大数据可视化技术

第五章

关系数据可视化

目 录

CONTENT

关系数据在大数据中的应用

关系数据在大数据中的应用

对于关系数据，我们要做的是尝试着探索事物的相关关系，而不再关注难以捉摸的因果关系。这种相关性往往不能告诉读者事物为何产生，但是会给读者一个事物正在发生的提醒。关系数据很容易通过数据进行验证的，也可以通过图表呈现，然后引导读者进行更加深入的研究和探讨。

分析数据时，也可以从整体进行观察，或者关注数据的分布。数据间是否存在重叠或者是否毫不相干？还可以更宽泛的角度观察各个分布数据的相关关系。其实最重要的一点，就是数据在进行可视化处理后，呈现在读者眼前的图表所表达的意义是什么。

关系数据具有关联性和分布性，下面我们将通过具体实例来了解关系数据的可视化分析，以及如何观察数据间的相关关系。

02

数据的关联性

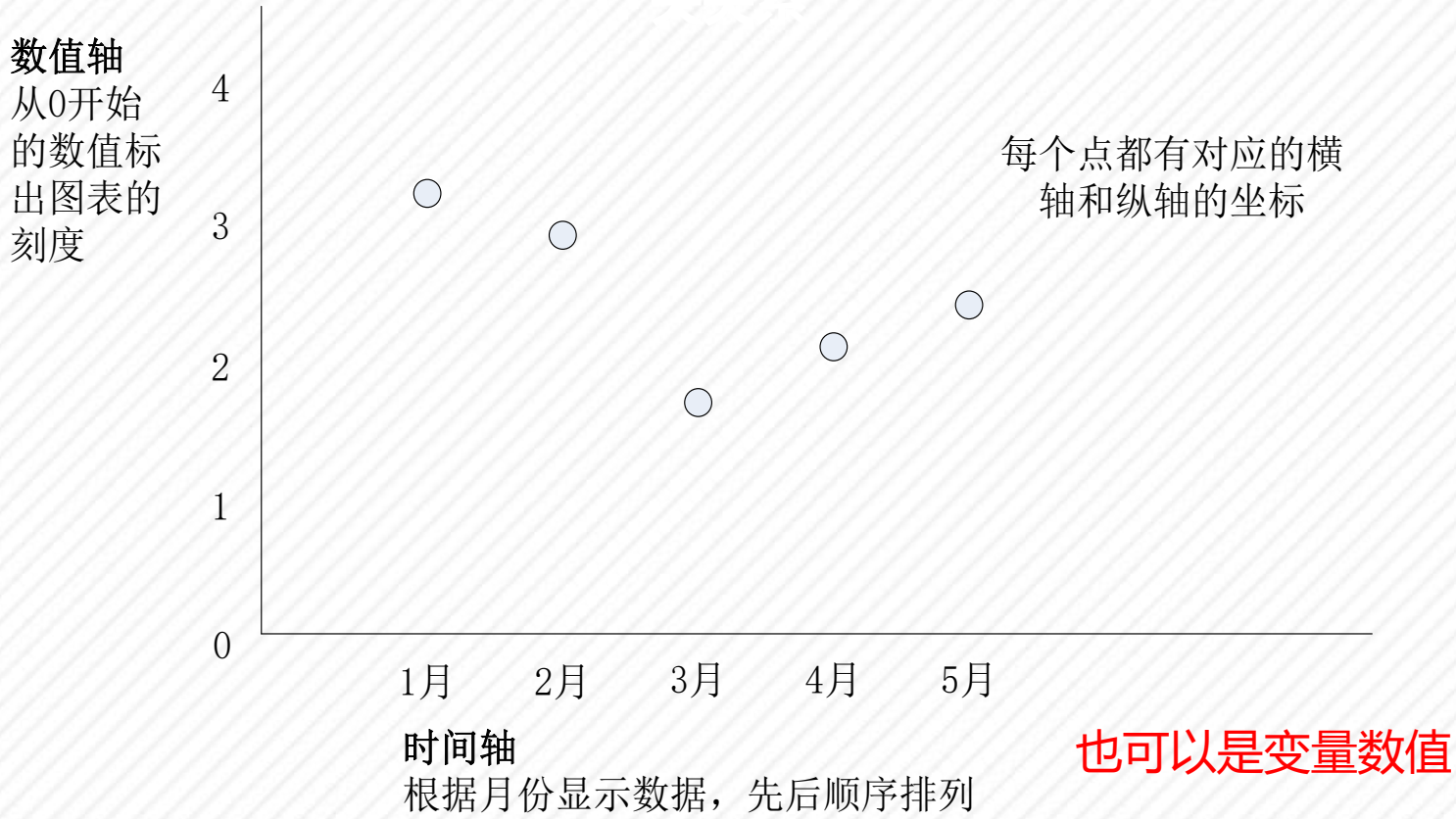
数据的关联性

数据的关联性，其核心就是指量化的两个数据间的数理关系。关联性强，是指当一个数值增长时，另一个数值也会随之发生变化。相反地，关联性弱，就是指一个数值增长时，另一个数值几乎没有发生变化。

数据的关联性主要有正相关、负相关和不相关关系。下面我们用散点图来研究数据的关联性。

散点图

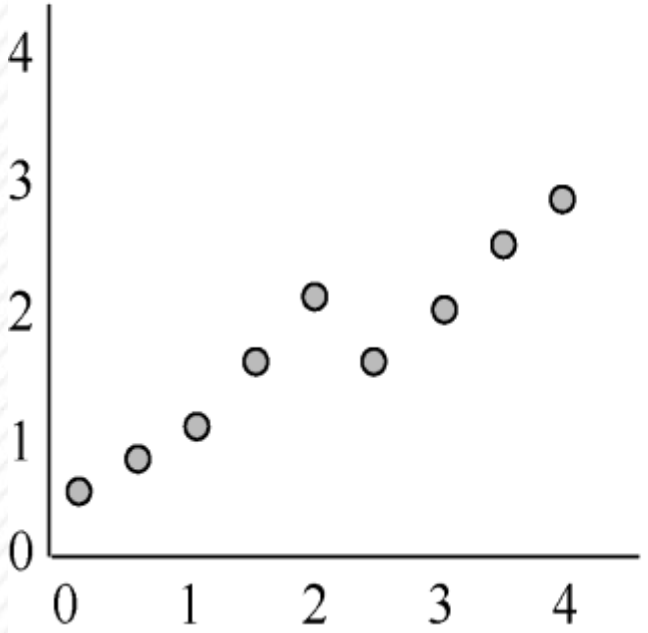
在第三章中我们已经了解到散点图是由一些散乱的点组成的图表，它的基本框架如下图所示：



散点图中显示关联性

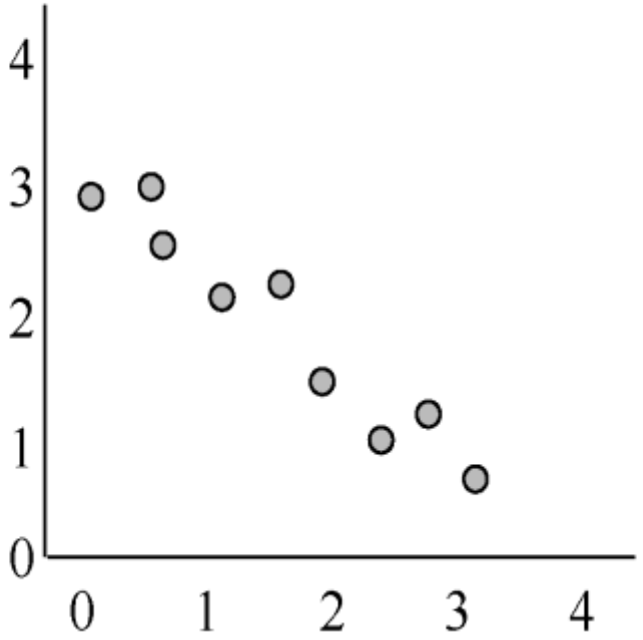
正相关

各圆点以右上的趋势上升



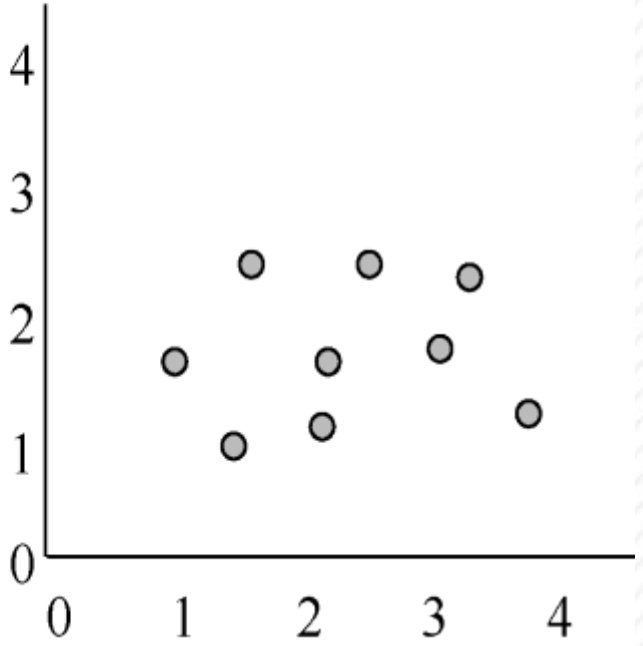
负相关

各圆点以右下的趋势下降



不相关

圆点排列无序



ggplot和pandas简介

python的常用可视化包是matplotlib，pyecharts。

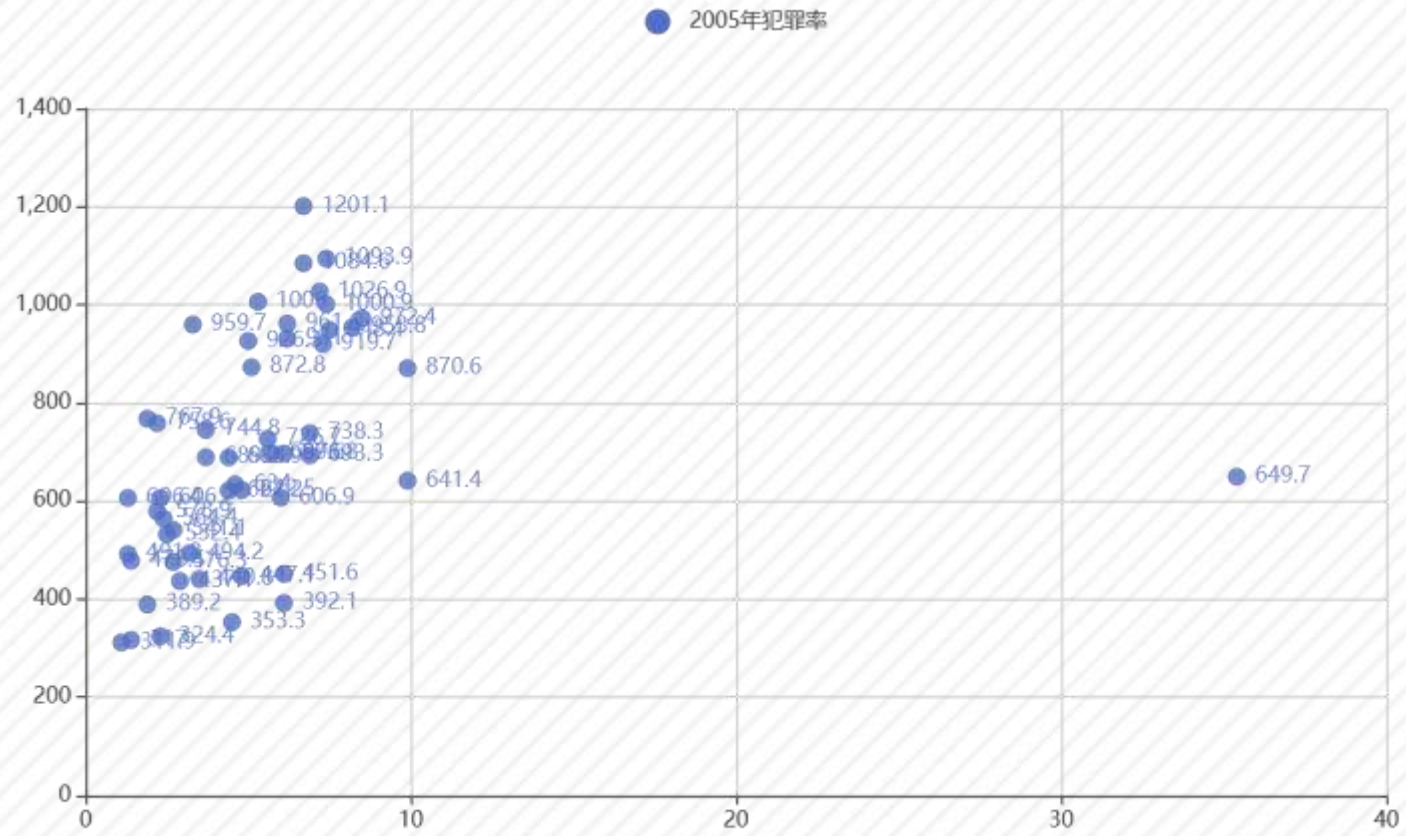
pandas 是基于NumPy 的一种工具，该工具是为了解决数据分析任务而创建的。Pandas 纳入了大量库和一些标准的数据模型，提供了高效地操作大型数据集所需的工具。pandas提供了大量能使我们快速便捷地处理数据的函数和方法。

基于Python的散点图实现

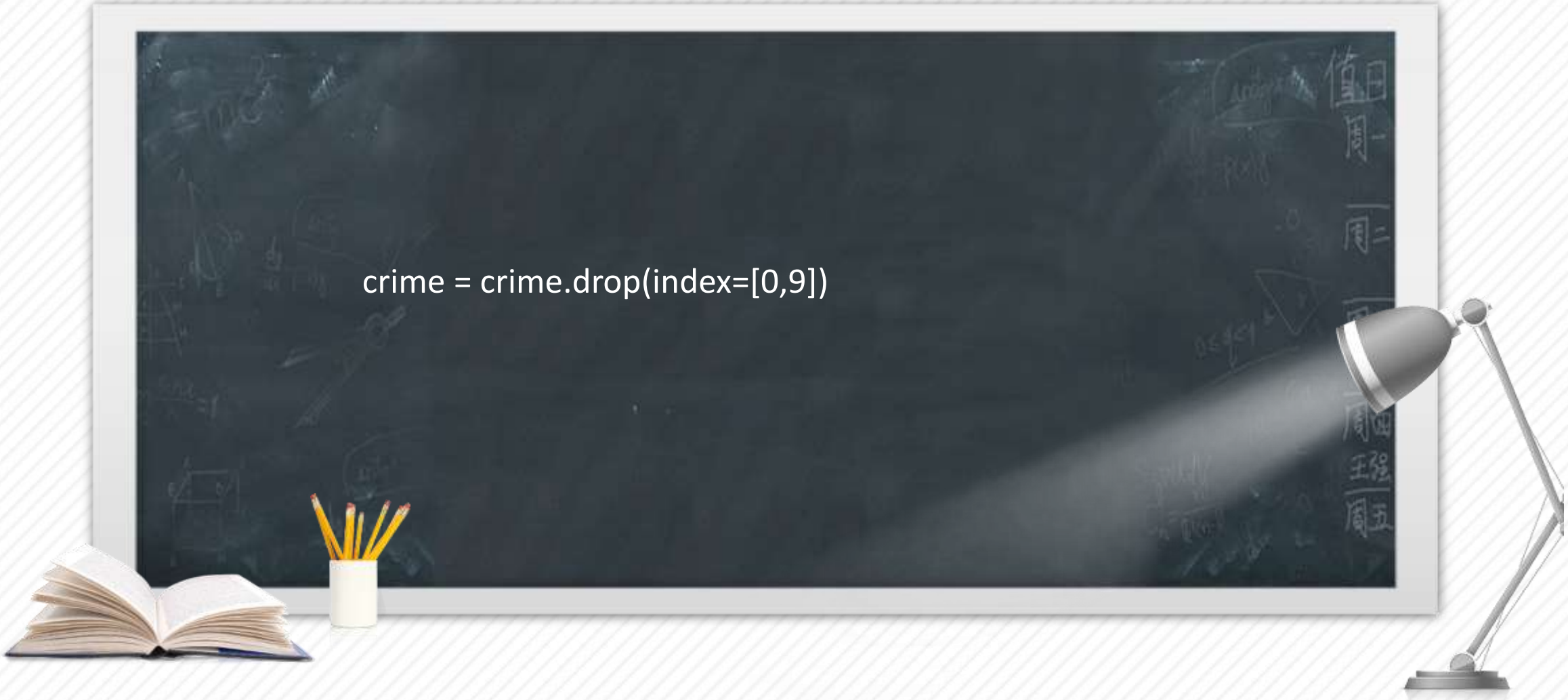
```
from pyecharts.charts import Scatter
import pandas as pd
crime = pd.read_csv("crimeRatesByState2005.csv")
(
    Scatter()
    .add_xaxis(crime.murder)
    .add_yaxis('2005年犯罪率',crime.burglary)
    .render()
)
```

运行截图

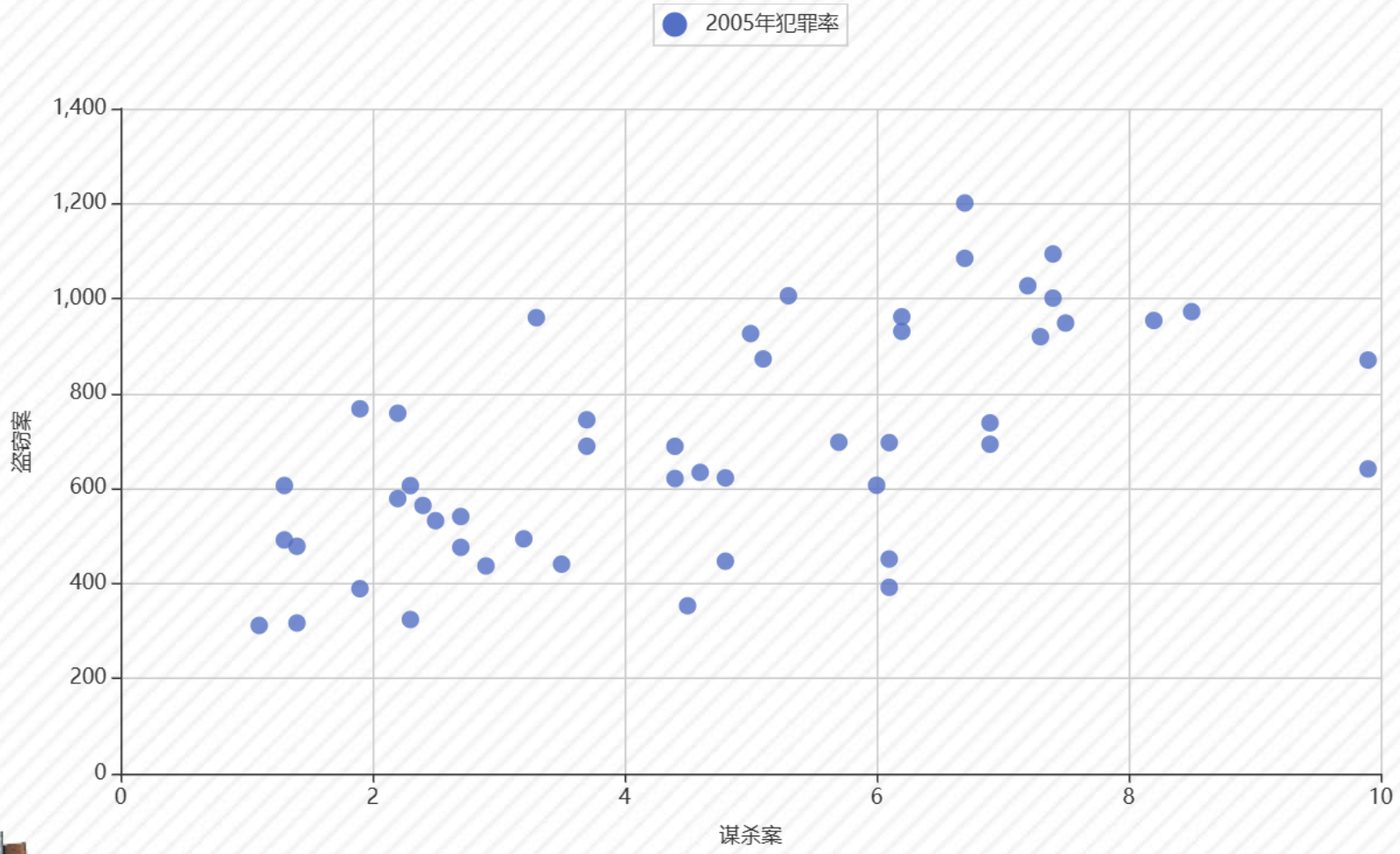
可以看出，谋杀案和入室盗窃案的两组数据似乎呈现一个正相关的关系。但是存在一个离群点，使得横轴延伸很长，整体数据布局不是很清楚，我们可以将其过滤掉。



基于Python的散点图实现



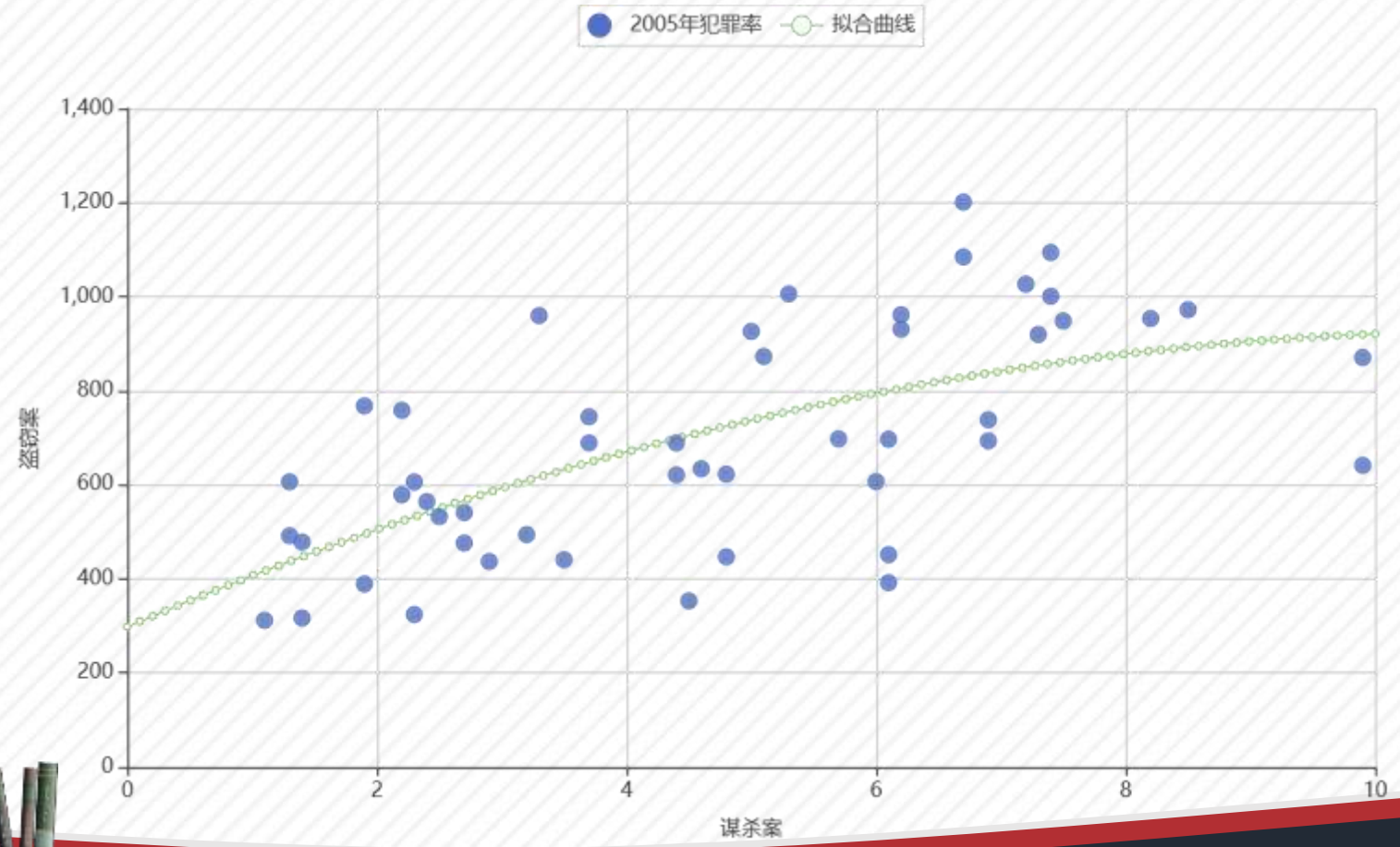
运行截图



拟合曲线？

运行截图

基本呈现的是正相关关系，这里加入了一条拟合曲线让这个图表更加有用，可以更加明确的看出谋杀案和入室抢劫案之间的关系。

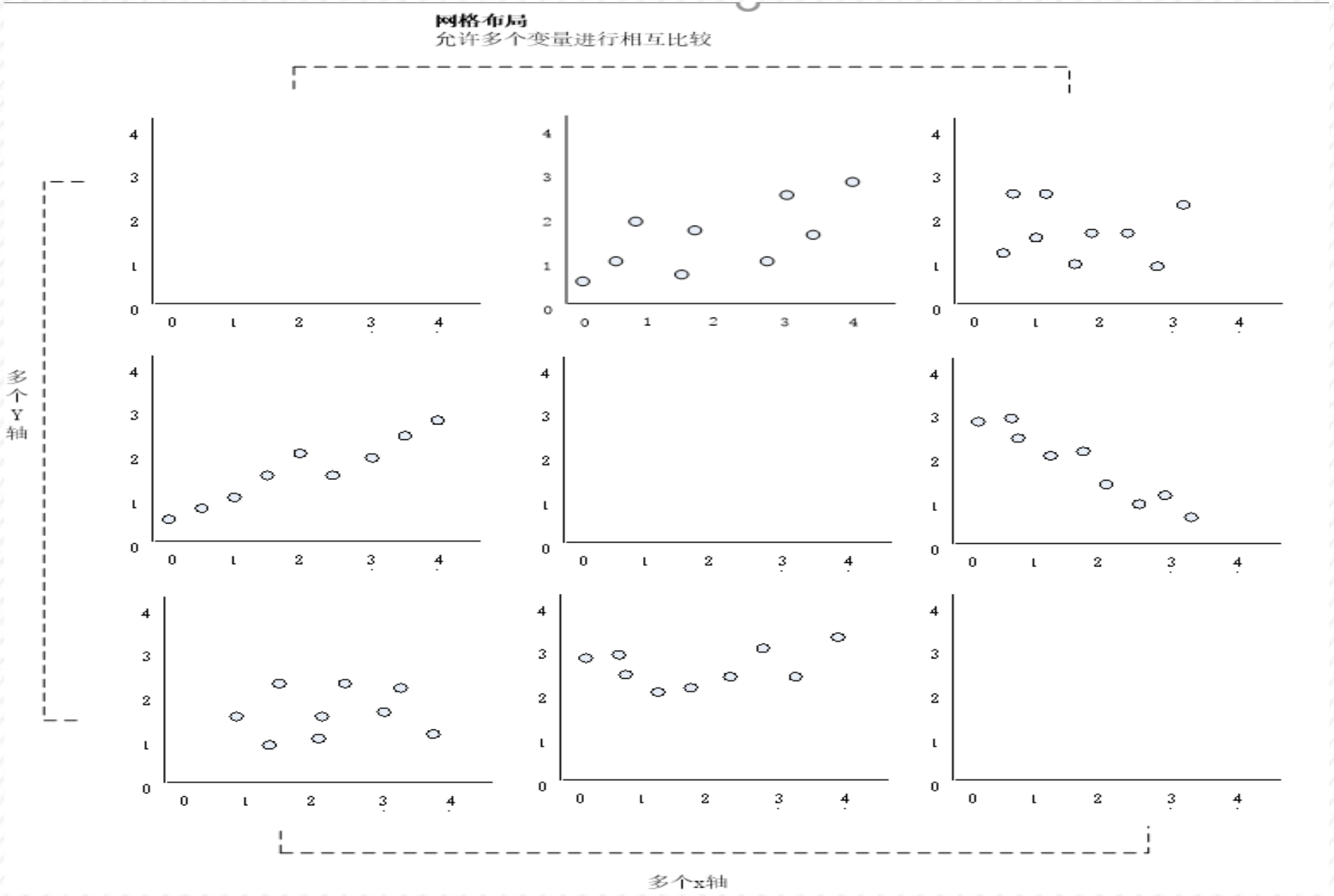


基于Python的散点图及拟合曲线实现

```
from pyecharts.charts import Scatter,Line
import pandas as pd
import numpy as np
from pyecharts import options as opts
crime = pd.read_csv("crimeRatesByState2005.csv")
crime=crime.drop(index=[0,9]).reset_index()
sandian=(
    Scatter()
    .add_xaxis(crime.murder)
    .add_yaxis('2005年犯罪率',crime.burglary)
    .set_series_opts(label_opts=opts.LabelOpts(is_show=False))
    .set_global_opts(xaxis_opts=opts.AxisOpts(name="谋杀案",
                                              name_location='middle',
                                             name_gap=30),
                     yaxis_opts=opts.AxisOpts(name="盗窃案",
                                              name_location='middle',
                                             name_gap=50)
                    )
)
p=np.polyfit(crime.murder,crime.burglary,2)
x=np.linspace(0,10,100)
y=np.polyval(p,x)
nihe=(
    Line()
    .add_xaxis(x)
    .add_yaxis('拟合曲线',y,)
    .set_series_opts(label_opts=opts.LabelOpts(is_show=False))
)
sandian.overlap(nihe).render_notebook()
```

散点矩阵图

散点矩阵图也叫散点图矩阵，它允许同时看到多个单独变量的分布和它们两两之间的关系，是多个散点图的有机结合，其基本框架如图所示。



Seaborn

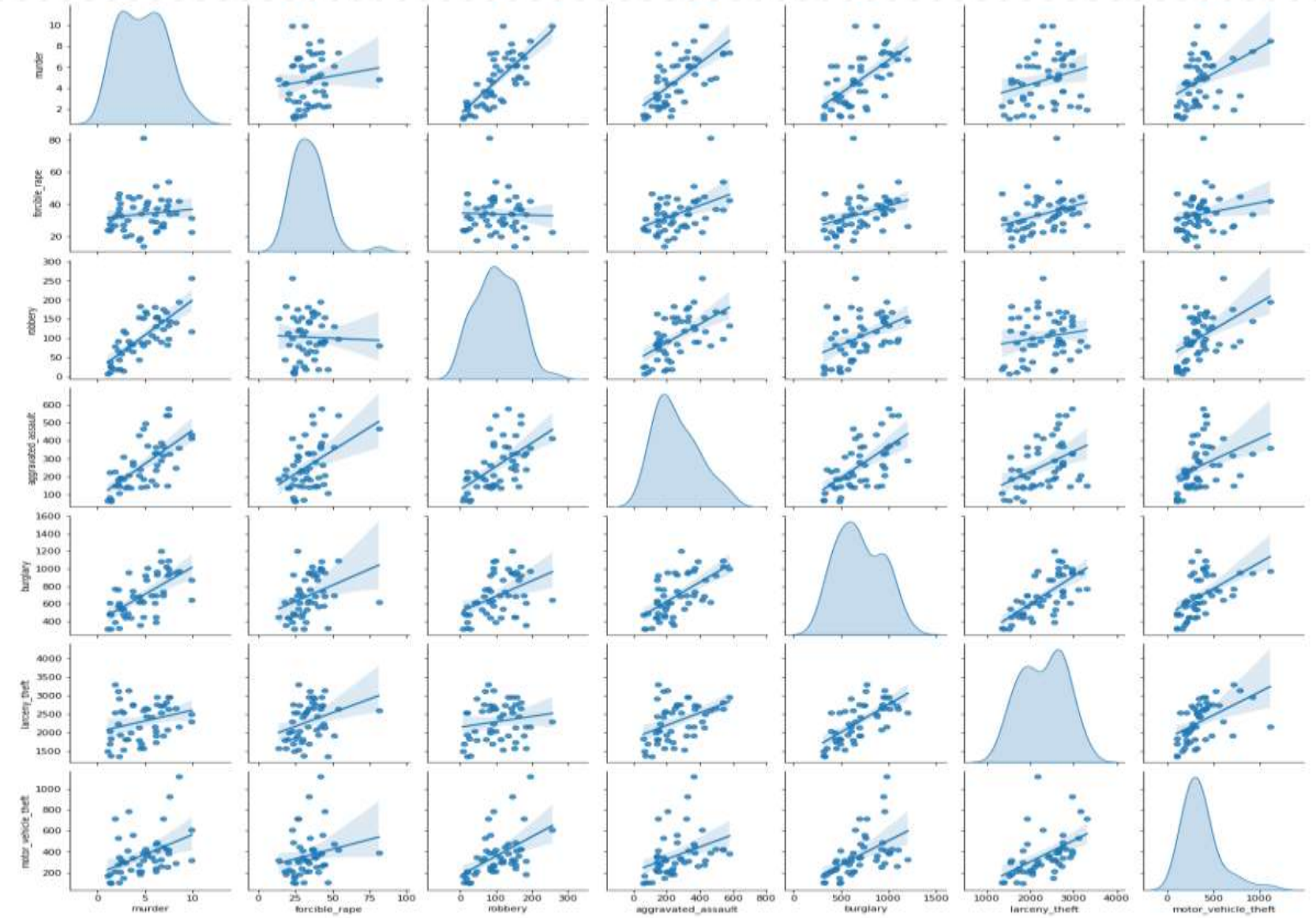
Seaborn是基于matplotlib的图形可视化python包。它提供了一种高度交互式界面，便于用户能够做出各种有吸引力的统计图表。

Seaborn是在matplotlib的基础上进行了更高级的API封装，从而使得作图更加容易，在大多数情况下使用seaborn能做出很具有吸引力的图，而使用matplotlib就能制作具有更多特色的图。应该把Seaborn视为matplotlib的补充，而不是替代物。同时它能高度兼容numpy与pandas数据结构。

基于Python的散点矩阵图实现

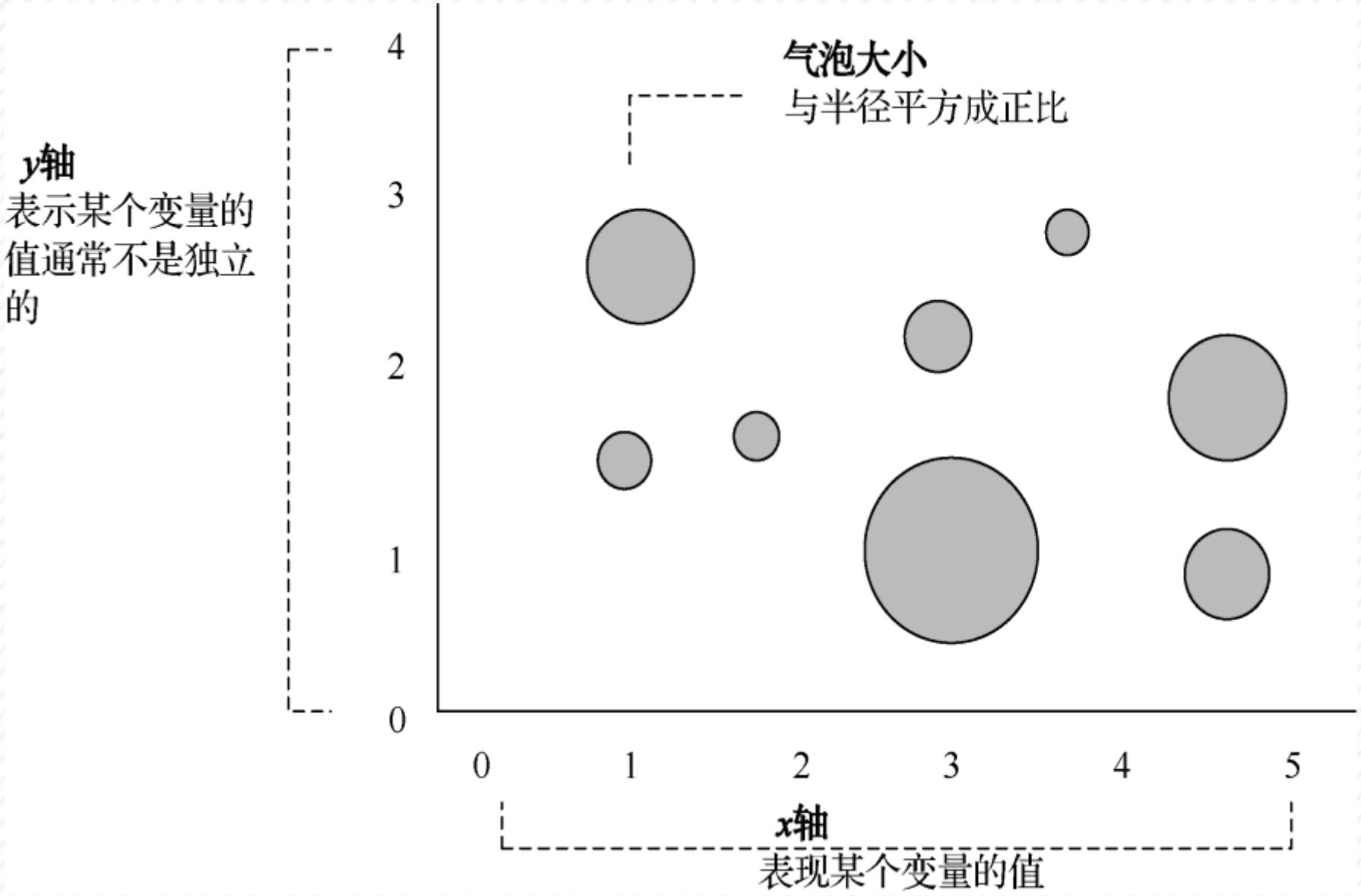
```
1. import pandas as pd
2. import matplotlib.pyplot as plt
3. import seaborn as sns
4. crime = pd.read_csv("crimeRatesByState2005.csv")
5. crime = crime.drop(index=[0,9])
6. crime = crime.drop(['state’, 'population'], axis=1)
7. g = sns.pairplot(crime, diag_kind="kde", kind="reg")
8. plt.show()
```

运行结果截图


murder
forcible_rape
robbery
aggravated_assault
burglary
larceny_theft
motor_vehicle_theft

气泡图

气泡图与散点图相似，不同之处在于，气泡图允许在图表中额外加入一个表示大小的变量。实际上，这就像以二维方式绘制包含三个变量的图表一样。其基本框架如图所示。

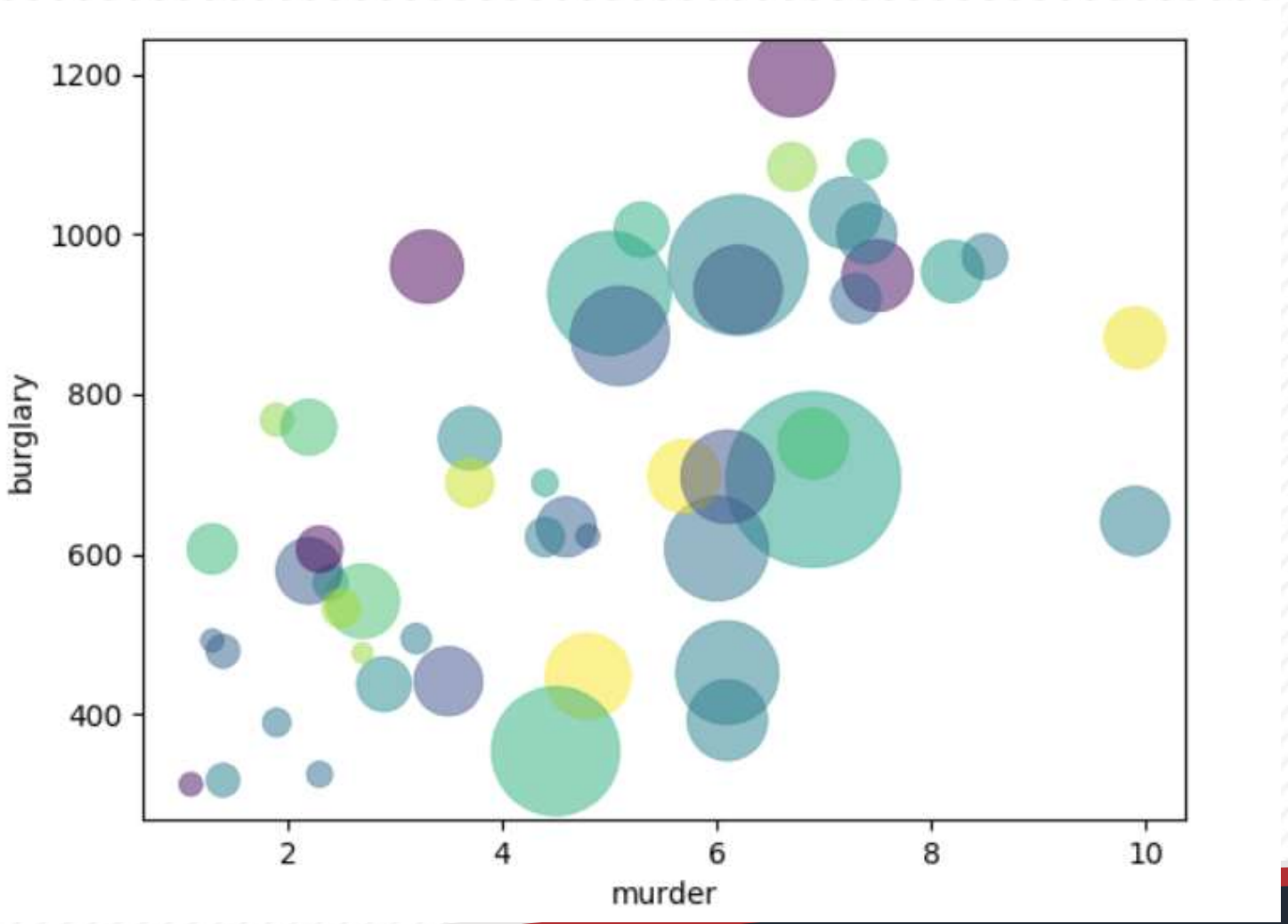


基于Python的气泡图实现

```
1. import matplotlib.pyplot as plt
2. import pandas as pd
3. import numpy as np
4. crime = pd.read_csv("crimeRatesByState2005.csv")
5. crime = crime.drop(index=[0,9])
6. s = list(crime.population/10000)
7. colors = np.random.rand(len(list(crime.murder)))
8. plt.scatter(x=list(crime.murder), y=list(crime.burglary), s=s, c=colors,
linewidth=0.5, alpha=0.5)
9. plt.xlabel("murder")
10. plt.ylabel("burglary")
11. plt.show()
```

运行截图

代码的运行结果如图所示。



颜色可用来表示什么数据？请展示

关系图

Graph 关系图通常用于分析具有复杂关系的数据。例如，如果想探索不同实体之间的关系，可以使用 Graph 关系图来描绘这些关系。

Graph 关系图通常用于社交网络分析、网络安全分析和生物信息学只研究等领域，以及其他任何需要探索复杂关系的应用场景。

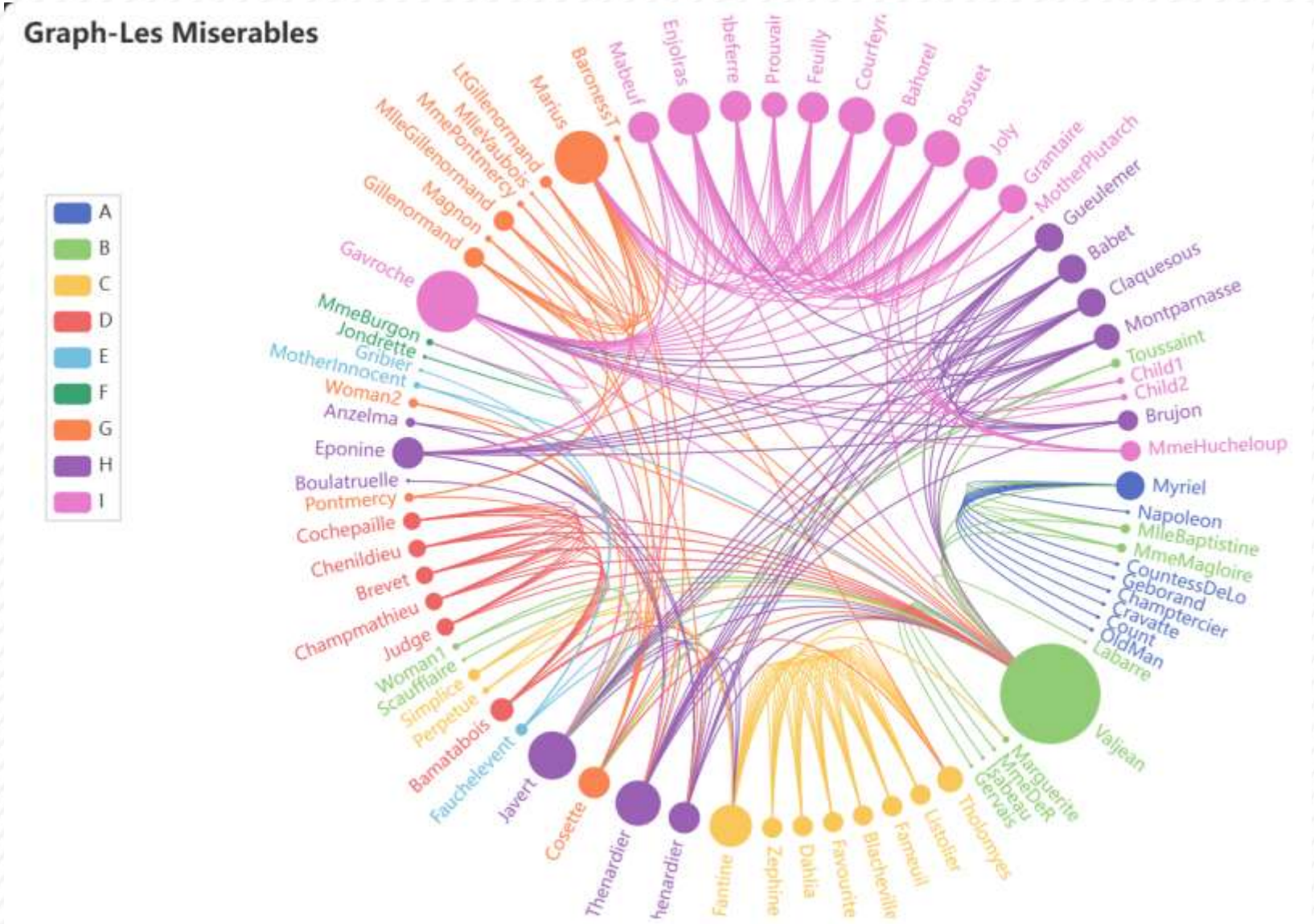
关系图需要的数据包括：GraphNode（节点数据项）、GraphLink（节点间的关系数据）和GraphCategory（节点分类类目）。GraphNode是各个节点基本参数。其中'name'和'value'是默认的浮动显示数据。'x'、'y'是不指定布局方式layout时才必需，'symbolSize'和'category'则可进一步描述节点数据的其它属性，比如用不同颜色来分门别类表示不同的'category'，节点的大小则由'symbolSize ‘决定。GraphLink相对简单，[{'source': x , 'target': y}]描述关系数据的起止节点。GraphCategory是节点类别的列表[{'name': '类目0'}]，也可指定各类节点的图标、尺寸等。

以《悲惨世界》中的人物关系为例，绘制关系图。

关系图

```python
# 导入所需的Python库
import json
from pyecharts import options as opts
from pyecharts.charts import Graph

# 从JSON文件中读取数据
with open("les-miserables.json", "r", encoding="utf-8") as f:
    j = json.load(f)
    nodes = j["nodes"]  # 获取节点数据
    links = j["links"]  # 获取链接数据
    categories = j["categories"]  # 获取节点分类数据

# 使用Pyecharts库创建Graph图
c = (
    Graph(init_opts=opts.InitOpts(width="1000px", height="600px"))  # 设置图的初始参数，包括宽度和高度
    .add(
        "",  # 添加图的系列名称（此处为空，表示使用默认名称）
        nodes=nodes,  # 设置节点数据
        links=links,  # 设置链接数据
        categories=categories,  # 设置节点分类数据
        layout="circular",  # 设置图的布局方式为circular（环形）
        is_rotate_label=True,  # 是否旋转标签
        linestyle_opts=opts.LineStyleOpts(color="source", curve=0.3),  # 设置线的样式
        label_opts=opts.LabelOpts(position="right"),  # 设置标签的位置
    )
    .set_global_opts(
        title_opts=opts.TitleOpts(title="Graph-Les Miserables"),  # 设置图的标题
        legend_opts=opts.LegendOpts(orient="vertical", pos_left="2%", pos_top="20%"),  # 设置图例的位置
    )
    .render("graph_les_miserables.html")  # 渲染图，并保存为HTML文件
)
```

关系图


Graph-Les Miserables
A
B
C
D
E
F
G
H
I
Napoleon
MlleBaptistine
MmeMagloire
CountessDeLo
Geborand
Champtercier
Cravatte
Count
OldMan
Labarre
Valjean
Marguerite
MmeDeR
Isabeau
Gervais
Thoiomyes
Listolier
Fameuil
Blacheville
Favourite
Dahlia
Zephine
Fantine
Cosette
Thenardier
henardier
Javert
Fauchelevent
Bamatabois
Perpetue
Simplice
Scaufflaire
Woman1
Judge
Champmathieu
Brevet
Chenildieu
Cochepaille
Pontmercy
Boulatruelle
Eponine
Anzelma
Woman2
MotherInnocent
Gribier
Jondrette
MmeBurgon
Gavroche
Gillenormand
Magnon
MlleGillenormand
MmePontmercy
MlleVaubois
LtGillenormand
Marius
BaronessT
Mabeuf
Enjolras
Prouvaire
Feuilly
Courfeyr.
Bahorel
Bossuet
Joly
Grantaire
MotherPlutarch
Gueulemer
Babet
Claquesous
Montparnasse
Toussaint
Child1
Child2
Brujon
MmeHucheloup

数据的分布性

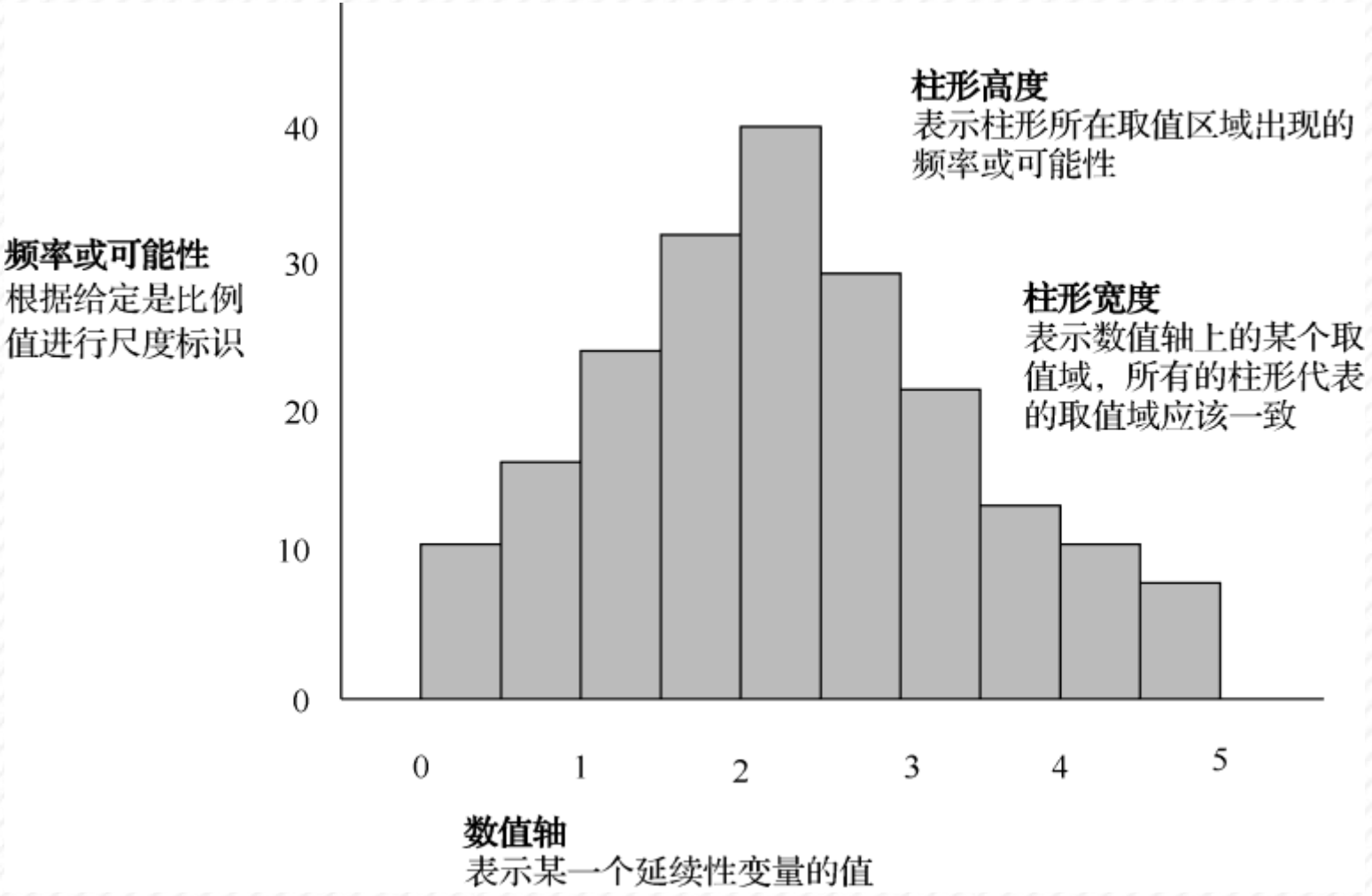
数据的分布性

反映一组数据的平均水平与波动大小的数字特征，可以用平均数、方差等， 它们从某一项侧面反映了一组数据的情况，但是在实际生活中，有时只知道这些情况还不够，还需要知道数据在整体上的分布情况。这就是我们研究数据分布性的目的所在。

数据分布（频率分布），是指在统计分组的基础上，将总体中各单位按组归类整理，按一定顺序排列，形成的总体中各单位在各组间的分布。其实质是，在各组按顺序排列的基础上，列出每个组的总体单位数，形成一个数列，称为次数分布数列，简称分配数列，各组的总体单位数叫次数或频数。

直方图

直方图，又称质量分布图，是一种统计报告图，由一系列高度不等的纵向条纹或线段表示数据分布的情况。一般用横轴表示数据类型，纵轴表示分布情况。直方图的基本架构如图所示。

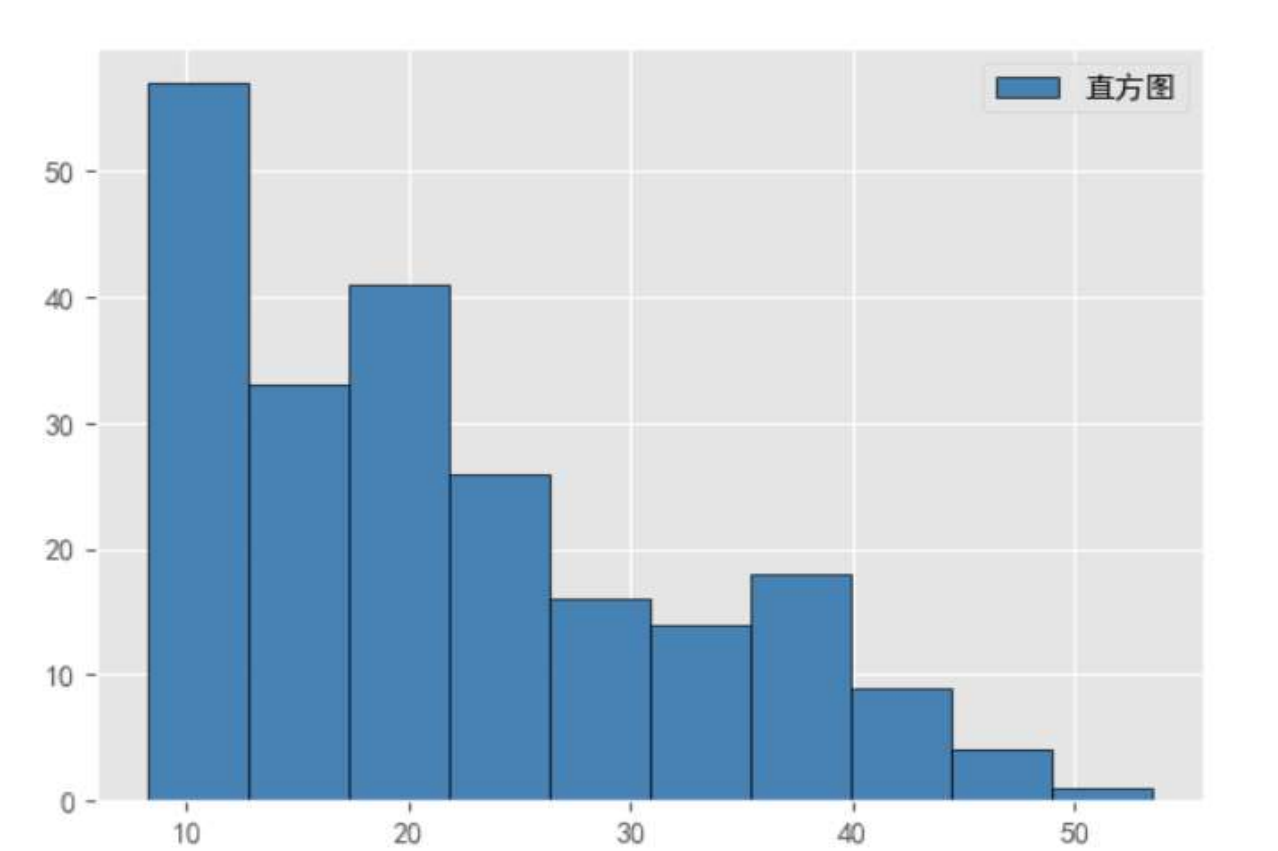


基于Python的直方图实现

```
1. import pandas as pd
2. import matplotlib.pyplot as plt
3. plt.rcParams['font.sans-serif'] = ['SimHei']  # 解决中文不能正常显示的问题
4. titanic = pd.read_csv('birth-rate.csv’)
5. titanic.dropna(subset=['2008'], inplace=True)
6. plt.hist(titanic['2008'], bins=10, color='steelblue', edgecolor='k', label="直方图")
7. plt.tick_params(top='off', right='off’)
8. plt.legend()
9. plt.show()
```

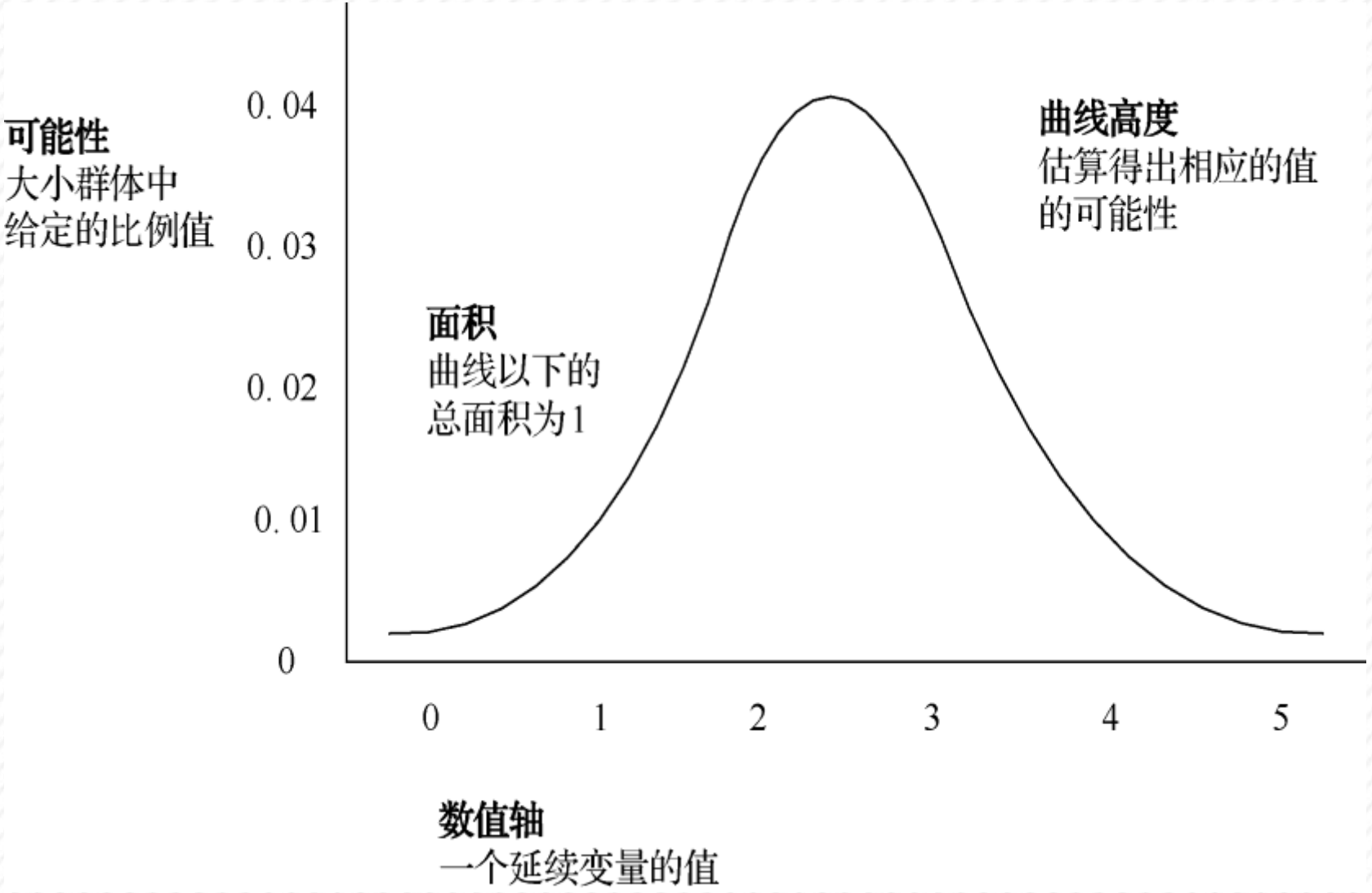
运行截图

上述代码呈现出的效果如图所示。



密度图

密度图表现与数据值对应的边界或域对象的一种理论图形表示方法。一般用于呈现连续变量。密度图的基本架构如图所示。

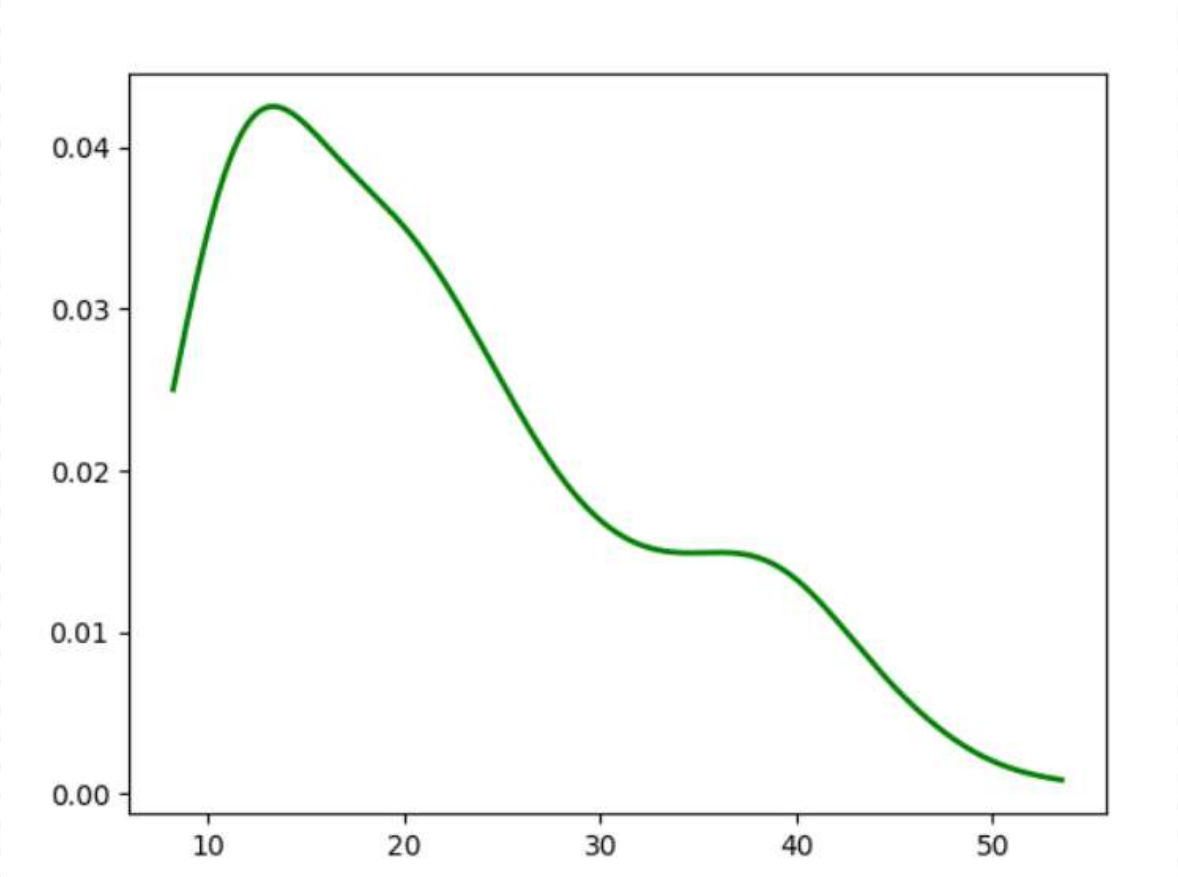


基于Python的密度图实现

```python
# 导入所需的Python库
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.mlab as mlab

# 设置中文字体和正负号显示
plt.rcParams['font.sans-serif'] = ['Microsoft YaHei']
plt.rcParams['axes.unicode_minus'] = False

# 读取CSV文件数据
titanic = pd.read_csv('birthrate.csv')

# 删除缺失值所在的行（subset参数指定了只考虑“2008”列的缺失值）
titanic.dropna(subset=['2008'], inplace=True)

# 使用高斯核密度估计函数，对“2008”列数据进行拟合
kde = mlab.GaussianKDE(titanic['2008'])

# 生成一组等间距的数值，用于绘制拟合曲线
x2 = np.linspace(titanic['2008'].min(), titanic['2008'].max(), 1000)

# 绘制拟合曲线，并设置曲线颜色和线宽
line2 = plt.plot(x2, kde(x2), 'g-', linewidth=2)

# 显示图形
plt.show()
```

运行截图

上述代码呈现出的效果如图所示。

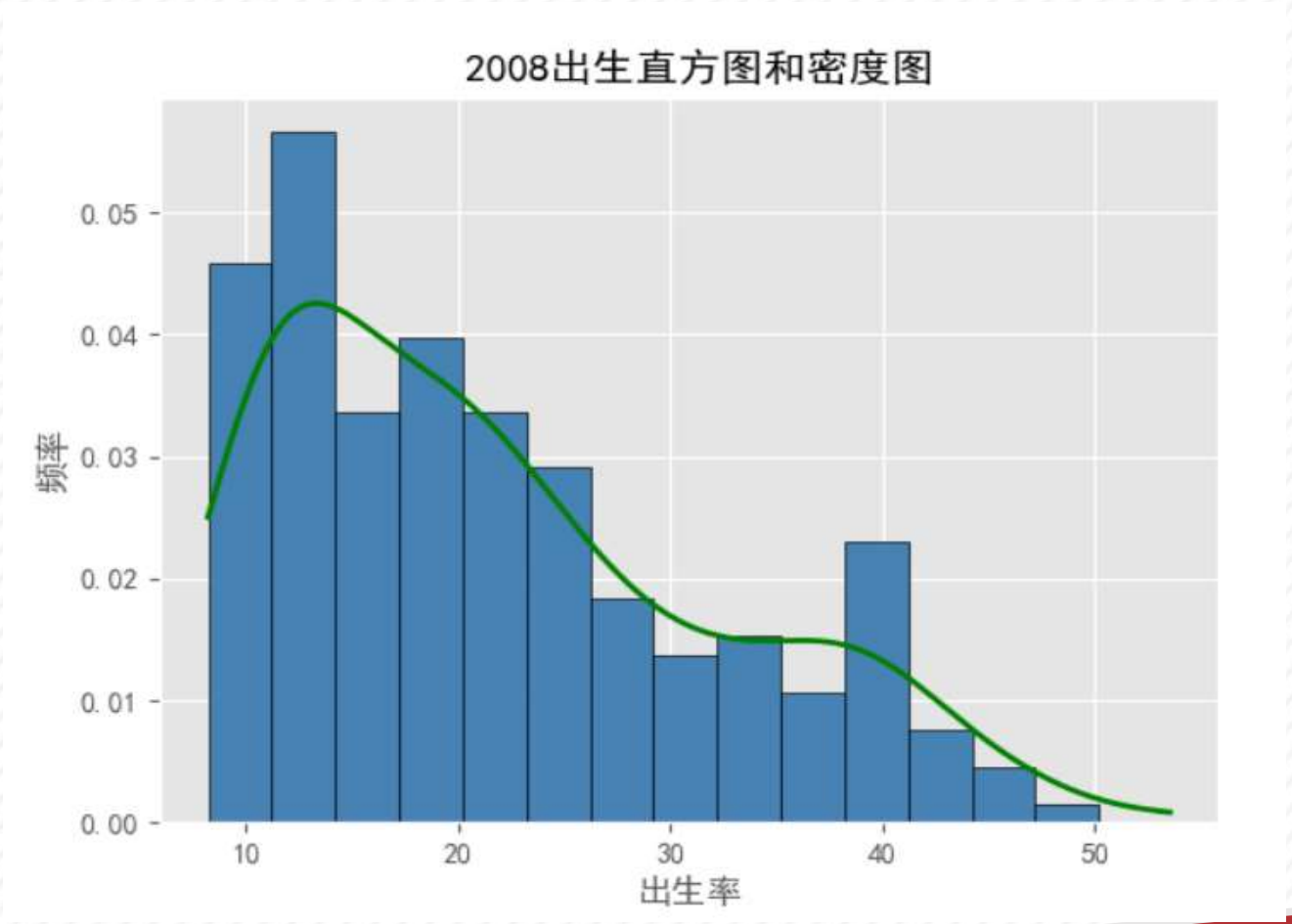


基于Python的直方图结合密度图的实现

```
# 导入所需的Python库
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.mlab as mlab

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

# 读取CSV文件数据
titanic = pd.read_csv('birthrate.csv')

# 删除缺失值所在的行（subset参数指定了只考虑“2008”列的缺失值）
titanic.dropna(subset=['2008'], inplace=True)

# 绘制直方图，设置组距、密度、颜色和边框颜色等参数，并添加标题、横轴标签和纵轴标签
plt.hist(titanic['2008'], bins=np.arange(titanic['2008'].min(), titanic['2008'].max(), 3),
         density=True, color='steelblue', edgecolor='k')
plt.title('2008出生直方图和密度图')
plt.xlabel('出生率')
plt.ylabel('频率')

# 使用高斯核密度估计函数，对“2008”列数据进行拟合，并生成一组等间距的数值，用于绘制拟合曲线
kde = mlab.GaussianKDE(titanic['2008'])
x2 = np.linspace(titanic['2008'].min(), titanic['2008'].max(), 1000)

# 绘制拟合曲线，设置曲线颜色和线宽
line2 = plt.plot(x2, kde(x2), 'g-', linewidth=2)

# 设置图形的边框线条是否显示
plt.tick_params(top='off', right='off')

# 显示图形
plt.show()
```

运行截图

上述代码呈现出的效果如图所示。



本章小结

- 关系数据在大数据中的应用
- 数据关联性的特性，以及图表表示法（散点图，散点图矩阵，气泡图，关系图）
- 数据分布性的特性，以及图表表示法（直方图，密度分布图）
- 各种可视化图表的python实现