

# 全球疫情分析

*SXM*

*2020-02*

## 1 读取数据及处理

```
1 import pandas as pd
2 import numpy as np
3
4 #疫情的确诊数 (confirmed)
5 path='./data/COVID-19/csse_covid_19_data/csse_covid_19_time_series/'
6 confirmed = pd.read_csv(path+'time_series_19-covid-Confirmed.csv')
7
8 #治愈数
9 recovered = pd.read_csv(path+'time_series_19-covid-Recovered.csv')
10
11 #死亡数
12 deaths = pd.read_csv(path+'time_series_19-covid-Deaths.csv')
13
14 print(confirmed.shape)
15
16 print(recovered.shape)
17
18 print(deaths.shape)
```

confirmed 表里面包含发生疫情的国家, 经纬度, 以及从 2020 年 1 月 22 日至今的每日的确诊数; recovered 表则记录了治愈数; deaths 表则记录了死亡数。

### 1.1 查看发生疫情国家

```
1 countries = confirmed['Country/Region'].unique()
2 print('发生疫情国家: \n{}'.format(countries))
3
4 ## 发生疫情国家:
5 ## ['Mainland China' 'Thailand' 'Japan' 'South Korea' 'Taiwan' 'US' '
6 ## 'Hong Kong' 'Singapore' 'Vietnam' 'France' 'Nepal' 'Malaysia' '
7 ## 'Australia' 'Cambodia' 'Sri Lanka' 'Germany' 'Finland'
8 ## 'United Arab Emirates' 'Philippines' 'India' 'Italy' 'UK' 'Russia'
9 ## 'Sweden' 'Spain' 'Belgium' 'Others' 'Egypt' 'Iran' 'Lebanon' 'Iraq'
10 ## 'Oman' 'Afghanistan' 'Bahrain' 'Kuwait' 'Algeria' 'Croatia' '
11 ## 'Austria' 'Israel' 'Pakistan' 'Brazil' 'Georgia' 'Greece'
12 ## 'North Macedonia' 'Norway' 'Romania' 'Denmark' 'Estonia' '
13 ## 'Netherlands'
14 ## 'San Marino' 'Belarus' 'Iceland' 'Lithuania' 'Mexico' 'New Zealand'
15 ## 'Nigeria' 'Ireland' 'Luxembourg' 'Monaco' 'Qatar' 'Ecuador' '
16 ## 'Azerbaijan'
```

```

12  ## 'Czech Republic' 'Armenia' 'Dominican Republic' 'Indonesia' '
    Portugal'
13  ## 'Andorra' 'Latvia' 'Morocco' 'Saudi Arabia' 'Senegal']
1  print('发生疫情国家数: \n{}'.format(countries.shape[0]))
1  ## 发生疫情国家数:
2  ## 75

```

## 1.2 每日所有地区新冠肺炎的确诊数，治愈数，死亡数。

```

1  all_confirmed = np.sum(confirmed.iloc[:,4:])
2  all_recovered = np.sum(recovered.iloc[:,4:])
3  all_deaths = np.sum(deaths.iloc[:,4:])
4  All = pd.DataFrame({'all_confirmed':all_confirmed,'all_recovered':all_
    recovered,'all_deaths':all_deaths})
5  All.to_csv('./data/All.csv')

```

表 1 全球每日新冠肺炎数据

日期	确诊数	治愈数	死亡数
1/22/20	555	28	17
1/23/20	653	30	18
1/24/20	941	36	26
1/25/20	1434	39	42
1/26/20	2118	52	56
1/27/20	2927	61	82
1/28/20	5578	107	131
1/29/20	6166	126	133
1/30/20	8234	143	171
1/31/20	9927	222	213
2/1/20	12038	284	259
2/2/20	16787	472	362
2/3/20	19881	623	426
2/4/20	23892	852	492
2/5/20	27636	1124	564
2/6/20	30818	1487	634
2/7/20	34392	2011	719
2/8/20	37121	2616	806
2/9/20	40151	3244	906
2/10/20	42763	3946	1013
2/11/20	44803	4683	1113
2/12/20	45222	5150	1118
2/13/20	60370	6295	1371
2/14/20	66887	8058	1523

表 1 全球每日新冠肺炎数据 (续)

日期	确诊数	治愈数	死亡数
2/15/20	69032	9395	1666
2/16/20	71226	10865	1770
2/17/20	73260	12583	1868
2/18/20	75138	14352	2007
2/19/20	75641	16121	2122
2/20/20	76199	18177	2247
2/21/20	76843	18890	2251
2/22/20	78599	22886	2458
2/23/20	78985	23394	2469
2/24/20	79570	25227	2629
2/25/20	80415	27905	2708
2/26/20	81397	30384	2770
2/27/20	82756	33277	2814
2/28/20	84122	36711	2872
2/29/20	86013	39782	2941
3/1/20	88371	42716	2996
3/2/20	90313	45604	3085

### 1.3 中国大陆新冠肺炎的情况

```

1 last_update=confirmed.columns[-1] #设置最新数据日期
2 China_cases=confirmed[['Province/State',last_update]][confirmed['Country
   /Region']=='Mainland China']
3 China_cases['recovered']=recovered[['last_update']][recovered['Country/
   Region']=='Mainland China']
4 China_cases['deaths']=deaths[['last_update']][deaths['Country/Region']=='
   Mainland China']
5 China_cases = China_cases.set_index('Province/State')
6 China_cases = China_cases.rename(columns = {last_update:'confirmed'})
7 China_cases.to_csv('./data/Chinacases.csv')

```

表 2 中国大陆新冠肺炎数据

省份	确诊数	治愈数	死亡数
Anhui	990	917	6
Beijing	414	282	8
Chongqing	576	469	6
Fujian	296	255	1
Gansu	91	85	2
Guangdong	1350	1059	7

表 2 中国大陆新冠肺炎数据 (续)

省份	确诊数	治愈数	死亡数
Guangxi	252	192	2
Guizhou	146	114	2
Hainan	168	151	5
Hebei	318	296	6
Heilongjiang	480	356	13
Henan	1272	1205	22
Hubei	67103	33934	2803
Hunan	1018	887	4
Inner Mongolia	75	54	0
Jiangsu	631	543	0
Jiangxi	935	850	1
Jilin	93	83	1
Liaoning	122	103	1
Ningxia	74	69	0
Qinghai	18	18	0
Shaanxi	245	216	1
Shandong	758	460	6
Shanghai	337	292	3
Shanxi	133	119	0
Sichuan	538	386	3
Tianjin	136	111	3
Tibet	1	1	0
Xinjiang	76	66	3
Yunnan	174	168	2
Zhejiang	1206	1069	1

#### 1.4 中国大陆治愈率 VS 死亡率

```

1 confirmed_china = confirmed[confirmed['Country/Region']=='Mainland China
   ']
2 confirmed_china = np.sum(confirmed_china.iloc[:,4:])
3 recovered_china = recovered[recovered['Country/Region'] == 'Mainland
   China']
4 recovered_china = np.sum(recovered_china.iloc[:,4:])
5 deaths_china = deaths[deaths['Country/Region'] == 'Mainland China']
6 deaths_china = np.sum(deaths_china.iloc[:,4:])
7 recover_rate = (recovered_china/confirmed_china)*100
8 recover_rate1=(recover_rate/100).apply(lambda x: format(x, '.2%'))
9 death_rate = (deaths_china/confirmed_china)*100
10 death_rate1 = (death_rate/100).apply(lambda x: format(x, '.2%'))

```

```

11 re_de=pd.DataFrame({'recover_rate':recover_rate1,'death_rate':death_
    rate1})
12 re_de.to_csv('./data/rede.csv')

```

表 3 中国大陆治愈率 VS 死亡率

日期	治愈率	死亡率
1/22/20	5.12%	3.11%
1/23/20	4.69%	2.82%
1/24/20	3.93%	2.84%
1/25/20	2.79%	3.00%
1/26/20	2.38%	2.72%
1/27/20	2.03%	2.86%
1/28/20	1.84%	2.38%
1/29/20	1.98%	2.19%
1/30/20	1.66%	2.10%
1/31/20	2.19%	2.18%
2/1/20	2.32%	2.18%
2/2/20	2.79%	2.17%
2/3/20	3.12%	2.16%
2/4/20	3.56%	2.07%
2/5/20	4.07%	2.05%
2/6/20	4.83%	2.07%
2/7/20	5.86%	2.10%
2/8/20	7.06%	2.19%
2/9/20	8.09%	2.27%
2/10/20	9.26%	2.39%
2/11/20	10.46%	2.51%
2/12/20	11.36%	2.50%
2/13/20	10.38%	2.29%
2/14/20	12.03%	2.29%
2/15/20	13.60%	2.43%
2/16/20	15.26%	2.51%
2/17/20	17.21%	2.57%
2/18/20	19.15%	2.70%
2/19/20	21.40%	2.84%
2/20/20	24.00%	2.98%
2/21/20	24.77%	2.96%
2/22/20	29.49%	3.17%
2/23/20	30.12%	3.18%

表 3 中国大陆治愈率 VS 死亡率 (续)

日期	治愈率	死亡率
2/24/20	32.39%	3.36%
2/25/20	35.60%	3.43%
2/26/20	38.50%	3.48%
2/27/20	41.91%	3.50%
2/28/20	46.04%	3.54%
2/29/20	49.56%	3.58%
3/1/20	52.76%	3.60%
3/2/20	55.99%	3.64%

## 1.5 其他地区治愈率 VS 死亡率

```

1 confirmed_others = confirmed[confirmed['Country/Region'] != 'Mainland
   China']
2 confirmed_others = np.sum(confirmed_others.iloc[:,4:])
3 recovered_others = recovered[recovered['Country/Region'] != 'Mainland
   China']
4 recovered_others = np.sum(recovered_others.iloc[:,4:])
5 deaths_others = deaths[deaths['Country/Region'] != 'Mainland China']
6 deaths_others = np.sum(deaths_others.iloc[:,4:])
7 other_recover_rate = (recovered_others/confirmed_others)*100
8 other_recover_rate1=(other_recover_rate/100).apply(lambda x: format(x, '
   .2%'))
9 other_death_rate = (deaths_others/confirmed_others)
10 other_death_rate1 = (other_death_rate/100).apply(lambda x: format(x, '
   .2%'))
11 other_re_de=pd.DataFrame({'recover_rate':other_recover_rate1,'death_rate
   ':other_death_rate1})
12 other_re_de.to_csv('./data/otherrede.csv')

```

表 4 其他地区治愈率 VS 死亡率

日期	治愈率	死亡率
1/22/20	0.00%	0.00%
1/23/20	0.00%	0.00%
1/24/20	0.00%	0.00%
1/25/20	0.00%	0.00%
1/26/20	5.36%	0.00%
1/27/20	4.69%	0.00%
1/28/20	7.14%	0.00%
1/29/20	6.25%	0.00%
1/30/20	7.27%	0.00%
1/31/20	5.56%	0.00%

表 4 其他地区治愈率 VS 死亡率 (续)

日期	治愈率	死亡率
2/1/20	5.39%	0.00%
2/2/20	5.00%	0.01%
2/3/20	4.79%	0.01%
2/4/20	4.25%	0.01%
2/5/20	3.96%	0.01%
2/6/20	4.15%	0.01%
2/7/20	4.10%	0.01%
2/8/20	6.12%	0.01%
2/9/20	7.20%	0.01%
2/10/20	6.35%	0.00%
2/11/20	10.08%	0.00%
2/12/20	13.58%	0.00%
2/13/20	15.24%	0.01%
2/14/20	14.29%	0.01%
2/15/20	14.74%	0.01%
2/16/20	15.00%	0.01%
2/17/20	14.29%	0.01%
2/18/20	15.32%	0.01%
2/19/20	15.43%	0.01%
2/20/20	14.58%	0.01%
2/21/20	14.37%	0.01%
2/22/20	11.87%	0.01%
2/23/20	10.94%	0.01%
2/24/20	9.80%	0.01%
2/25/20	9.26%	0.02%
2/26/20	9.93%	0.02%
2/27/20	8.90%	0.02%
2/28/20	7.93%	0.02%
2/29/20	7.44%	0.02%
3/1/20	7.00%	0.01%
3/2/20	7.72%	0.02%

## 1.6 世界其他地区疫情数量

```

1 others = confirmed[['Country/Region',last_update]][confirmed['Country/
   Region'] != 'Mainland China']
2 others['recovered'] = recovered[['last_update']][recovered['Country/Region
   '] != 'Mainland China']

```

```

3 others['death'] = deaths[[last_update]][deaths['Country/Region'] != '
   Mainland China']
4 others_countries = others.rename(columns = {last_update:'confirmed'})
5 others_countries = others_countries.set_index('Country/Region')
6 others_countries = others_countries.groupby('Country/Region').sum()
7 others_countries.to_csv('./data/otherscountries.csv')

```

表 5 世界其他地区疫情数量

地区	确诊数	治愈数	死亡数
Afghanistan	1	0	0
Algeria	3	0	0
Andorra	1	0	0
Armenia	1	0	0
Australia	30	11	1
Austria	18	0	0
Azerbaijan	3	0	0
Bahrain	49	0	0
Belarus	1	0	0
Belgium	8	1	0
Brazil	2	0	0
Cambodia	1	1	0
Canada	27	6	0
Croatia	7	0	0
Czech Republic	3	0	0
Denmark	4	0	0
Dominican Republic	1	0	0
Ecuador	6	0	0
Egypt	2	1	0
Estonia	1	0	0
Finland	6	1	0
France	191	12	3
Georgia	3	0	0
Germany	159	16	0
Greece	7	0	0
Hong Kong	100	36	2
Iceland	6	0	0
India	5	3	0
Indonesia	2	0	0
Iran	1501	291	66
Iraq	26	0	0



表 5 世界其他地区疫情数量 (续)

地区	确诊数	治愈数	死亡数
Ireland	1	0	0
Israel	10	1	0
Italy	2036	149	52
Japan	274	32	6
Kuwait	56	0	0
Latvia	1	0	0
Lebanon	13	0	0
Lithuania	1	0	0
Luxembourg	1	0	0
Macau	10	8	0
Malaysia	29	18	0
Mexico	5	0	0
Monaco	1	0	0
Morocco	1	0	0
Nepal	1	1	0
Netherlands	18	0	0
New Zealand	1	0	0
Nigeria	1	0	0
North Macedonia	1	0	0
Norway	25	0	0
Oman	6	1	0
Others	705	10	6
Pakistan	4	0	0
Philippines	3	1	1
Portugal	2	0	0
Qatar	3	0	0
Romania	3	0	0
Russia	3	2	0
San Marino	8	0	0
Saudi Arabia	1	0	0
Senegal	1	0	0
Singapore	108	78	0
South Korea	4335	30	28
Spain	120	2	0
Sri Lanka	1	1	0
Sweden	15	0	0

表 5 世界其他地区疫情数量 (续)

地区	确诊数	治愈数	死亡数
Switzerland	42	0	0
Taiwan	41	12	1
Thailand	43	31	1
UK	40	8	0
US	105	9	6
United Arab Emirates	21	5	0
Vietnam	16	16	0

## 2 数据可视化

### 2.1 全球疫情变化趋势图

```

1 import matplotlib.pyplot as plt
2 import matplotlib.ticker as tk
3 fig,ax = plt.subplots()
4 plt.rcParams['font.sans-serif']=['SimHei']#用来正常显示中文标签
5 plt.rcParams['axes.unicode_minus']=False#用来显示正常负号
6 xt=[d[:-3] for d in all_confirmed.index]#取月日
7 ax.plot(xt,all_confirmed,c='r',label='确诊(例)',marker='o',linewidth
      =1,markersize=2)
8 ax.plot(xt,all_recovered,c='b',label='治愈(例)',marker='o',
      linewidth=1,markersize=2)
9 ax.plot(xt,all_deaths,c='lime',label='死亡(例)',marker='o',
      linewidth=1,markersize=2)
10 ax.xaxis.set_major_locator(tk.MultipleLocator(2))
11 ax.xaxis.set_minor_locator(tk.MultipleLocator(1))
12 plt.xticks(rotation=45)

1 plt.yticks()

1 plt.xlabel('时 间')
2 plt.ylabel('数 目')
3 plt.legend(loc="upper left",fontsize=8)
4 plt.tight_layout()
5 plt.show()

```

图1显示，新冠肺炎确诊人数逐渐上升，治愈人数也在上升，死亡人数上升缓慢。

### 2.2 中国大陆每个省份的疫情数量图

```

1 Mainland_china = China_cases.sort_values(by='confirmed',ascending=True)
2 Mainland_china.plot(kind='barh', color = ['red','blue','lime'],figsize
      =(20,30),
3 width=1)
4 plt.ylabel('省/市',fontsize=35)
5 plt.xlabel('数量',fontsize=35)

```

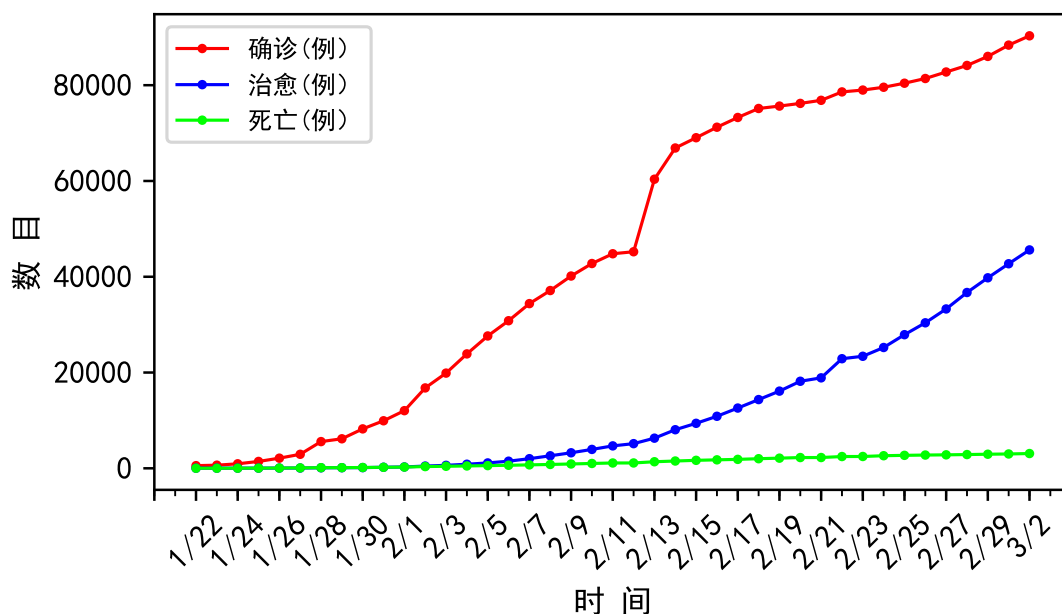


图 1 全球疫情变化趋势

```

6 plt.yticks(fontsize = 30)
1 plt.xticks(fontsize = 30)
1 plt.legend(bbox_to_anchor=(0.95,0.95),fontsize = 30)
2 plt.tight_layout()
3 plt.show()

```

从图2可以看到，湖北省三项数据高居第一位，且远远高于其他省份。

### 2.3 中国大陆治愈率 VS 死亡率趋势图

```

1 fig,ax = plt.subplots()
2 xt=[d[:-3] for d in all_confirmed.index]#取月日
3 ax.plot(xt,recover_rate, color = 'blue', label = '治愈率 (%)', marker =
  'o',linewidth=1,markersize=2)
4 ax.plot(xt,death_rate, color = 'lime', label = '死亡率 (%)', marker = 'o',
  linewidth=1,markersize=2)
5 ax.xaxis.set_major_locator(tk.MultipleLocator(2))
6 ax.xaxis.set_minor_locator(tk.MultipleLocator(1))
7 plt.ylabel('数量')
8 plt.xlabel('时间')
9 plt.xticks(rotation=45)
1 plt.yticks()
1 plt.legend(loc = "upper left",fontsize = 8)
2 plt.tight_layout()
3 plt.show()

```

图3显示在 1 月 25 日-1 月 31 日期间，中国大陆死亡率略高于治愈率，但其他时间段，治愈率远远高于死亡率

### 2.4 其他地区治愈率 VS 死亡率趋势图

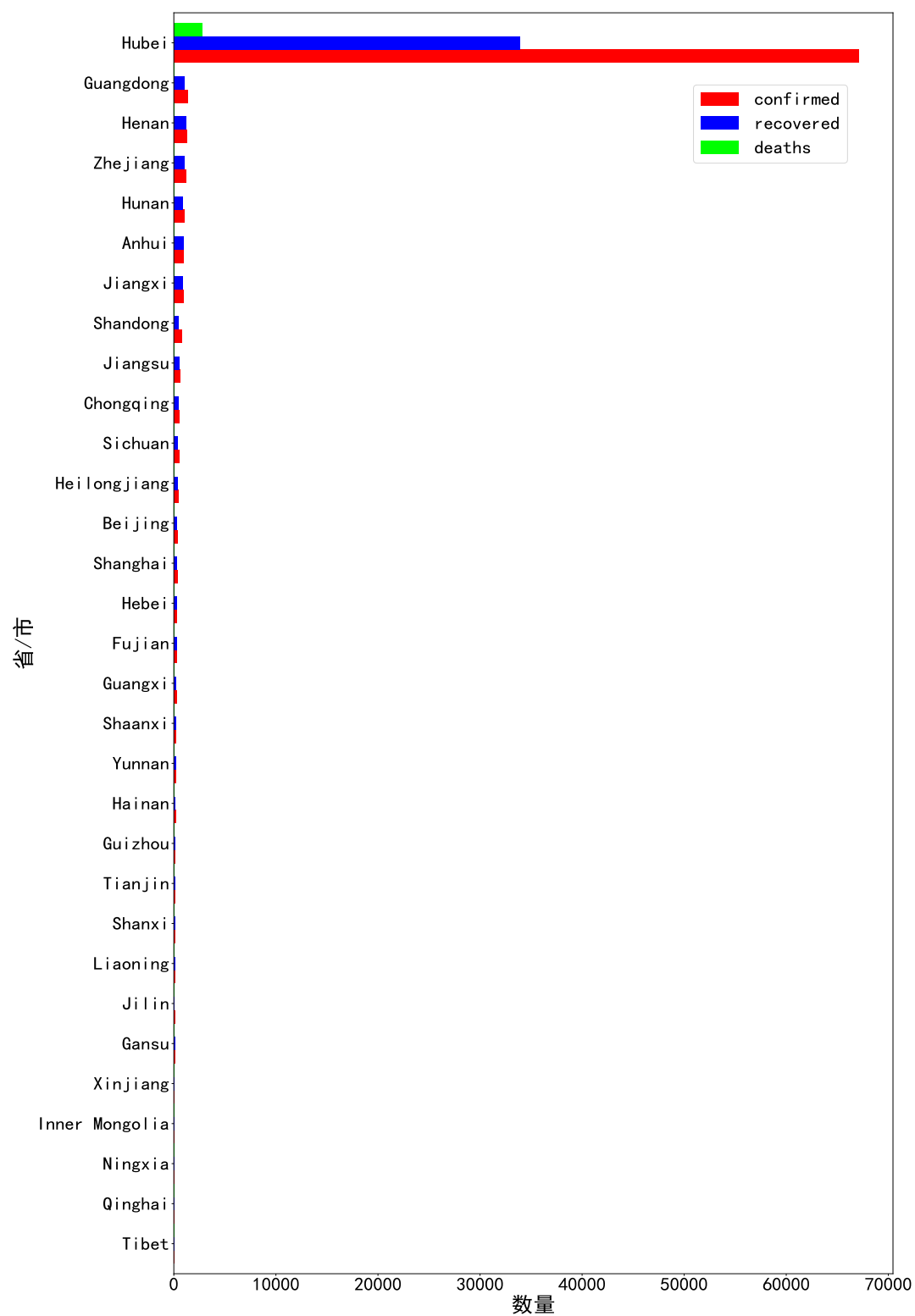


图 2 中国大陆各省市疫情数量

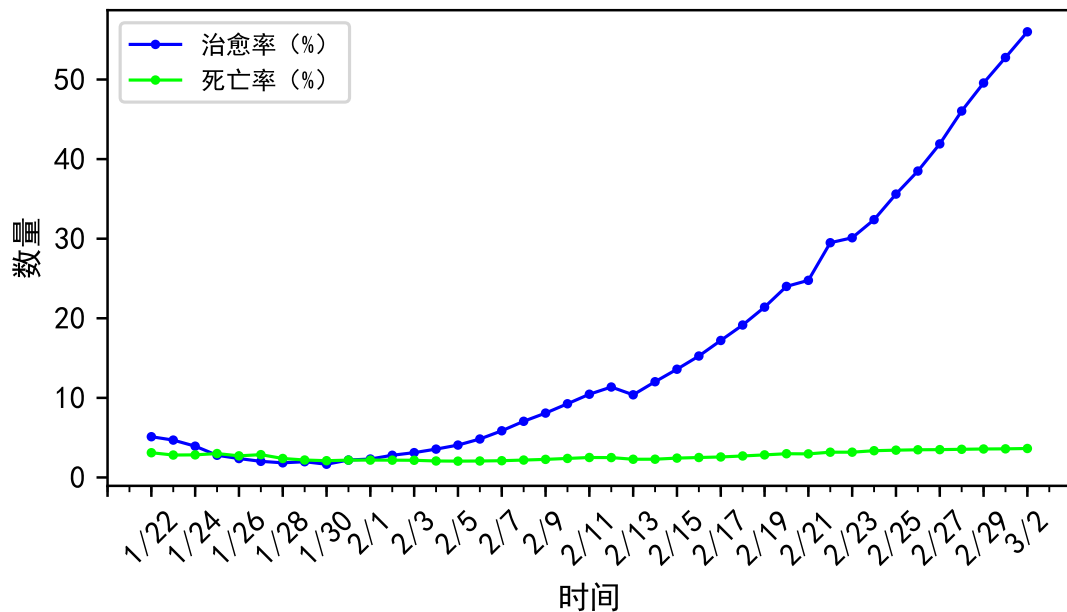


图 3 中国大陆治愈率 VS 死亡率

```

1 fig,ax = plt.subplots()
2 xt=[d[:-3] for d in all_confirmed.index]#取月日
3 ax.plot(xt,other_recover_rate, color = 'blue', label = '治愈率 (%) ',
4         marker = 'o',linewidth=1,markersize=2)
5 ax.plot(xt,other_death_rate, color = 'lime', label = '死亡率 (%) ',
6         marker = 'o',linewidth=1,markersize=2)
7 ax.xaxis.set_major_locator(tk.MultipleLocator(2))
8 ax.xaxis.set_minor_locator(tk.MultipleLocator(1))
9 plt.ylabel('数量')
10 plt.xlabel('时间')
11 plt.xticks(rotation=45)

1 plt.yticks()

1 plt.legend(loc = "upper left",fontsize = 8)
2 plt.tight_layout()
3 plt.show()

```

从图4可以看出，其他地区的治愈率从 2 月 24 日开始显著下降，说明新冠肺炎疫情已经蔓延至其他国家，患病人数普遍增加导致治愈率下降。

## 2.5 世界其他地区疫情数量

```

1 others_countries.sort_values(by = 'confirmed',ascending = True).plot(
2     kind='barh', color = ['red','blue','lime'],figsize=(20,30), width=1)
3 plt.ylabel('Country/Region',fontsize = 35)
4 plt.xlabel('数量',fontsize = 35)
5 plt.yticks(fontsize = 30)

1 plt.xticks(fontsize = 30)

1 plt.legend(bbox_to_anchor=(0.95,0.95),fontsize = 30)
2 plt.tight_layout()

```

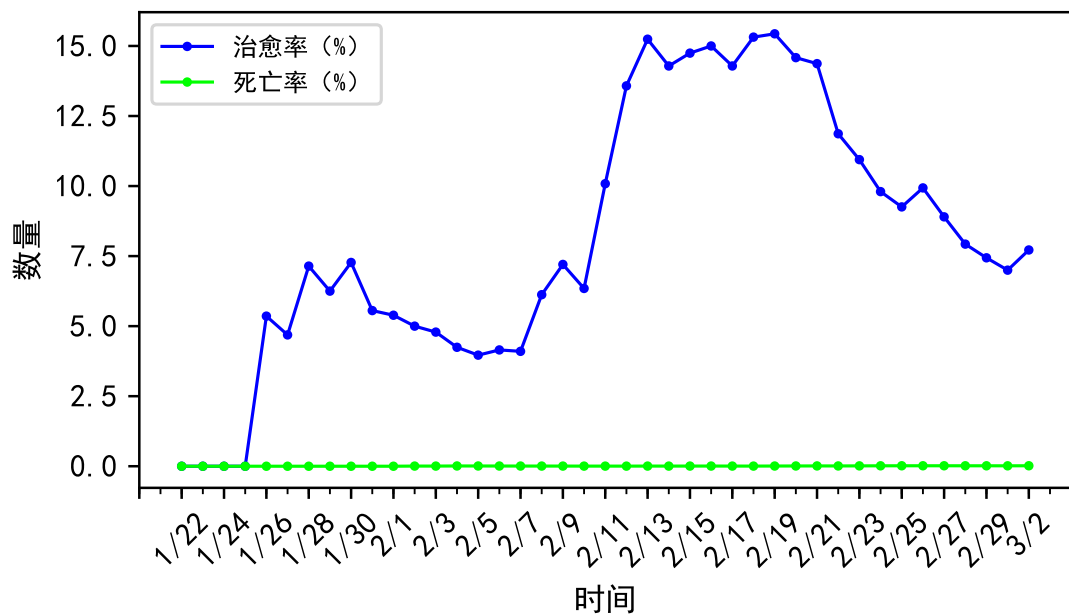


图 4 其他地区治愈率 VS 死亡率

```
plt.show()
```

图5显示，韩国、意大利、伊朗，日本疫情较为严重，成为除中国外疫情较为严重的国家。

### 3 绘制疫情地图

#### 3.1 用 folium 包绘制

```
1 # 疫情地图数据
2 others=confirmed[['Country/Region','Lat','Long',last_update]][confirmed[
   'Country/Region'] != 'Mainland China']
3 others['recovered'] = recovered[[last_update]][recovered['Country/Region']
   != 'Mainland China']
4 others['death'] = deaths[[last_update]][deaths['Country/Region'] != '
   Mainland China']
5 others_countries = others.rename(columns = {last_update:'confirmed'})
6 others_countries.loc['94'] = ['Mainland China',30.9756,112.2707,
   confirmed_china[-1],recovered_china[-1],deaths_china[-1]]

1 import folium
2 world_map = folium.Map(location=[10, -20], zoom_start=2.3,tiles='Stamen
   Terrain')
3
4 for lat, lon, value, name in zip(others_countries['Lat'], others_
   countries['Long'],
5 others_countries['confirmed'], others_countries['Country/Region']):
6     folium.CircleMarker([lat, lon],
7         radius=10,
8         popup = ('<strong>Country</strong>: ' + str(name).capitalize() + '<
   br>'
```

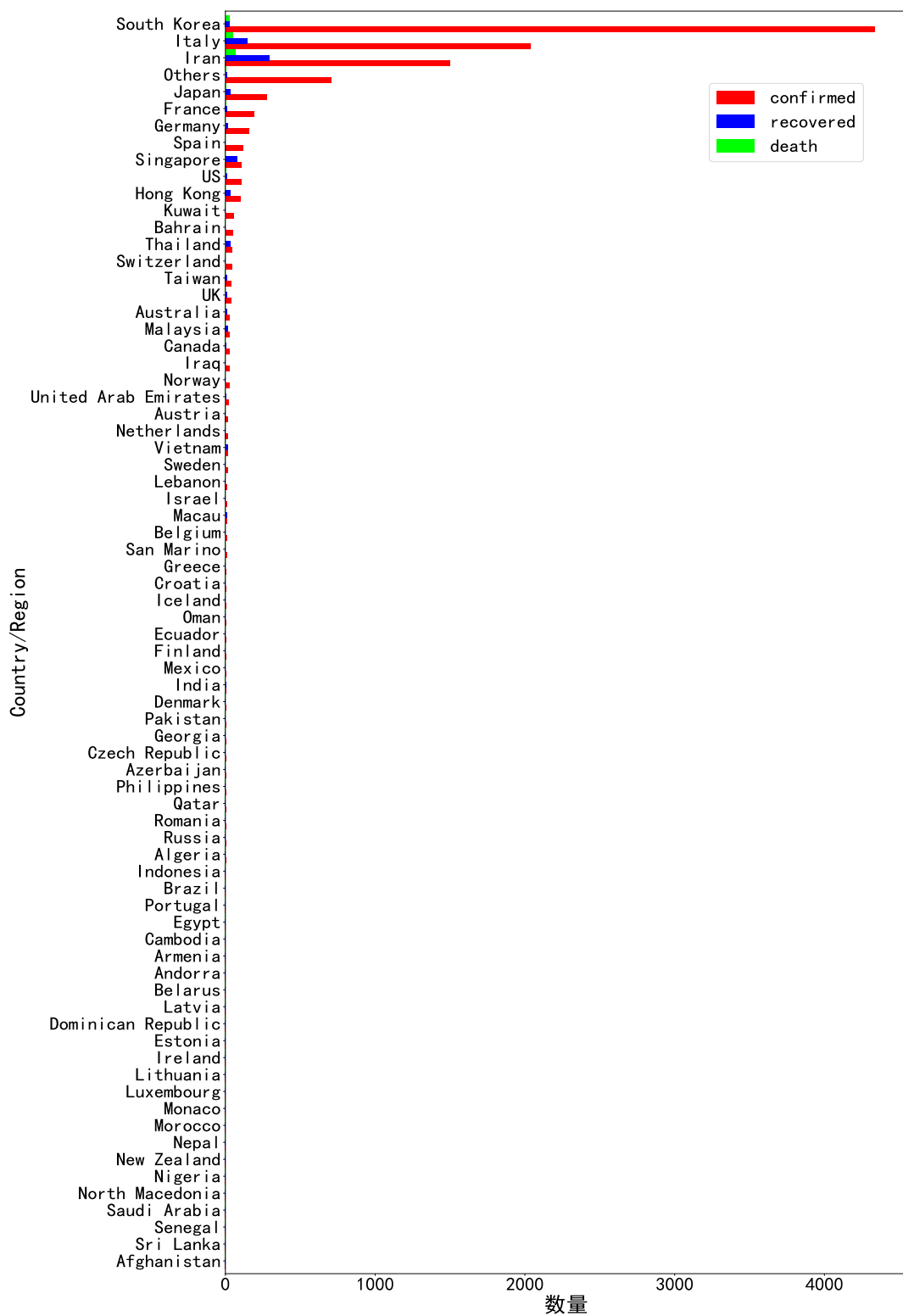


图 5 世界其他地区疫情数量

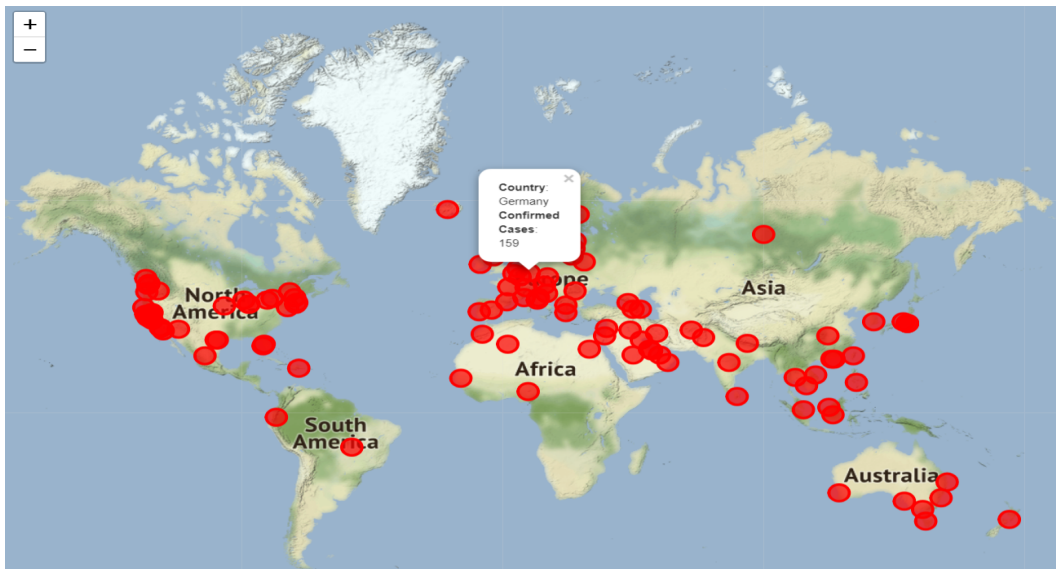


图 6 亚洲地区疫情扩散图

```

9     '<strong>Confirmed Cases</strong>: ' + str(value) + '<br>'),
10     color='red',
11     fill_color='red',
12     fill_opacity=0.7 ).add_to(world_map)

1 world_map

1 world_map.save("wordmap.html")
2 import webbrowser
3 webbrowser.open('wordmap.html')
```

用 folium 绘制每日疫情扩散地图如图6所示。这是一种可交互的地图，可以随意移动缩放，鼠标点击地图上红点，即可出现地区的疫情信息。展示图为静态，运行代码，可在网页中出现动态图

### 3.2 用 plotly 绘制每日疫情扩散地图

```

1 import plotly.express as px
2
3 #确诊数
4 confirmed = confirmed.melt(id_vars = ['Province/State', 'Country/Region',
5     'Lat', 'Long'],var_name='date',value_name = 'confirmed')
6
7 #把date列转换成datetime格式
8 confirmed['date_dt'] = pd.to_datetime(confirmed.date, format="%m/%d/%y")
9 confirmed.date = confirmed.date_dt.dt.date
10 confirmed.rename(columns={'Country/Region': 'country', 'Province/State':
11     'province'}, inplace=True)
12
13 #治愈数、死亡数
14 recovered = recovered.melt(id_vars = ['Province/State', 'Country/Region',
15     'Lat', 'Long'],var_name='date',value_name = 'recovered')
16 recovered['date_dt'] = pd.to_datetime(recovered.date, format="%m/%d/%y")
```



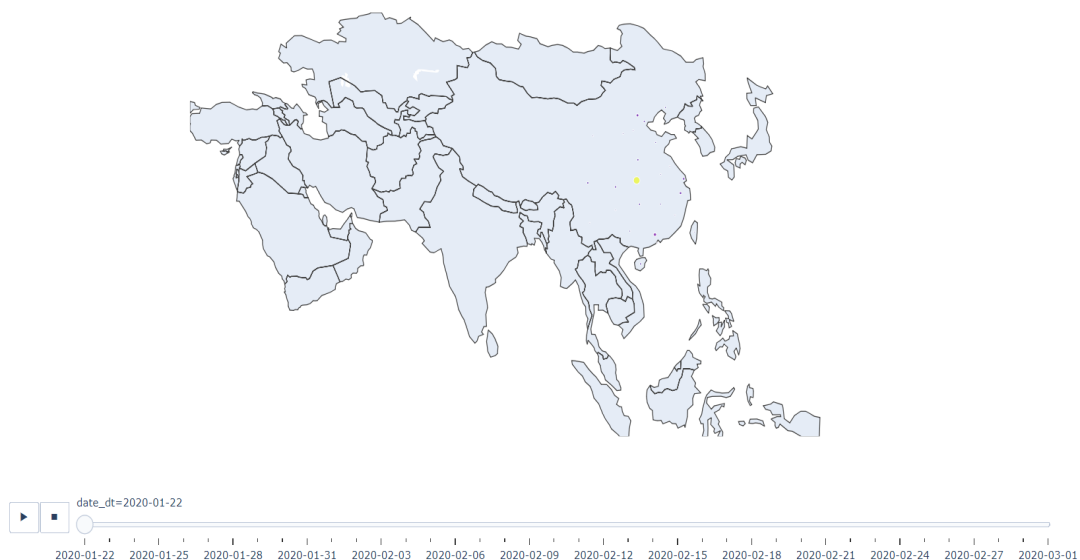


图 7 亚洲地区疫情扩散图

```

16 recovered.date = recovered.date_dt.dt.date
17 recovered.rename(columns={'Country/Region': 'country', 'Province/State':
    'province'}, inplace=True)
18
19 deaths = deaths.melt(id_vars = ['Province/State', 'Country/Region', 'Lat
    ', 'Long'], var_name='date', value_name = 'deaths')
20 deaths['date_dt'] = pd.to_datetime(deaths.date, format="%m/%d/%y")
21 deaths.date = deaths.date_dt.dt.date
22 deaths.rename(columns={'Country/Region': 'country', 'Province/State': '
    province'}, inplace=True)
23
24 #将三种数据合并在一起
25 merge_on = ['province', 'country', 'date']
26 all_date = confirmed.merge(deaths[merge_on + ['deaths']], how='left', on
    =merge_on). \
27 merge(recovered[merge_on + ['recovered']], how='left', on=merge_on)
28
29 Coronavirus_map = all_date.groupby(['date_dt', 'province'])['confirmed',
    'deaths',
30 'recovered', 'Lat', 'Long'].max().reset_index()
31 Coronavirus_map['size'] = Coronavirus_map.confirmed.pow(0.5) # 创建实心圆
    大小
32 Coronavirus_map['date_dt'] = Coronavirus_map['date_dt'].dt.strftime('%Y
    -%m-%d')
33
34 fig = px.scatter_geo(Coronavirus_map, lat='Lat', lon='Long', scope='asia'
    ,
35 color="size", size='size', hover_name='province',
36 hover_data=['confirmed', 'deaths', 'recovered'],
37 projection="natural earth", animation_frame="date_dt",
38 title='亚洲地区疫情扩散图')
39 fig.update(layout_coloraxis_showscale=False)
1 fig.show()

```

用 plotly 绘制每日疫情扩散地图如图7所示，展示图为静态，运行代码，可在网页中出现动态图。