1. 数据可视化

文不如字，字不如表，表不如图，这句话充分说明了可视化的重要性。从事与数据相关的工作者经常会作一些总结或者展望性的报告，如果报告中密密麻麻的都是文字，相信听众和老板一定会烦，如果报告中呈现的是大量的图形化结果，就会边到众人的喜爱，因为图形更加直观醒目。本章的重点就是利用 Python 绘制常见的统计图形，如条形图、饼图、直方图、折线图、散点图等。这些图形的绘制可以通过 matplotlib 模块、pandas 模块或者 seaborn 模块实现。

Matplotlib 是最流行的用于制图及其它二维数组可视化的库，它与生态系统的其他库良好整合。

1.1. 离散型变量

1.1.1. 饼图

饼图，也叫饼状图，是一个划分为几个扇区的圆形统计图表，用于描述量、频率、或百分比之间的相对关系。在饼图中，每个扇区的弧长（圆心角和面积）大小为其所表示的数量的比例。这些扇区合在一起刚好是一个完整的圆形，这些扇区就好像拼成一个切开的饼形图案。

1、Matplotlib 模块

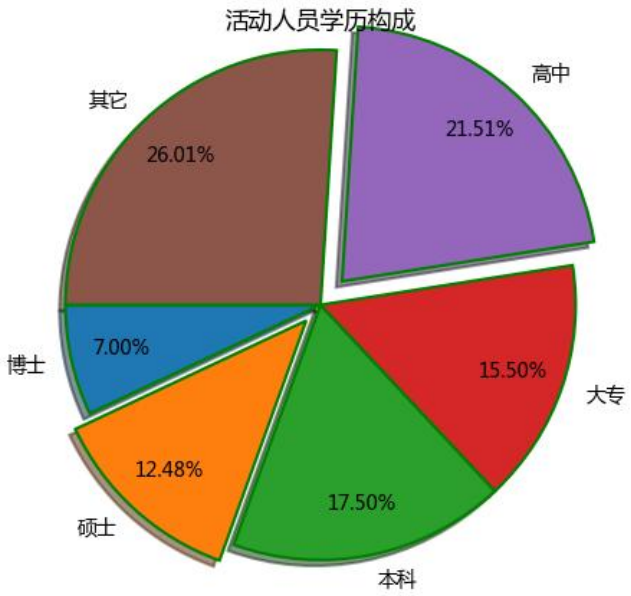
首先要导入 matplotlib 模块的子模块 pyplot，然后利用模块中的 pie 函数。

（1）示例 1

```
import matplotlib.pyplot as plt
# 处理中文乱码
plt.rcParams['font.sans-serif'] = ['Microsoft YaHei']
#构造数据
data = [255,455,638,565,784,948]
labels = ['博士','硕士','本科','大专','高中','其它']

#绘制饼图
plt.pie(x = data,                  # 数据
        labels = labels,           # 标签
        autopct = '%.2f%%'         # 格式
        )
plt.show()
```

（2）示例 2

```
import matplotlib.pyplot as plt
# 处理中文乱码
plt.rcParams['font.sans-serif'] = ['Microsoft YaHei']
plt.rcParams['axes.unicode_minus'] = False
# 标准化处理，确保饼图是个正圆，否则是椭圆
plt.axes(aspect='equal')

#构造数据
data = [255,455,638,565,784,948]
labels = ['博士','硕士','本科','大专','高中','其它']
# 设置突出显示的部分
explode = [0,0.1,0,0,0.15,0]

#绘制饼图
plt.pie(x = data,                  # 数据
        labels = labels,           # 标签
        autopct = '%.2f%%',        # 格式
        explode = explode ,        # 突出显示某个部分
        pctdistance = 0.8,         # 百分比标签到圆心的距离
        labeldistance = 1.1,       # 标签到圆心的距离
        startangle = 180,          # 饼图初始角度
        radius = 1.2,              # 饼图半径
        shadow= True,              # 阴影效果
        wedgeprops = {'linewidth':1.5, 'edgecolor':'green'}, # 内外边界属性值
        textprops = {'fontsize':10, 'color':'black'}         # 设置文本标签属性值
        )
plt.title('活动人员学历构成')
plt.show()
```



注意：

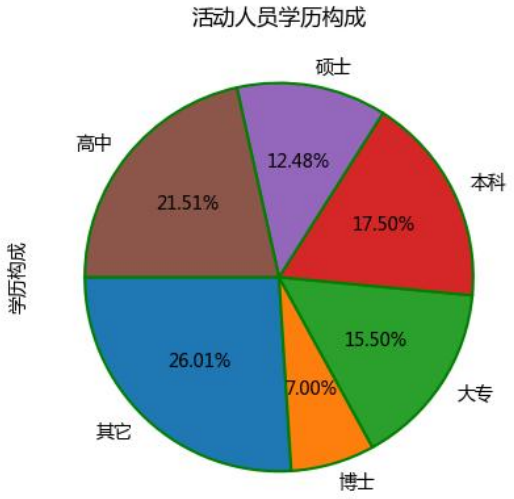
（A）如果绘制的图中涉及中文及数字中的负号，需要通过 rcParams 进行了控制；

（B）不加修饰的饼图更像是一个椭圆，所以需要利用 axes 函数强制为正圆。

2、Pandas 模块

（1）示例

```
import pandas as pd
import matplotlib.pyplot as plt

# 处理中文乱码
plt.rcParams['font.sans-serif'] = ['Microsoft YaHei']

data = pd.Series({'博士':255,'硕士':455,'本科':638,'大专':565,'高中':784,'其它
':948,})
data.name = '学历构成'
data.plot(kind = 'pie',
          autopct = '%.2f%%',
          radius = 1,
          startangle = 180,
          title = '活动人员学历构成',
          wedgeprops = {'linewidth':1.5, 'edgecolor':'green'}, # 内外边界属性值
          # 设置文本标签属性值
          textprops = {'fontsize':10, 'color':'black'}
          )
#显示图形
plt.show()
```



1.1.2. 条形图

条形图亦称条图、条状图、棒形图、柱状图，是一种以长方形的长度为变量的统计图表。长条图用来比较两个或者两个以上的数值（不同时间或者不同条件），只有一个变量，通常利用最小的数据集分析。

长条图也可横向排列，也可用多维表达方式。

1、matplotlib 模块

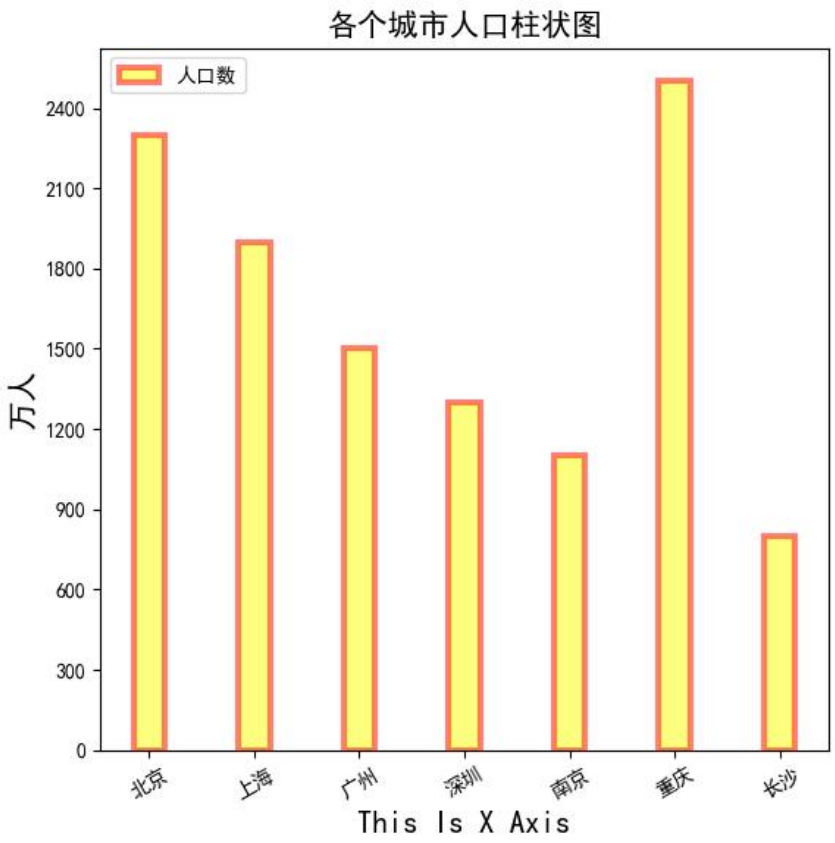
matplotlib.pyplot.bar(left, height, alpha=1, width=0.8, color=, edgecolor=, label=, lw=3)

1. left：x 轴的位置序列，一般采用 range 函数产生一个序列，但是有时候可以是字符串

2. height：y 轴的数值序列，也就是柱形图的高度，一般就是我们需要展示的数据；
3. alpha：透明度，值越小越透明
4. width：为柱形图的宽度，一般这是为 0.8 即可；
5. color 或 facecolor：柱形图填充的颜色；
6. edgecolor：图形边缘颜色
7. label：解释每个图像代表的含义，这个参数是为 legend()函数做铺垫的，表示该次 bar 的标签
8. linewidth or linewidths or lw：边缘 or 线的宽

```
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt
# 设置中文字体
mpl.rcParams['font.sans-serif'] = ['SimHei']

xdata = ["北京","上海","广州","深圳","南京","重庆","长沙"]
ydata = [2300,1900,1500,1300,1100,2500,800]

plt.bar(xdata, ydata, alpha=0.5, width=0.3, color='yellow', edgecolor='red',
```

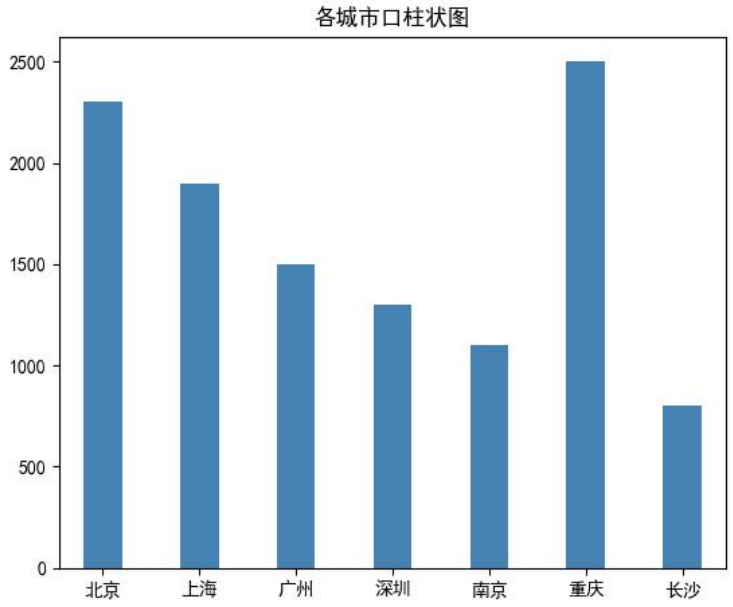
```
label='人口数', lw=3)
plt.legend(loc='upper left')

plt.xlabel('This Is X Axis', fontsize=15)
plt.ylabel('万人', fontsize=15)
plt.title('各个城市人口柱状图', fontsize=15)

# 如果想自己给 x 轴的 bar 加上标签,rotation 控制倾斜角度
plt.xticks(rotation=30)
plt.yticks(np.arange(0, 2500, 300))

plt.show()
```



2、pandas 模块

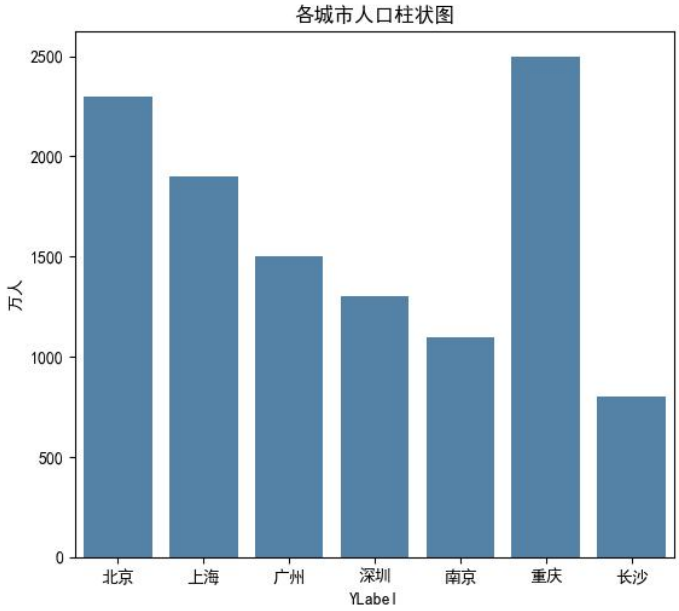
```
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt
# 设置中文字体
mpl.rcParams['font.sans-serif'] = ['SimHei']

data = pd.Series([2300,1900,1500,1300,1100,2500,800],
                 index = ["北京","上海","广州","深圳","南京","重庆","长沙"])
```

```
data.plot(kind = 'bar',
          width =0.4,
          rot = 0,
          color='steelblue',
          title = '各城市口柱状图')
plt.show()
```



3、seaborn 模块

```
import seaborn as sns
import matplotlib as mpl
import matplotlib.pyplot as plt

# 设置中文字体
mpl.rcParams['font.sans-serif'] = ['SimHei']

xdata = ["北京","上海","广州","深圳","南京","重庆","长沙"]
ydata = [2300,1900,1500,1300,1100,2500,800]

sns.barplot(y = ydata,
            x = xdata,
            color = 'steelblue'
            )

plt.xlabel('YLabel')
plt.ylabel('万人')
plt.title('各城市人口柱状图')

plt.show()
```



1.2. 数值型变量

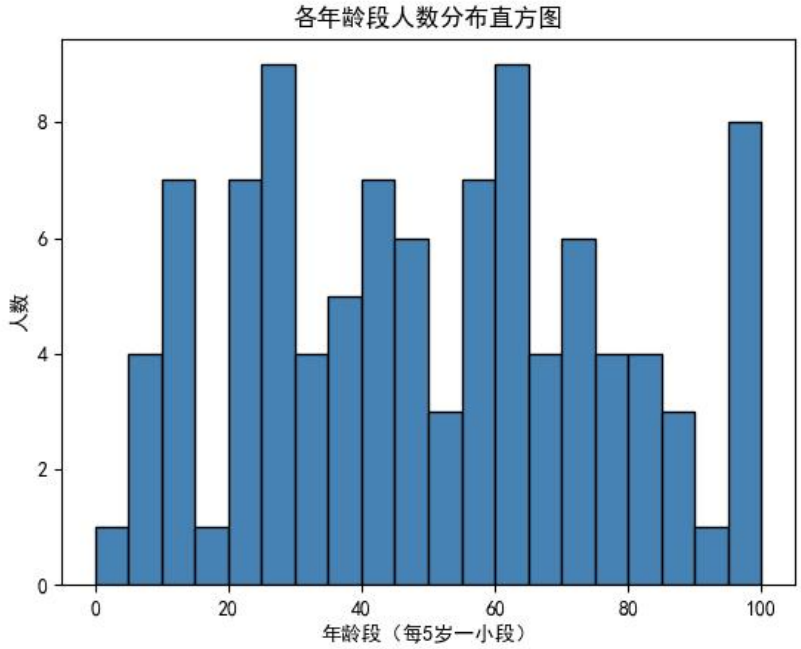
1.2.1. 直方图

直方图一般用来观察数据的分布形态，横坐标代表数值的均匀分段，纵坐标代表每个段内的观测数量，一般直方图都会与核密度图搭配使用，目的是为了更加清晰地掌握数据的分布特征。

1、matplotlib 模块

```
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt
import random

# 设置中文字体
mpl.rcParams['font.sans-serif'] = ['SimHei']

data = pd.DataFrame({'age':[random.randint(0, 100) for i in range(100)]})

plt.hist(x = data['age'],          # 数据
         bins = 20,                # 列数
         range = (0,100),          # 范围
         color = 'steelblue',      # 颜色
         edgecolor = 'black')      # 边界颜色

plt.xlabel('年龄段（每 5 岁一小段）')
```

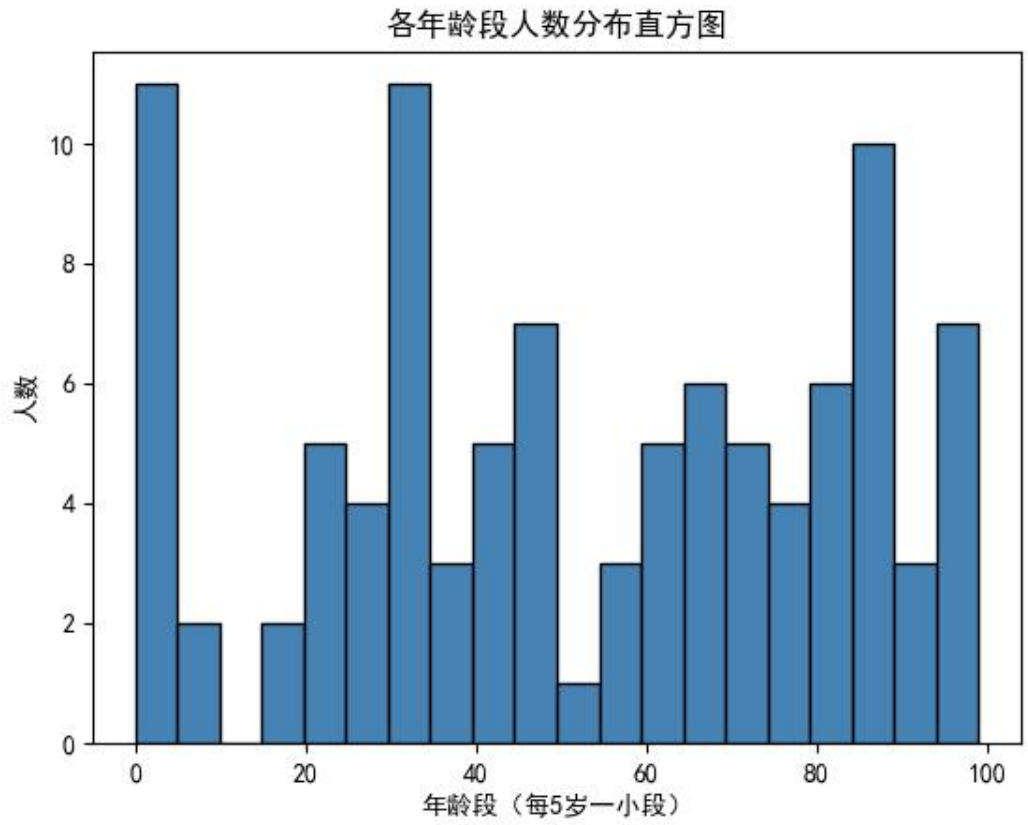
```
plt.ylabel('人数')
plt.title('各年龄段人数分布直方图')

plt.show()
```



2、pandas 模块

```
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt
import random

# 设置中文字体
mpl.rcParams['font.sans-serif'] = ['SimHei']

data = pd.Series([random.randint(0, 100) for i in range(100)])

data.plot(kind = 'hist',
          bins = 20,
          color = 'steelblue',
          edgecolor = 'black',
          label = '直方图'
          )

# data.plot(kind = 'kde',color = 'red',label = '核密度图')
```

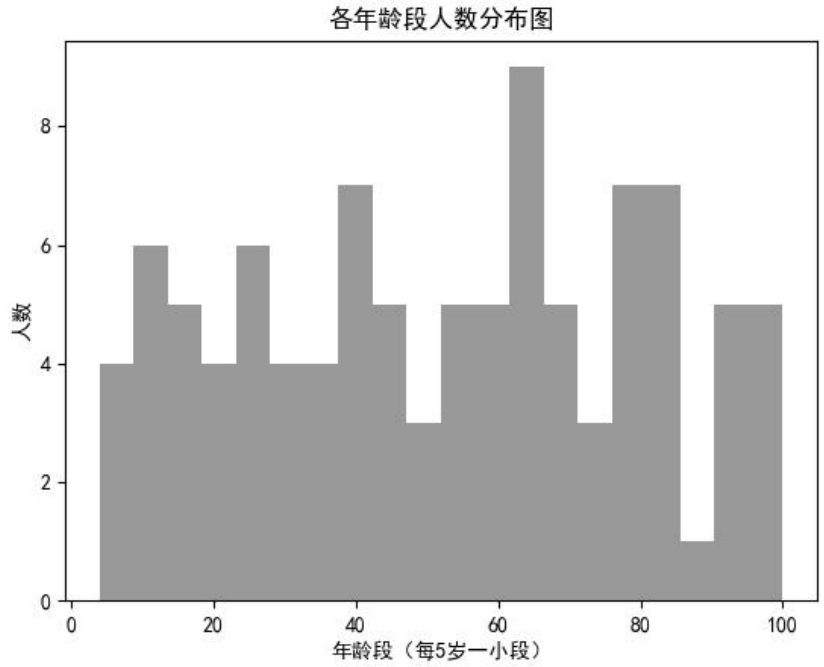
```
plt.xlabel('年龄段（每 5 岁一小段）')
plt.ylabel('人数')
plt.title('各年龄段人数分布直方图')

plt.show()
```



3、seaborn 模块

```
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt
import seaborn as sbn
import random

# 设置中文字体
mpl.rcParams['font.sans-serif'] = ['SimHei']

data = pd.Series([random.randint(0, 100) for i in range(100)])
```

```
sbn.distplot(data,
             bins = 20,
             kde = False,
             hist_kws = {'color':'black'},
             label = '年龄分布'
)

plt.xlabel('年龄段（每 5 岁一小段）')
plt.ylabel('人数')
plt.title('各年龄段人数分布图')

plt.show()
```

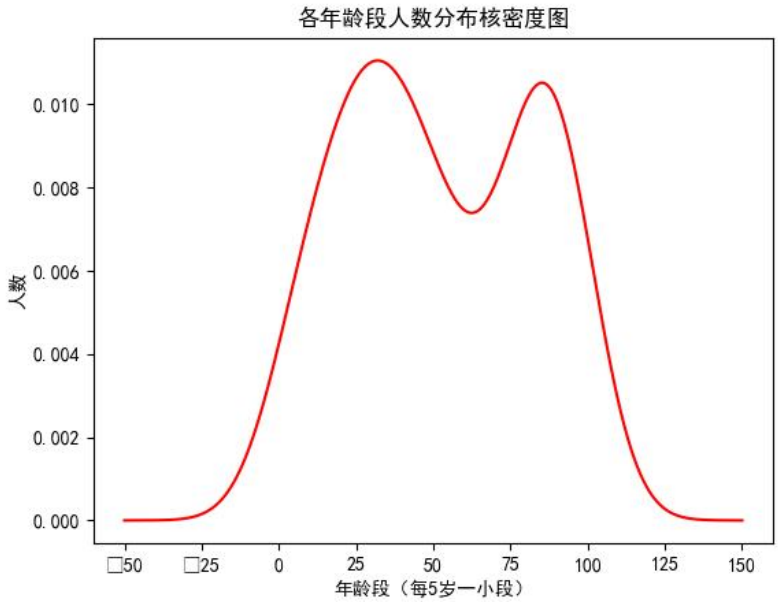


1.2.2. 核密度图

1、pandas 模块

```
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt
import random

# 设置中文字体
mpl.rcParams['font.sans-serif'] = ['SimHei']

data = pd.Series([random.randint(0, 100) for i in range(100)])
```

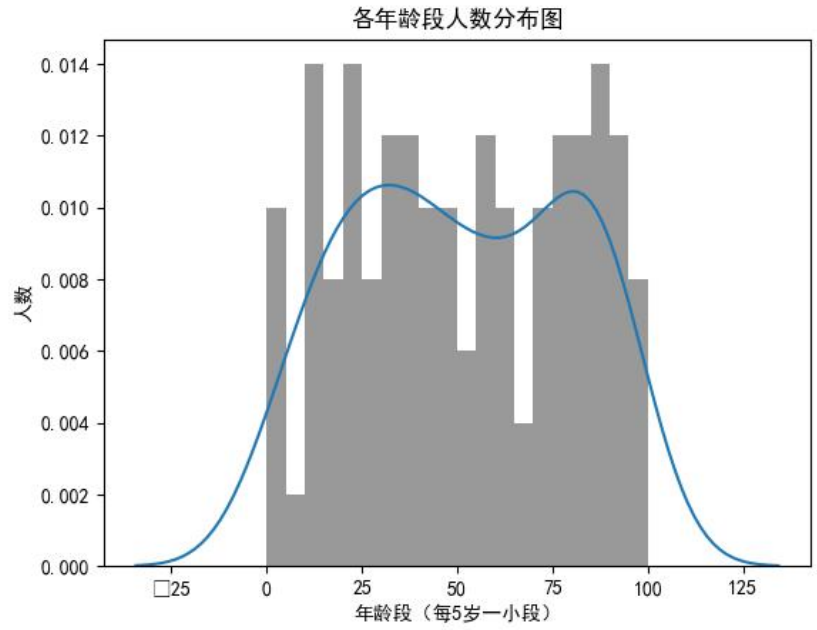
```
data.plot(kind = 'kde',
          color = 'red',
          label = '核密度图')

plt.xlabel('年龄段（每 5 岁一小段）')
plt.ylabel('人数')
plt.title('各年龄段人数分布核密度图')

plt.show()
```



2、seaborn 模块

```
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt
import seaborn as sbn
import random

# 设置中文字体
mpl.rcParams['font.sans-serif'] = ['SimHei']

data = pd.Series([random.randint(0, 100) for i in range(100)])

sbn.distplot(data,
             bins = 20,
             kde = True,
             hist_kws = {'color':'black'},
             label = '年龄分布'
```

```
)

plt.xlabel('年龄段（每5岁一小段）')
plt.ylabel('人数')
plt.title('各年龄段人数分布图')

plt.show()
```

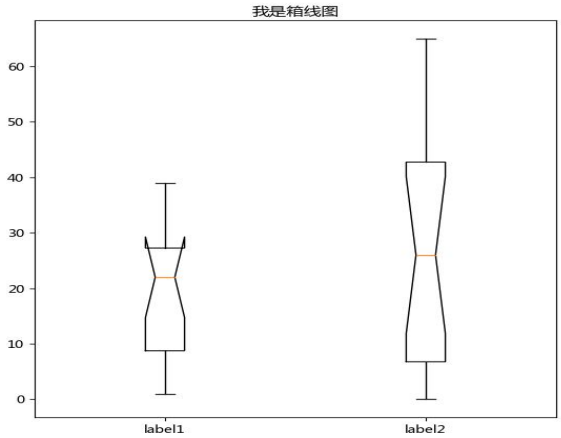


1.2.3. 箱线图

箱线图是也称箱须图，能提供有关数据位置和分散情况的关键信息，尤其是在比较不同特征时，更可表现其分散程度差异。箱线图中利用五个统计量（最小值、下四分位数、中位数、上四分位数、最大值）。

```
import matplotlib.pyplot as plt
# 处理中文乱码
plt.rcParams['font.sans-serif'] = ['Microsoft YaHei']

data = [[1,3,6,8,9,23,24,26,28,27,39,35,32,21,15,17],
        [0,4,5,6,7,21,25,26,27,26,59,65,42,41,45,57]]
labels = ['label1','label2']

plt.boxplot(data,notch=True,labels=labels,meanline=True)
```

```
plt.title("我是箱线图")
plt.show()
```



1.2.4. 小提琴图

1.2.5. 折线图

对于时间序列数据而言，一般都会使用折线图反映数据背后的超势。通常折线的横坐标指代日期数据，纵坐标代表某个数值型变量，当然，还可以使用第三个离散变量对折线图进行分组处理。

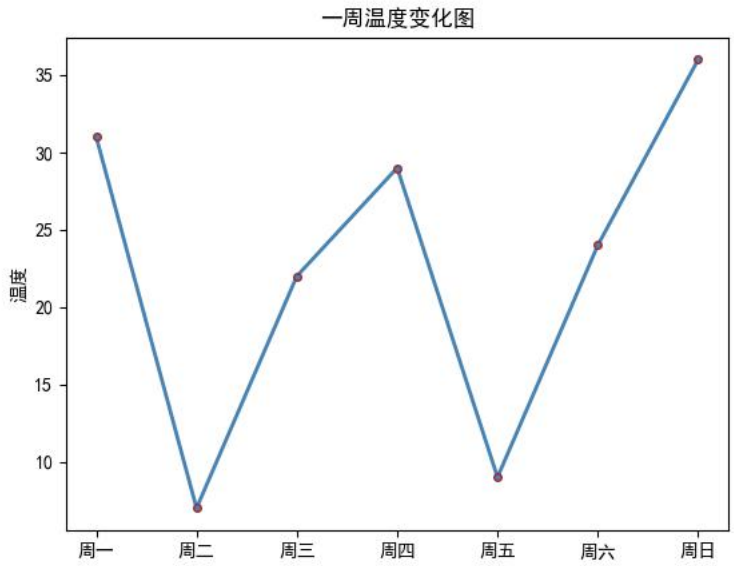
Python 中的 matplotlib 模块和 pandas 模块实现折线图的绘制。但是 seaborn 模块也可以绘制，它的 tsplot 函数非常不合理，所以不介绍。

1、matplotlib 模块

Matplotlib 模块中的 plot 函数可以绘制折线图。

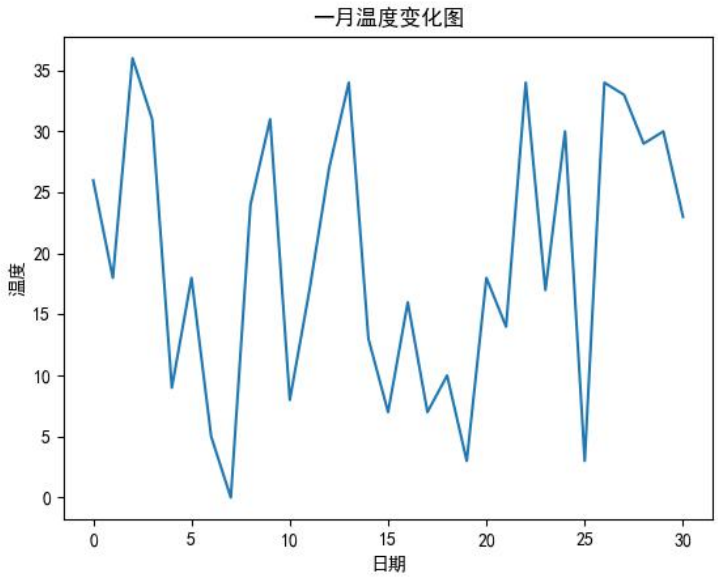
```
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt
import random

# 设置中文字体
mpl.rcParams['font.sans-serif'] = ['SimHei']

data = pd.Series([random.randint(0, 37) for i in range(7)],
                 index = ['周一','周二','周三','周四','周五','周六','周日'])

plt.plot( data.index,
          data,
          linestyle = '-',
          linewidth = 2,
          color = 'steelblue',
          marker = 'o',
```

```
        markersize = 4,
        markeredgecolor = 'brown'
)

plt.ylabel('温度')
plt.title('一周温度变化图')

plt.show()
```



2、pandas 模块

```
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt
import random

# 设置中文字体
mpl.rcParams['font.sans-serif'] = ['SimHei']
data = pd.Series([random.randint(0, 37) for i in range(31)])

data.plot()

plt.ylabel('温度')
plt.xlabel('日期')
plt.title('一月温度变化图')

plt.show()
```



1.3. 关系型数据

在众多的可视化图形中，有一类图形专门用来探究两个或者三个变量之间的关系，例如，散点图用于发现两个数值变量之间的关系，气泡图用来展现三个数值之间的关系，热力图则体现了两个离散变量之间的组合关系。

1.3.1. 散点图

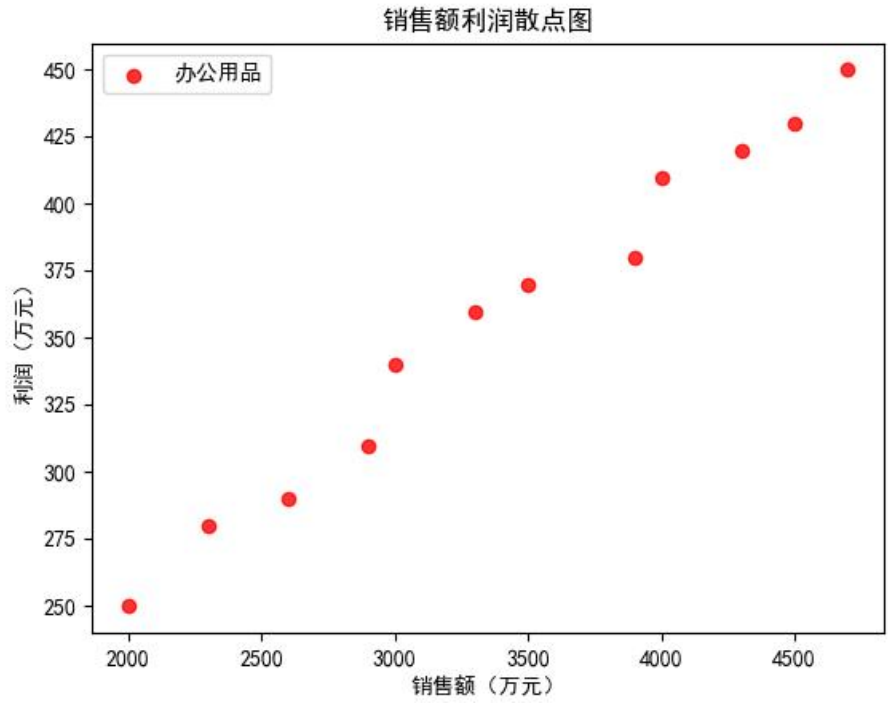
如果需要研究两个数值变量之间是否存在某种关系，例如正向的线性关系，或者是趋势性的非线性关系，那么散点图将是最佳的选择。

1、matplotlib 模块

Matplotlib 模块中的 scatter 函数可以非常方便地绘制两个数值型变量的散点图。

```
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt
import random

# 设置中文字体
mpl.rcParams['font.sans-serif'] = ['SimHei']

data = pd.DataFrame(
{'销售额':[2000,2300,2600,2900,3000,3300,3500,3900,4000,4300,4500,4700],
'利润': [250,280,290,310,340,360,370,380,410,420,430,450]},
```

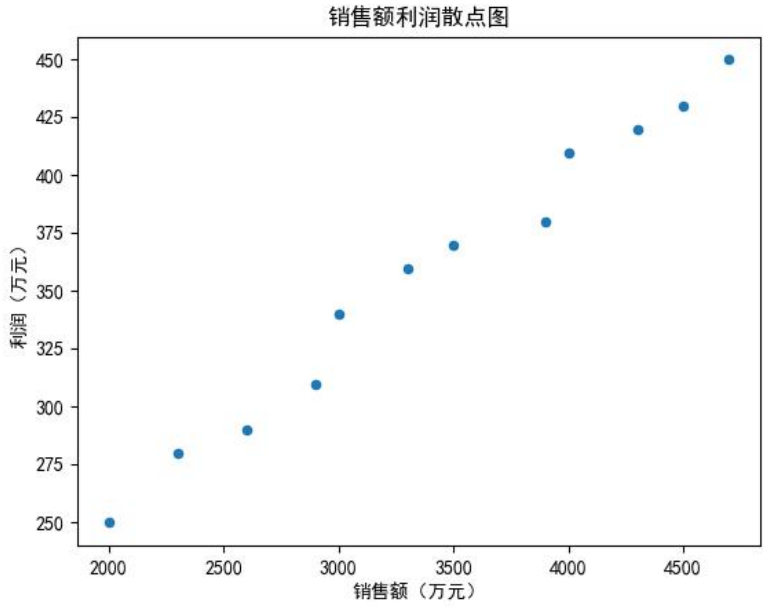
```
                    index = [1,2,3,4,5,6,7,8,9,10,11,12]
                )
print(data)

plt.scatter(x=data['销售额'],
            y=data['利润'],
            label ='办公用品',
            color ='red',
            alpha= 0.8
)

plt.xlabel('销售额（万元）')
plt.ylabel('利润（万元）')

plt.title("销售额利润散点图")
plt.legend()
plt.show()
```



2、pandas 模块

```
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt
import random

# 设置中文字体
```

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

data = pd.DataFrame({'销售额
':[2000,2300,2600,2900,3000,3300,3500,3900,4000,4300,4500,4700],
                     '利润': [250,280,290,310,340,360,370,380,410,420,430,450]},
                    index = [1,2,3,4,5,6,7,8,9,10,11,12]
                   )
print(data)

data.plot(x = '销售额',
          y = '利润',
          kind = 'scatter')

plt.xlabel('销售额（万元）')
plt.ylabel('利润（万元）')

plt.title("销售额利润散点图")
plt.show()
```

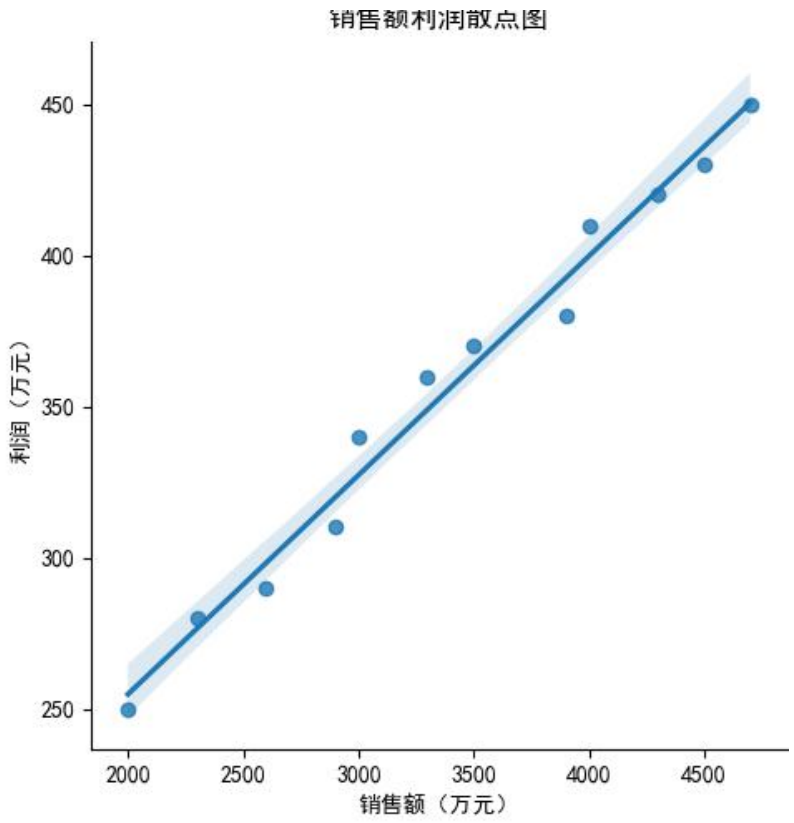


3、seaborn 模块

尽管使用 matplotlib 和 pandas 可以非常方便地绘制出散点图，但是绘制分组散点图会稍微复杂一点了。如果使用 seaborn 中的 lmplot 函数，那就非常方便了，而且，它还可以根据散点图添加线性拟合线。

Lmplot 函数还可以指定 lowess 参数拟合局部多项式回归，指定 logistic 参数拟合逻辑回归，指定 order 参数做多项式回归，指定 robust 做鲁棒回归。

```
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt
import seaborn as sns

# 设置中文字体
mpl.rcParams['font.sans-serif'] = ['SimHei']

data = pd.DataFrame({'销售额
':[2000,2300,2600,2900,3000,3300,3500,3900,4000,4300,4500,4700],
                     '利润': [250,280,290,310,340,360,370,380,410,420,430,450]},
                    index = [1,2,3,4,5,6,7,8,9,10,11,12]
                  )

sns.lmplot(x = '销售额',
           y = '利润',
           data = data,
           legend_out= False,
           truncate= True )

plt.xlabel('销售额（万元）')
plt.ylabel('利润（万元）')
plt.title("销售额利润散点图")
plt.show()
```

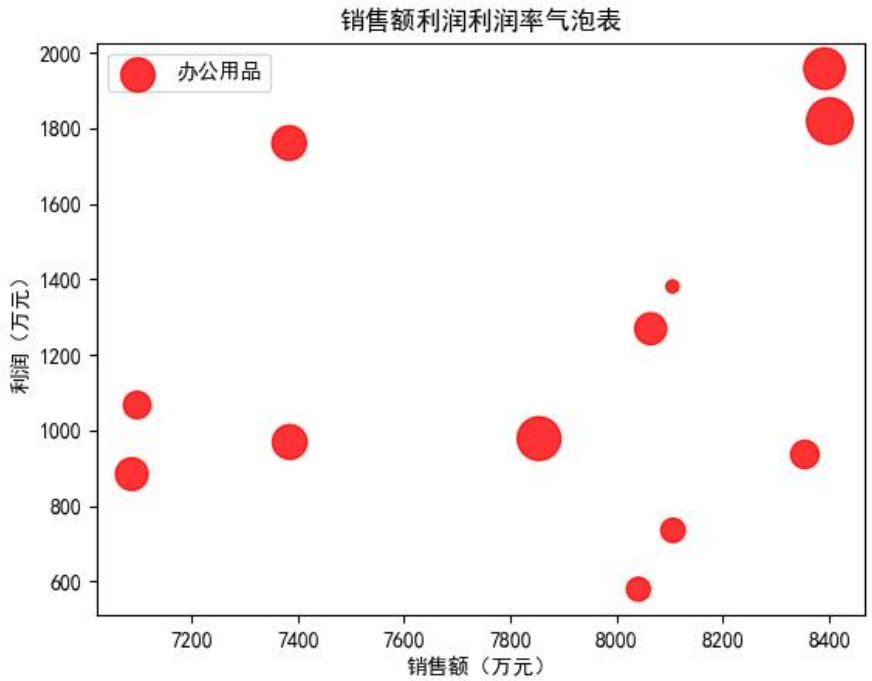


1.3.2. 气泡图

散点图一般用来反应两个离散型数据变量的关系，如果还想通过散点图添加第三个数值变量的信息，一般可以使用用气泡图。气泡图的实质就是通过第三个数值型变量控制每个散点的大小，点越大，代表第三维的数值越高，反之亦然。

```
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt
import random

# 设置中文字体
mpl.rcParams['font.sans-serif'] = ['SimHei']

data = pd.DataFrame({'销售额':[random.randint(7000,9000) for i in range(12)],
                     '利润': [random.randint(500, 2000) for i in range(12)],
                     '利润率': [random.randint(1, 100)*5 for i in range(12)]},
                    index = [1,2,3,4,5,6,7,8,9,10,11,12]
                )
print(data)

plt.scatter(x=data['销售额'],
            y=data['利润'],
```

```
            s=data['利润率'],
            label = '办公用品',
            color = 'red',
            alpha= 0.8
)

plt.xlabel('销售额（万元）')
plt.ylabel('利润（万元）')

plt.title("销售额利润利润率气泡表")
plt.legend()
plt.show()
```



pandas 与 seaborn 中没有绘制气泡图的方法或函数。

Python 中的 bokeh 模块，也可以实现绘制气泡图的功能。

1.3.3. 热力图

热力图，也叫做交叉填充图，它的用法是实现列联表的可视化。通过图形的方式展现两个离散变量之间的组合关系，

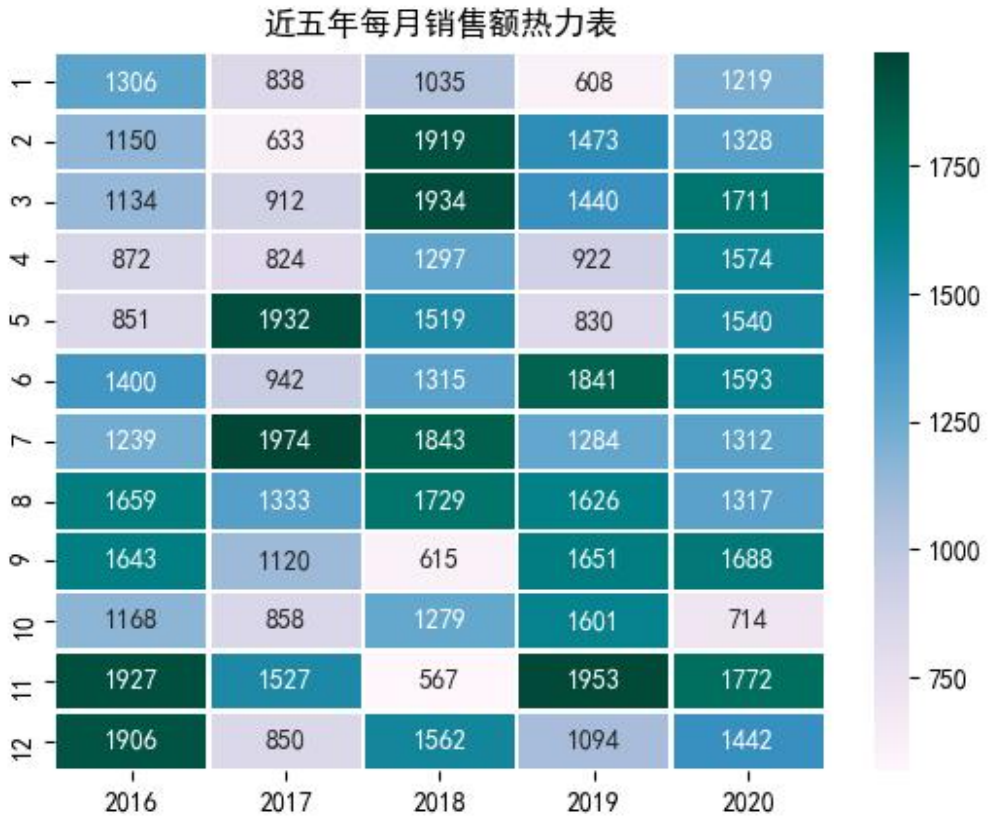
Seaborn 中的 heatmap 函数，可以完成热力图的绘制。

```
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt
import seaborn as sns
```

```
import random

# 设置中文字体
mpl.rcParams['font.sans-serif'] = ['SimHei']
data = pd.DataFrame({'2016':[random.randint(500,2000) for i in range(12)],
                     '2017': [random.randint(500, 2000) for i in range(12)],
                     '2018': [random.randint(500, 2000) for i in range(12)],
                     '2019': [random.randint(500, 2000) for i in range(12)],
                     '2020': [random.randint(500, 2000) for i in range(12)],},
                    index = [1,2,3,4,5,6,7,8,9,10,11,12]
                   )
print(data)
sns.heatmap(data = data,
            linewidths=1,
            cmap= 'PuBuGn', # 填充颜色
            annot = True,   # 显示数值
            fmt= ''         # 不用科学计数法
            )


plt.title("近五年每月销售额热力表")
plt.show()
```

输出:

	2016	2017	2018	2019	2020
1	1306	838	1035	608	1219
2	1150	633	1919	1473	1328
3	1134	912	1934	1440	1711
4	872	824	1297	922	1574
5	851	1932	1519	830	1540
6	1400	942	1315	1841	1593
7	1239	1974	1843	1284	1312
8	1659	1333	1729	1626	1317
9	1643	1120	615	1651	1688
10	1168	858	1279	1601	714
11	1927	1527	567	1953	1772
12	1906	850	1562	1094	1442



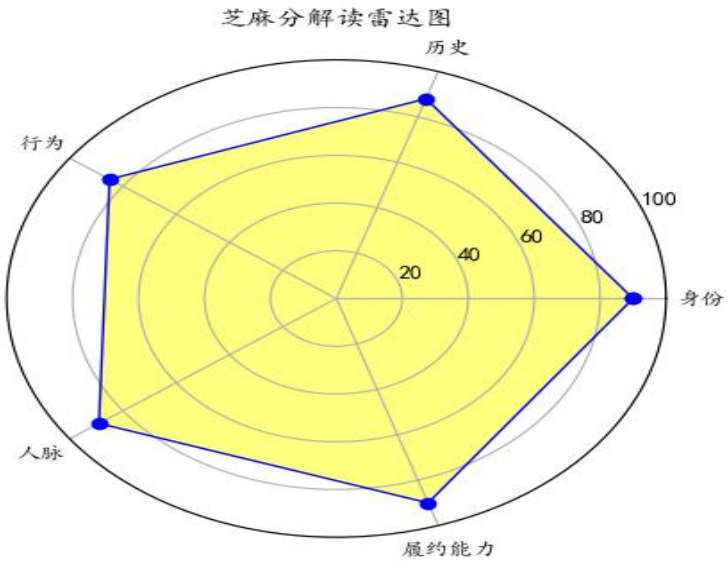
1.4. 其它数据

1.4.1. 雷达图

雷达图是以从同一点开始的轴上表示的三个或更多个定量变量的二维图表的形式显示多变量数据的图形方法。轴的相对位置和角度通常是无信息的。 雷达图也称为网络图，蜘蛛图，星图，星图，蜘蛛网图，不规则多边形，极坐标图或 Kiviat 图。它相当于平行坐标图，轴径向排列。雷达图主要应用于企业经营状况--收益性、生产性、流动性、安全性和成长性的评价。上述指标的分布组合在一起非常象雷达的形状，因此而得名。

```
import numpy as np
import matplotlib.pyplot as plt
# 显示中文
plt.rcParams['font.sans-serif'] = ['KaiTi']
# 标签
labels = np.array(['身份', '历史', '行为', '人脉', '履约能力', ])
data = np.array([90, 88, 85, 89, 90])
dataLenth = 5    # 数据长度
# 分割圆周长
angles = np.linspace(0, 2*np.pi, dataLenth, endpoint=False)
# 闭合
data = np.concatenate((data, [data[0]]))
```

```
angles = np.concatenate((angles, [angles[0]]))
# 做极坐标系
plt.polar(angles, data, 'bo-', linewidth=1)
# 做标签
plt.thetagrids(angles * 180/np.pi, labels)
# 填充
plt.fill(angles, data, facecolor='yellow', alpha=0.5)
plt.ylim(0, 100)
plt.title("芝麻分解读雷达图")
plt.show()
```




```
import matplotlib.pyplot as plt
import numpy as np
employee = ["Rahul", "Joy", "Abhishek","Tina", "Sneha"]
actual = [41, 57, 59, 63, 52, 41]
expected = [40, 59, 58, 64, 55, 40]

plt.figure(figsize=(10, 6))
plt.subplot(polar=True)
# 分割圆周长
theta = np.linspace(0, 2 * np.pi, len(actual))
lines, labels = plt.thetagrids(range(0, 360, int(360 / len(employee))),
                               (employee))
plt.plot(theta, actual)
plt.fill(theta, actual, 'b', alpha=0.1)
plt.plot(theta, expected)

plt.legend(labels=('Actual', 'Expected'),loc=1)
```

```
plt.title("Actual vs Expected sales by Employee")
plt.show()
```