5.9. Python编程实践

▲5.8.数据形态及其规整化方法

▼5.10.继续学习本章知识

Python编程实践

【分析对象】

Excel 文件—文件名为“deaths.xlsx”和“icd-main.xlsx”。读者可以在本书配套资源中找到上述两个数据集。

“deaths.xlsx”来自墨西哥的个体死亡数据库数据集，主要记录的是个体的死亡数据，包含带有5 个属性的539 530 个样本数据。其属性含义如下。

yod:个体的死亡年份。 mod:个体的死亡月份。 dod:个体的死亡日期。 hod:个体的死亡时间(小时)。 cod:个体的死亡原因编号。

“icd-main.xlsx”数据集主要记录的是死亡原因数据，包含带有 2 个属性的 1858 个样本数据，其属性含义如下。

code:死亡原因编号。 disease:具体死亡原因。

【分析目的及任务】

理解数据规整工具在数据科学中的应用，以进行数据加工。 首先，数据读入并对其进行条件过滤、缺失值处理、JOIN 操作、分组统计和数据 排序等预处理操作 其次，通过可视化来理解每小时死亡人数变化情况 。接着，对每种病的死亡人数和每种病每小时的死亡比例与每小时的总死亡比例间的距离进行线性回归分析，并绘制拟合曲线。最后，根据残差筛选非寻常模式死亡数据，以总死亡人数350为边界划分非寻常模式死亡数据集，分别绘制死亡时间图并查看死亡原因 。

【分析方法及工具】 Python。

【主要步骤】

数据加工的步骤主要包括数据读入、数据预处理(条件过滤、缺失值处理、JOIN 操 作、分组统计和数据排序)、数据可视化及数据建模。

Step 1:数据读入

通常，我们采用 Pandas 包提供的 read_excel()函数将当前工作目录下的两个 Excel 文件—“deaths.xlsx”和 “icd-main.xlsx”分别读入至 Pandas 数据框 deaths 和 code 中。本书第2章中已详细介绍过当前工作目录的查看和设置方法，以及基于Pandas 包读取数据文件的方法，在此不再赘述。数据读入示例如下。

In[1]
```
#导入 Pandas 包和 Numpy 包
import pandas as pd
import numpy as np

#读入数据文件 deaths.xlsx 至 Pandas 对象 deaths deaths = pd.read_excel('./dataset/deaths.xlsx‘)
#读入数据文件 icd-main.xlsx 至 Pandas 对象 code
 code = pd.read_excel('./dataset/icd-main.xlsx‘)
#显示 Pandas 对象 deaths 和 code 的列名 print(deaths.columns,'\n',code.columns)
#【提示】读取个体死亡数据与死亡原因数据，并查看两个数据集的列名
```

对应输出结果为：

```
Index(['yod', 'mod', 'dod', 'hod', 'cod'], dtype='object')
Index(['code', 'disease'], dtype='object')
```

Step 2:数据预处理

（1）条件过滤

（2）缺失值处理

（3）分组统计

（4）JOIN 操作

（5）数据排序

（1）条件过滤

在此，我们将先从数据集deaths中筛选出2008年的个体死亡数据，并查看所得到的数据集形状。示例如下。

```
In[2] #进行条件过滤
mexico_deaths_2008 = deaths[(deaths[‘yod’]== 2008) & (deaths[‘mod’]!= 0)&
(deaths['dod']!=0)].dropna()
#【提示】筛选条件是“年份”为 2008，且“月份”和“日期”不为 0
#查看条件过滤后的数据形状 print(mexico_deaths_2008.shape)
```

对应输出结果为：

```
(502520, 5)
```

（2）缺失值处理

本章的“5.4 缺失数据及其处理”中介绍了缺失值的常用处理方法。在此，我们采用Pandas 提供的dropna()函数直接删除数据框deatshs 中的含有缺失数据的行。

```
In[3] #删除缺失数据
      mexico_deaths = deaths.dropna()
      #查看缺失数据处理后的数据框 mexico_deaths 的前 10 行 mexico_deaths.head(10)
      【提示】已进行缺失数据的删除，如第 35 行数据已被删除
```

对应输出结果为

	yod	mod	dod	hod	cod
34	1920	11	17	3.0	W78
36	1923	2	4	16.0	J44
37	1923	6	23	19.0	E12
38	1926	2	5	16.0	C67
39	1926	4	1	16.0	J44
40	1928	10	30	19.0	I27
41	1929	4	23	15.0	I25
42	1930	9	11	19.0	E14
43	1930	12	22	19.0	E11

（3）分组统计

分组统计是数据加工中常用的方法之一。在数据科学中通常采用 Pandas 的 groupby()函数进行分组统计。本例中，我们将按死亡时间和死亡原因对数据 框 mexico_ deaths 中的“死亡人数”进行分组统计，并在原数据框 mexico_deaths 中掺 入一个新列—freq，用于记录分组统计后的结果。示例如下。

```
In[4] | m_d = mexico_deaths.groupby([‘hod’,‘cod’]).size().reset_index().
      | rename(columns={0:'freq','cod':'code'}).dropna()
      | #【提示】此处，将原“cod”列重命名为“code”
```

（4）JOIN 操作

与关系数据库中的关系表的JOIN 操作相似，Pandas 的数据框也可以进行连接操作，常用方法为调用pandas 提供的merge()函数。示例如下。

```
In[5] #将整理后的个体死亡数据集与死亡原因数据集连接操作
      m_d = pd.merge(left=m_d, right=code, on='code') m_d[:10]
```

对应输出结果为

	hod	code	freq	disease			
0	1.0	A01	3	Typhoid	and	paratyphoid	fevers
1	2.0	A01	1	Typhoid	and	paratyphoid	fevers
2	3.0	A01	4	Typhoid	and	paratyphoid	fevers
3	5.0	A01	5	Typhoid	and	paratyphoid	fevers
4	6.0	A01	1	Typhoid	and	paratyphoid	fevers
5	8.0	A01	1	Typhoid	and	paratyphoid	fevers
6	10.0	A01	2	Typhoid	and	paratyphoid	fevers
7	11.0	A01	2	Typhoid	and	paratyphoid	fevers
8	12.0	A01	1	Typhoid	and	paratyphoid	fevers
9	13.0	A01	6	Typhoid	and	paratyphoid	fevers

（4）JOIN 操作

其次，计算不同原因导致的死亡人数及不同死亡时间的人数，示例如下。

```
In[6]: # 计算不同原因导致的死亡人数
       code_sum = mexico_deaths['cod'].dropna().value_counts()
       # 计算不同死亡时间的人数
       hour_sum = mexico_deaths['hod'].dropna().value_counts()
```

再次，查看 code_sum 的值，即不同原因导致的死亡人数最高的前 10 项。

```
In[7]: #
       code_sum[:10]
```

对应输出结果为：

```
I21     46794
E11     42421
E14     27330
J44     16043
K70     12860
J18     12516
K74     12486
I25     10059
I50      8486
X59      8188
Name: cod, dtype: int64
```

（4）JOIN 操作

最后，显示 hour_sum 的值，即不同死亡时间的人数。示例如下。

```
In[8]|#查看不同死亡时间的人数
     |hour_sum
```

对应输出结果为：

```
18.0   24380
10.0   24321
16.0   23890
11.0   23843
6.0    23787
17.0   23625
12.0   23392
13.0   23284
15.0   23278
14.0   23053
20.0   22926
19.0   22919
9.0    22401
5.0    22126
8.0    21915
7.0    21822
23.0   21446
21.0   20995
22.0   20510
1.0    20430
4.0    20239
3.0    19729
2.0    18962
Name: hod, dtype:
int64
```

（5）数据排序

在分组统计的基础上，我们还可以进行更多的数据分析，如分类分析和数据排序。在此，我们调用 Pandas 提供的 apply()函数计算每种死因每

对应输出结果为

	hod	code	freq	disease	n	prop	freq_all	prop_all	dist
0	1.0	A01	3	Typhoidand paratyphoid fevers	52	0.057692	20430	0.039803	0.000320
1	2.0	A01	1	Typhoidand paratyphoid fevers	52	0.019231	18962	0.036943	0.000314
2	3.0	A01	4	Typhoidand paratyphoid fevers	52	0.076923	19729	0.038438	0.001481
3	5.0	A01	5	Typhoidand paratyphoid fevers	52	0.096154	22126	0.043108	0.002814
4	6.0	A01	1	Typhoidand paratyphoid fevers	52	0.019231	23787	0.046344	0.000735
5	8.0	A01	1	Typhoidand paratyphoid fevers	52	0.019231	21915	0.042697	0.000551
6	10.0	A01	2	Typhoidand paratyphoid fevers	52	0.038462	24321	0.047384	0.000080
7	11.0	A01	2	Typhoidand paratyphoid fevers	52	0.038462	23843	0.046453	0.000064
8	12.0	A01	1	Typhoidand paratyphoid fevers	52	0.019231	23392	0.045574	0.000694
9	13.0	A01	6	Typhoidand paratyphoid fevers	52	0.115385	23284	0.045364	0.004903

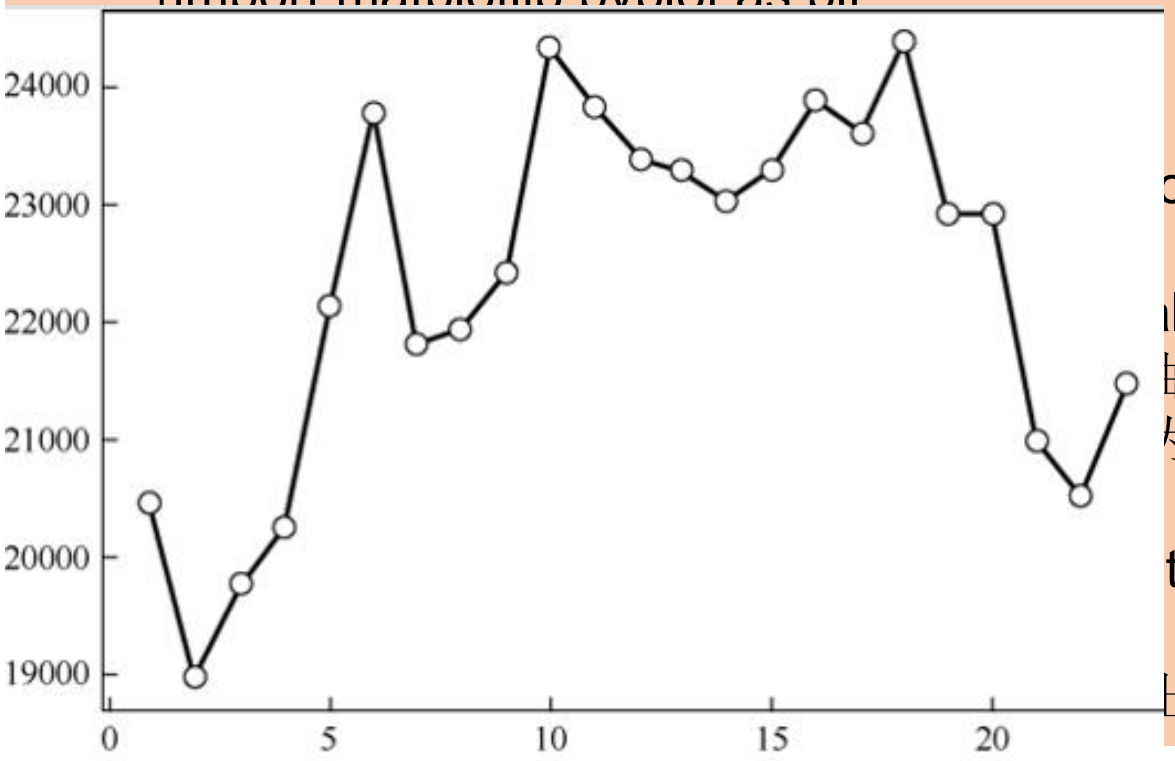
a row:

In

Step 3:数据可视化

在数据加工的基础上，我们可以调用Python第三方数据可视化包进行数据可视化。

对应输出结果：




```
#导入 matplotlib 包
import matplotlib.pyplot as plt
Text(0.5,1,'每小时死亡人数变化折线图')
ort_values('hod')
ll'], marker='o', mec= 'r', mfc='w')
轴数据、y 轴数据、marker 表示节
为红色，mfc 设置标记填充颜色为白色
t.rcParams['axes.unicode_minus'] =False
出英文，如果输出中文，要对字体进行调整
```

Step 4:数据建模

在数据加工的基础上，我们还可以采用统计学或机器学习的方法对数据进行建模。关于统计学模型和机器学学习算法的讨论，建议参考第2章和第3章的内容。当然，在机器学习和统计建模过程中，也需要进一步进行数据加工和数据预处理。

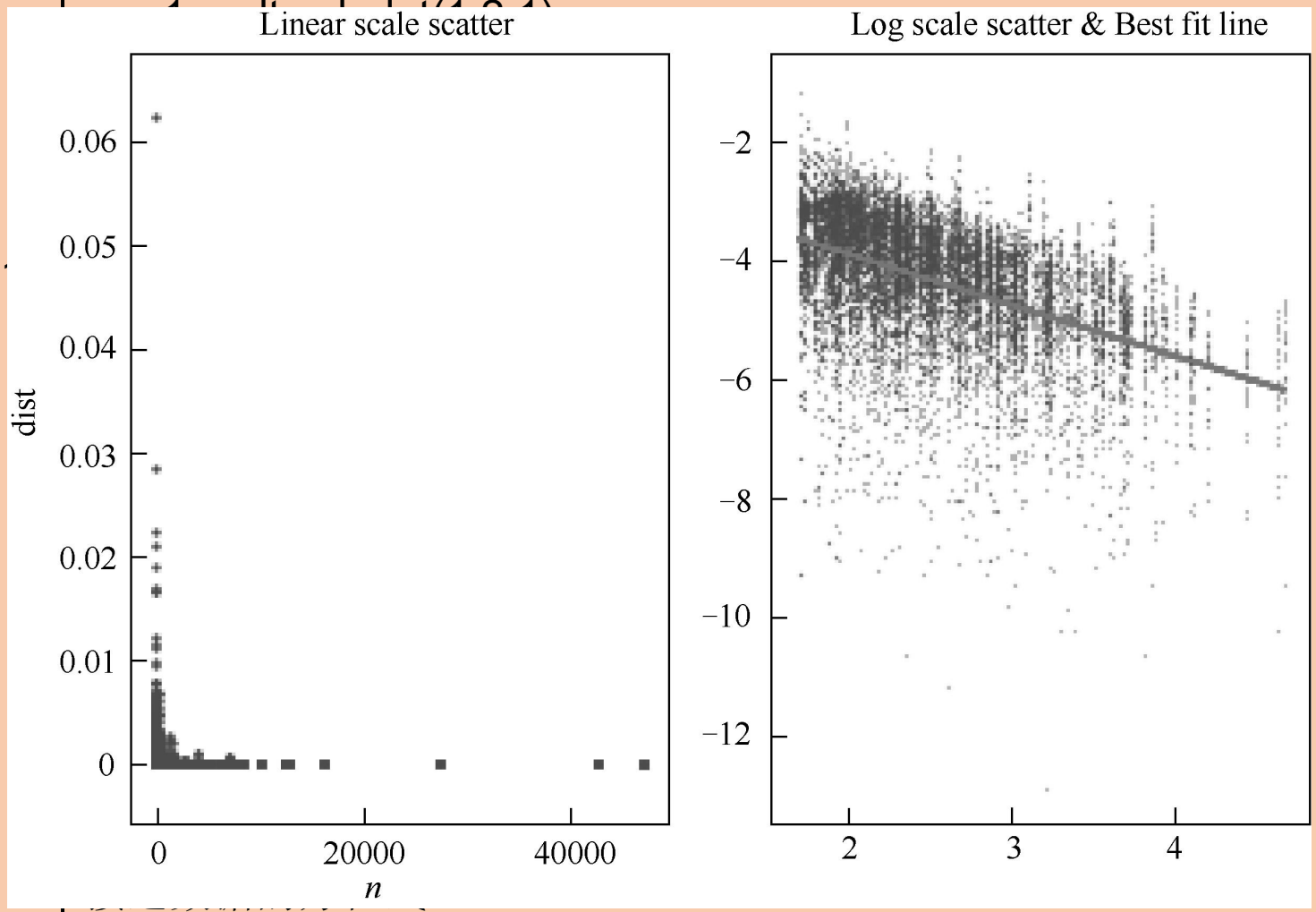
```
In[11]|#根据死因筛选出总死亡人数大于 50 的数据
      |m_d_50 = m_d[m_d['n'] > 50] m_d_50[:10]
```

对应输出结果为

	hod	code	freq	disease	n	prop	freq_all	prop_all	dist
0	1.0	A01	3	Typhoidand paratyphoid fevers	52	0.057692	20430	0.039803	0.000320
1	2.0	A01	1	Typhoidand paratyphoid fevers	52	0.019231	18962	0.036943	0.000314
2	3.0	A01	4	Typhoidand paratyphoid fevers	52	0.076923	19729	0.038438	0.001481
3	5.0	A01	5	Typhoidand paratyphoid fevers	52	0.096154	22126	0.043108	0.002814
4	6.0	A01	1	Typhoidand paratyphoid fevers	52	0.019231	23787	0.046344	0.000735
5	8.0	A01	1	Typhoidand paratyphoid fevers	52	0.019231	21915	0.042697	0.000551
6	10.0	A01	2	Typhoidand paratyphoid fevers	52	0.038462	24321	0.047384	0.000080
7	11.0	A01	2	Typhoidand paratyphoid fevers	52	0.038462	23843	0.046453	0.000064
8	12.0	A01	1	Typhoidand paratyphoid fevers	52	0.019231	23392	0.045574	0.000694
9	13.0	A01	6	Typhoidand paratyphoid fevers	52	0.115385	23284	0.045364	0.004903

（1）绘制线性空间下散点图

对应输出结果为：

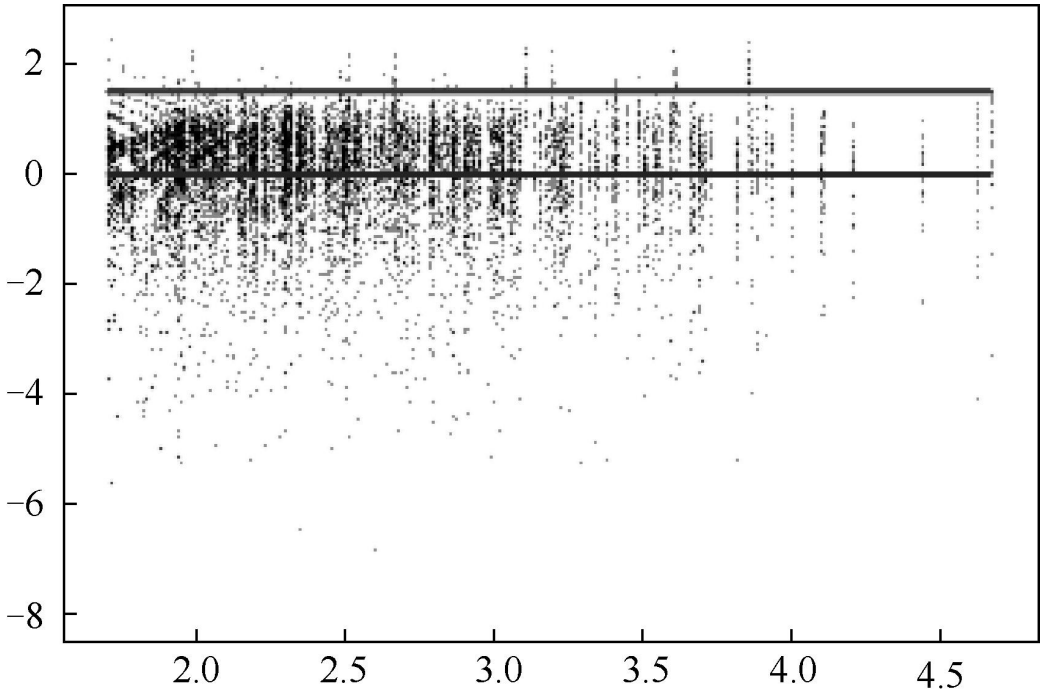


（2）绘制log空间下的线性预测函数残差散点图

```
In[13]
y = np.poly1d(parameter)
#【提示】利用 np.poly1d()可以获得一元多次方程式
#计算残差
m_d_50[‘resid’] = m_d.apply(lambda row:
np.log(row['dist'])-y (np.log(row['n'])), axis=1)
#【提示】用真实值减去预测值计算残差 resid
#绘制残差散点图
plt.scatter(x_log, y_log-y(x_log), s=0.01, c='b', marker='o‘)
#计算残差平均值
mean_resid = 0*x_log + (y_log-y(x_log)).mean()
 #1.5 作为正常死亡与非正常死亡的边界
 unusual_line = 0*x_log + 1.5
 #绘制残差平均值曲线
 plt.plot(x_log, mean_resid)
 #绘制异常死亡边界线
 plt.plot(x_log, unusual_line, c='r')
```

对应输出结果为：

[<matplotlib.lines.Line2D at 0x22d9c3ae710>]



（3）将与预测值偏差超过1.5的标记为非寻常模式死亡

In[14]:
```
m_d_unusual = m_d_50[m_d_50['resid']>1.5]
#【提示】筛选出偏差超过 1.5 的非寻常模式死亡数据
print(m_d_unusual['code'].unique())
#【提示】查看导致非寻常死亡的死因编码
```

对应输出结果为：

```
['R95' 'V09' 'V89' 'W69' 'W73' 'W74' 'X95' 'X97' 'X99' 'Y21' 'V79' 'X33' 'V95']
```

（4）以总死亡人数 350 为边界，划分非寻常模式死亡疾病数据

```
In[15]| m_d_unusual_bigger = m_d_unusual[m_d_unusual['n']>350]
      | #【提示】筛选出总死亡人数超过 350 的非寻常模式死亡数据
      | m_d_unusual_smaller = m_d_unusual[m_d_unusual['n']<=350]
      | #【提示】筛选出总死亡人数不超过 350 的非寻常模式死亡数据
```

（5）查看总死亡人数超过350的非寻常死亡疾病

```
In[16]: m_d_unusual_bigger['code'].unique()
```

对应输出结果为：

```
array(['V09', 'V89', 'W69', 'W74', 'X95', 'X99'], dtype=object)
```

（6）绘制总死亡人数超过350的非寻常死亡模式死亡时间图

V09 : Pedestrian injured in other and unspecified transport accidents
V89 : Motor- or nonmotor-vehicle accident, type of vehicle unspecified
W67 : Drowning and submersion while in natural water
W74 : Unspecified drowning and submersion
X95 : Assault (homicide) by other and unspecified firearm discharge
X99 : Assault (homicide) by sharp object

```
In[17]: fig = plt.figure()
        for index,cod in enumerate(m_d_unusual_bigger['code'].unique()):
            dise = m_d_50[m_d_50['code']==cod]
```

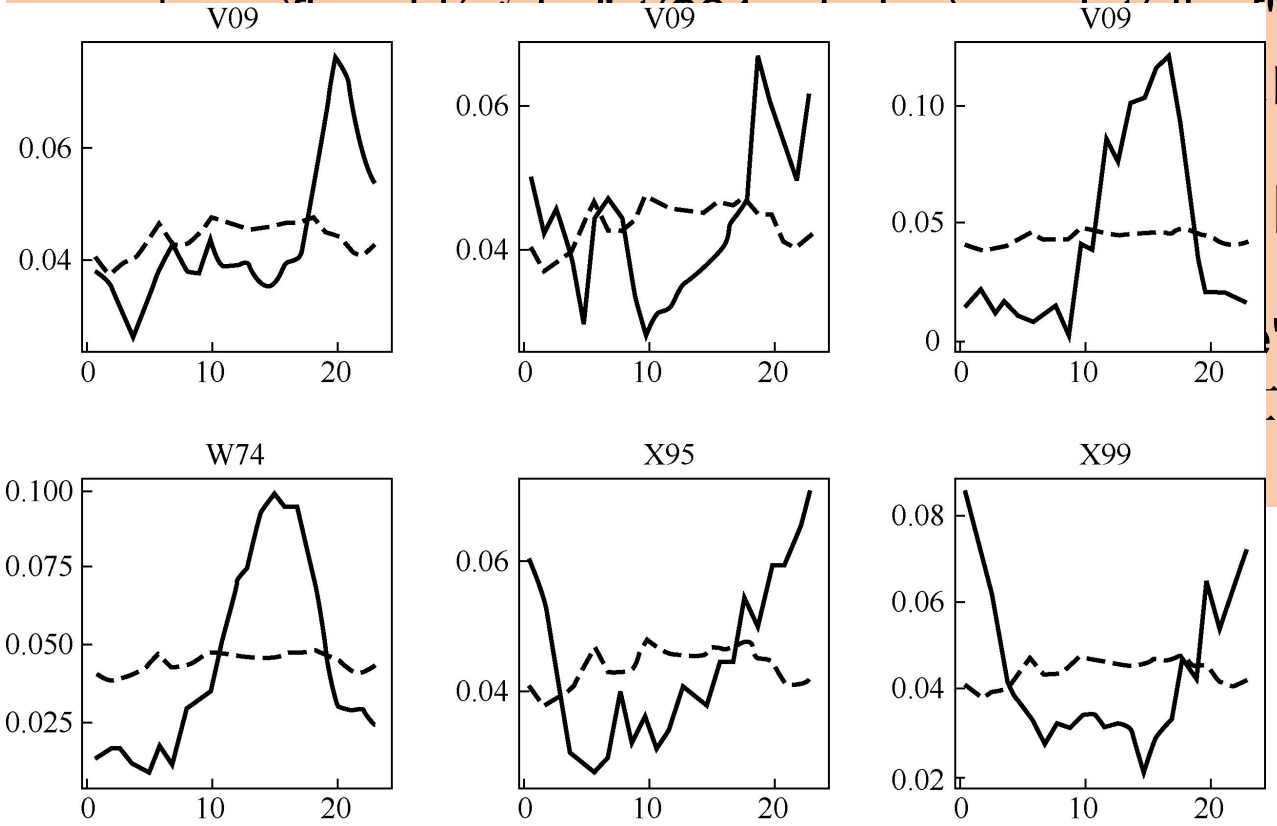
```
'hod'],dise['prop'])
```

的曲线图

图

```
'].item())
```

死因



(7)查看总死亡人数超过 350 的非寻常死亡疾病

```
In[18]|m_d_unusual_smaller['code'].unique()
```

对应输出结果为：

```
array(['R95', 'W73', 'X97', 'Y21', 'V79', 'X33', 'V95'], dtype=object)
```

(8)绘制总死亡人数小于350的非寻常死亡疾病死亡时间图

对应输出结果为：

R95 : Sudden infant death syndrome
W73 : Other specified drowning and submersion
X97 : Assault (homicide) by smoke, fire, and flames
Y21 : Drowning and submersion undetermined intent
V79 : Bus occupant injured in other and unspecified transport accidents
X33 : Victim of lightning
V95 : Accident to powered aircraft causing injury to occupant

```
In[18]:fig2 = plt.figure()
for index,cod in enumerate(m_d_unusual_smaller['code'].unique()):
    dise = m_d_50[m_d_50['code']==cod]
    ax = fig2.add_subplot(331 + index)
    ax.plot(dise['hold'],dise['prop'])
```

间变化的曲线图

b_all']）

的曲线图

lisease'].item())

非寻常死亡死因