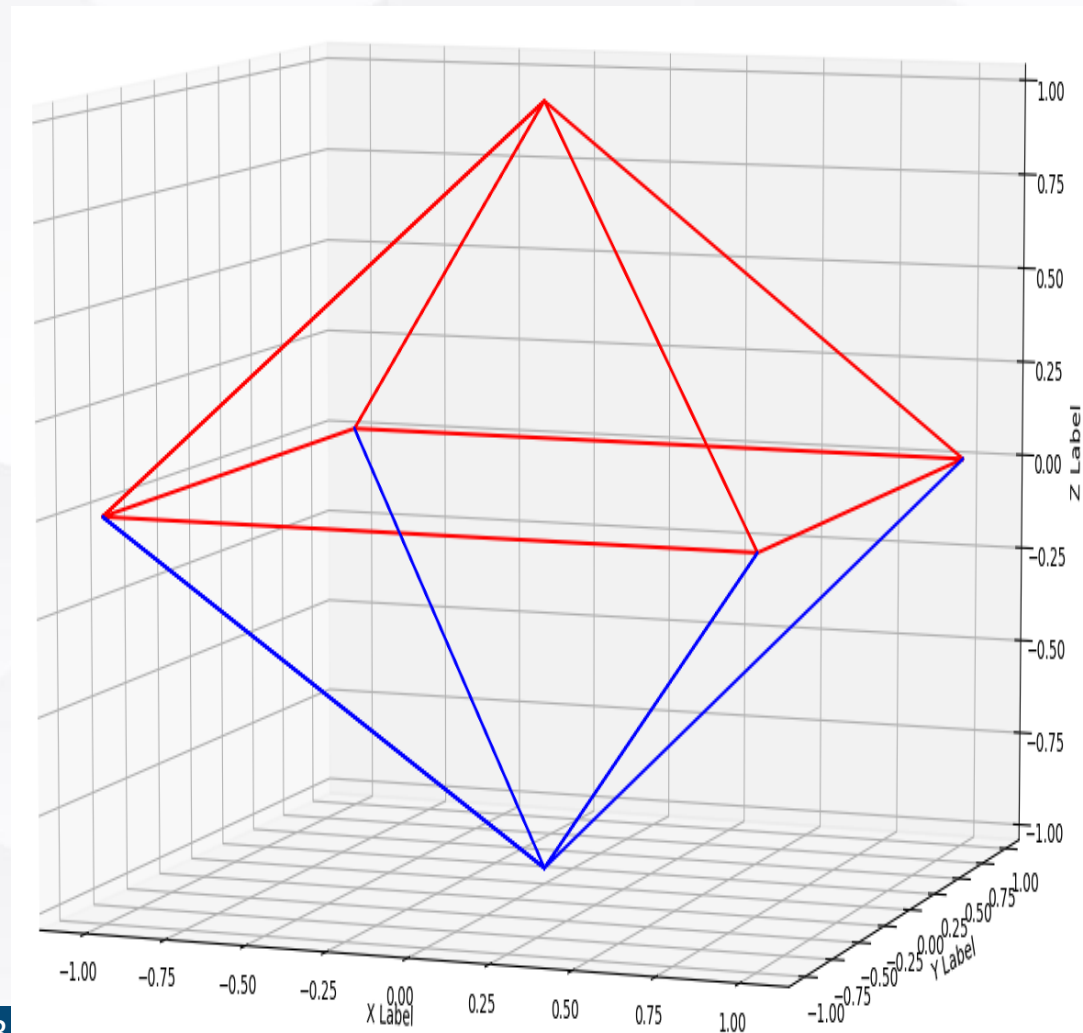
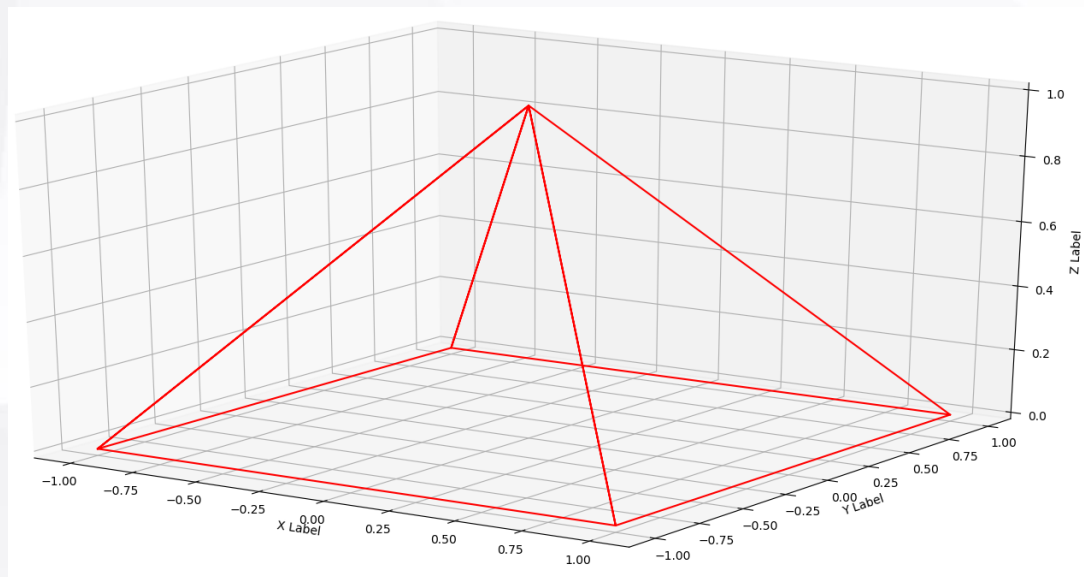
生成一个直方图，有25根直方柱。要求直方柱的最小值是1，最大值是25，要求沿着边缘，从外到内逐步增大



14.9 三维图形

作业11

- 1: 生成一个金字塔的线图
- 2: 生成一上一下两个金字塔，叠放在一起



14.9 三维图形

作业12

生成一个散点图，如下图所示。 $z=x^2+y^2$

