

# 全球抗“疫”：用 Python 带你了解世界疫情

王梦圆

2020-02

## 1 数据的读取及处理

本次使用的数据是 Github 上一个项目里的，也可以直接用 pandas 包导入，需要注意的是不能直接使用 Github 那个网址，否则会报错，需要将前面部分改成 <https://raw.githubusercontent.com/>，然后就是加入数据的目录地址。数据主要是三个文件，包含了疫情的确诊数（confirmed），治愈数（recovered），死亡数（deaths）。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 import numpy as np
2 import pandas as pd
3 import matplotlib.pyplot as plt
4 confirmed = pd.read_csv('../data/COVID-20/csse_covid_19_data/csse_covid_19_time_series/time_series_19-covid-Confirmed.csv')
5 recovered = pd.read_csv('../data/COVID-20/csse_covid_19_data/csse_covid_19_time_series/time_series_19-covid-Recovered.csv')
6 deaths = pd.read_csv('../data/COVID-20/csse_covid_19_data/csse_covid_19_time_series/time_series_19-covid-Deaths.csv')
```

数据已经导入了，让我们来看看数据是啥样的吧。head(5) 是查看数据前五；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 c1 = confirmed.head(5)
2 c1.to_csv('c1.csv')

1 recovered.head(5)

1 deaths.head(5)

1 print(confirmed.shape)

1 print(recovered.shape)

1 print(deaths.shape)
```

## 2 数据可视化

```
1 plt.rcParams['font.sans-serif'] = ['SimHei'] #用来正常显示中文标签
2 plt.rcParams['axes.unicode_minus'] = False #用来正常显示负号
3
4 countries = confirmed['Country/Region'].unique()
5 print(countries) #可以看出一共75个国家/地区都有新冠肺炎病例
6
7 #计算出每日所有地区新冠肺炎的确诊数，治愈数，死亡数
```

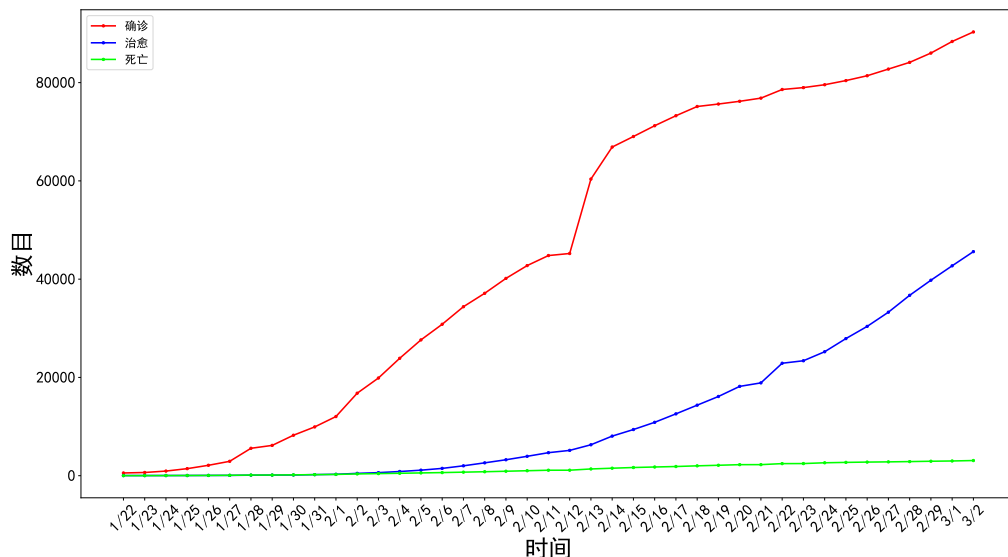


图 1 全球疫情变化趋势

```

1 plt.figure(figsize=(18,10))
2 all_confirmed = np.sum(confirmed.iloc[:,4:])
3 all_recovered = np.sum(recovered.iloc[:,4:])
4 all_deaths = np.sum(deaths.iloc[:,4:])
5 all_confirmed.index = [d[:-3] for d in all_confirmed.index]
6 all_recovered.index = [d[:-3] for d in all_recovered.index]
7 all_deaths.index = [d[:-3] for d in all_deaths.index]
8 plt.plot(all_confirmed,color = 'red',label = '确诊',marker = 'o',
9          linewidth=2,markersize=3)
10 plt.plot(all_recovered,color = 'blue',label = '治愈',marker = 'o',
11          linewidth=2,markersize=3)
12 plt.plot(all_deaths,color = 'lime',label = '死亡',marker = 'o',linewidth
13          =2,markersize=3)
14 plt.xticks(rotation = 45,size = 20)
15
16 plt.yticks(size = 20)
17
18 plt.xlabel('时间',size = 30)
19 plt.ylabel('数目',size = 30)
20 plt.legend(loc = 'upper left',fontsize = 15)
21 plt.tight_layout()
22 plt.show()

```

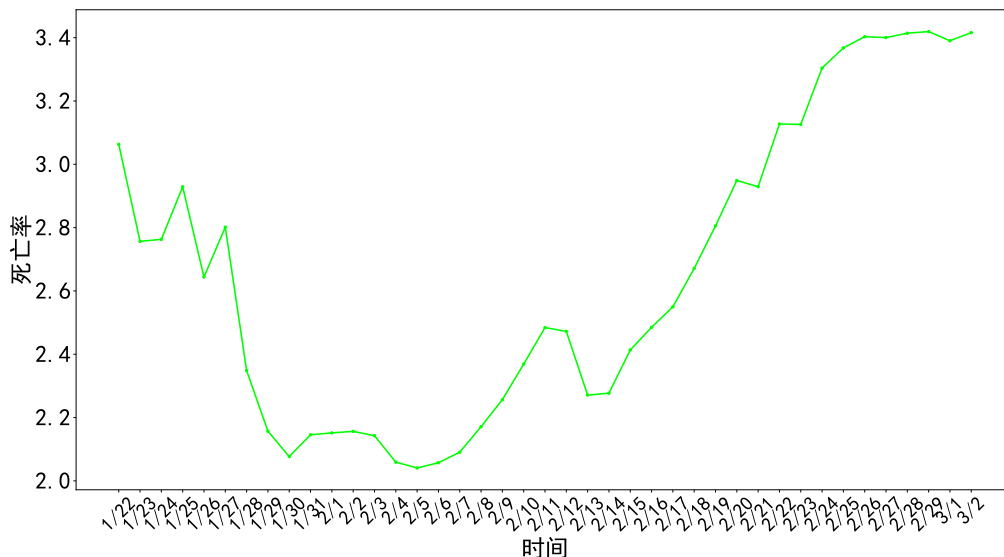
可以看出，目前新冠肺炎确诊病例还在持续增加，不过令人高兴的是治愈数也在持续增长，死亡数很少

下面看看新冠肺炎的死亡率，首先计算死亡率数据，然后就可以直接画图

```

1 plt.figure(figsize=(18,10))
2 death_rate = (all_deaths/all_confirmed)*100
3 death_rate
4
5 plt.plot(death_rate,color = 'lime',label = '死亡',marker = 'o',linewidth
6          =2,markersize=3)
7 plt.xticks(rotation = 45,size = 25)
8
9 plt.yticks(size = 30)

```



由于本次疫情主要发生在中国大陆，下面来具体研究下中国大陆的疫情情况，首先从全部数据中提取出中国大陆的数据。里面包含了省份，以及每个省最新的确诊数，治愈数，死亡数。

```
1 plt.xlabel('时间', fontsize = 30)
2 plt.ylabel('死亡率', fontsize = 30)
3 plt.tight_layout()
4 plt.show()
```

由于本次疫情主要发生在中国大陆，下面来具体研究下中国大陆的疫情情况，首先从全部数据中提取出中国大陆的数据。里面包含了省份，以及每个省最新的确诊数，治愈数，死亡数。

```
1 last_update = '3/2/20' #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
```

下面画出中国大陆每个省份的疫情数量图

```
1 Mianland_China = China_cases.sort_values(by='confirmed',ascending=True)
2 Mianland_China.plot(kind='barh',figsize=(20,30),color = ['red','blue','
    lime'],width = 1,rot = 2)
3 plt.xlabel('省/市',size = 30)
4 plt.ylabel('数量',size = 30)
5 plt.xticks(size = 25)

1 plt.yticks(size = 30)

1 plt.legend(bbox_to_anchor = (0.95,0.95),fontsize = 20)
2 plt.tight_layout()
3 plt.show()
```

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。

下面看看中国大陆的治愈率和死亡率数据，数据使用下面的代码即可计算出

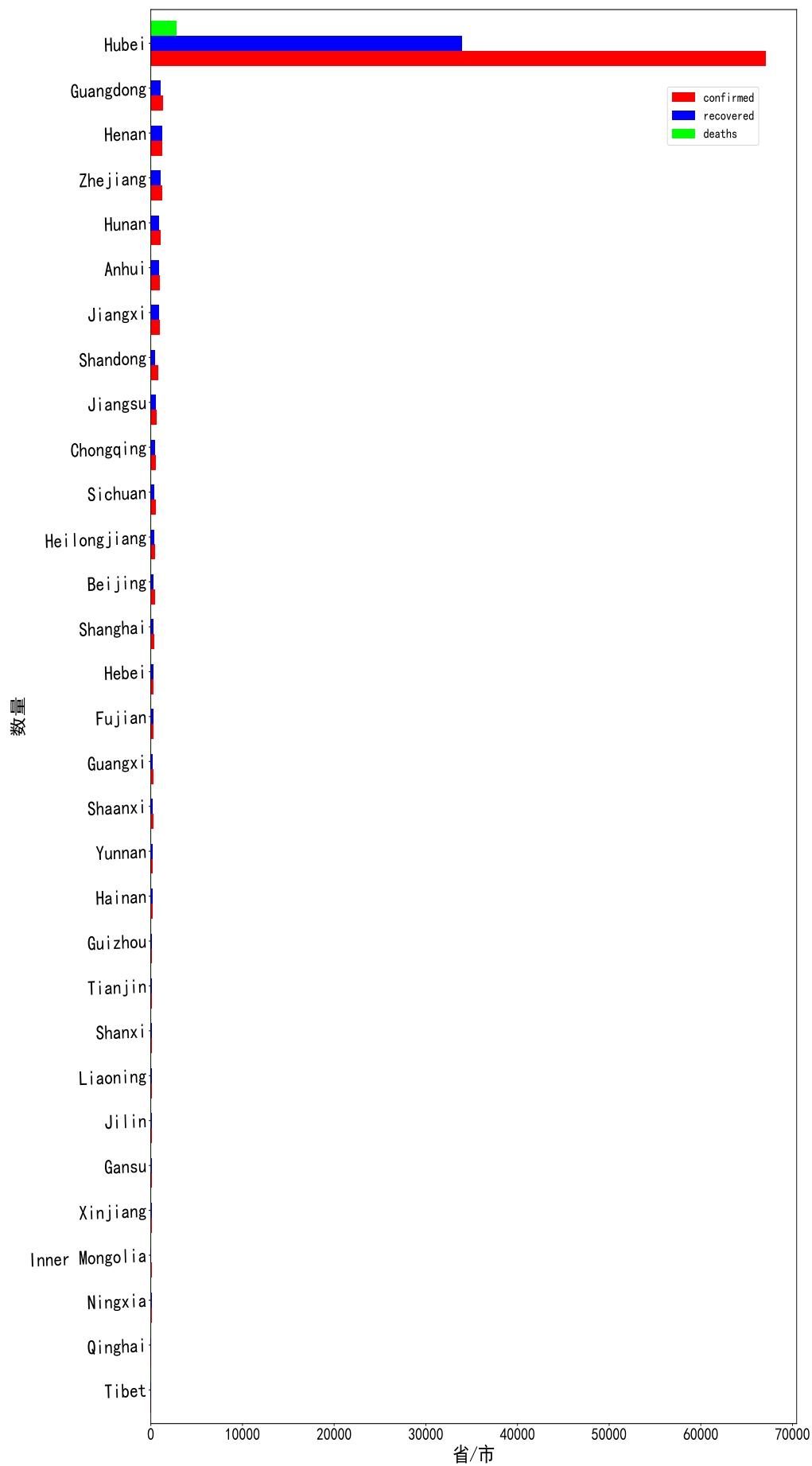


图 3 中国大陆各省市疫情数量

来，最终结果在 recover\_rate 和 death\_rate 里。

```
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 #中国地区的治愈率
8 deaths_rate = (deaths_China/confirmed_China)*100#中国各地区的死亡率
9 deaths_rate.index = [d[:-3] for d in deaths_rate.index]
10 deaths_rate

1 recover_rate.index = [d[:-3] for d in recover_rate.index]
2 recover_rate
3 #接下来画图
4 #plt.figure(figsize=(10,18))

1 plt.plot(recover_rate,color = 'blue',label = '治愈率',marker = 'o',
    linewidth=2,markersize=3)
2 plt.plot(deaths_rate,color = 'lime',label = '死亡率',marker = 'o',
    linewidth=2,markersize=3)
3 plt.xlabel('时间',size = 30)
4 plt.ylabel('数量',size = 30)
5 plt.xticks(rotation = 45,size = 25)

1 plt.yticks(size = 30)

1 plt.legend(loc = 'upper left',fontsize = 20)
```

虽然在 1 月 25 日-1 月 31 日期间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

那中国大陆其他地区这一情况咋样呢？代码大同小异，我们一起来看看，首先还是提取出其他地区的数据。

```
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 recover_rate_others = (recovered_others/confirmed_others)*100 #其他地区的治愈率
8 deaths_rate_others = (deaths_others/confirmed_others)*100#其他各地区的死亡率
9 #接下来画图
10 plt.plot(recover_rate_others,color = 'blue',label = '治愈率',marker = 'o')
11 plt.plot(deaths_rate_others,color = 'lime',label = '死亡率',marker = 'o')
12 plt.title('其他地区治愈率 VS 死亡率',size = 30)
```

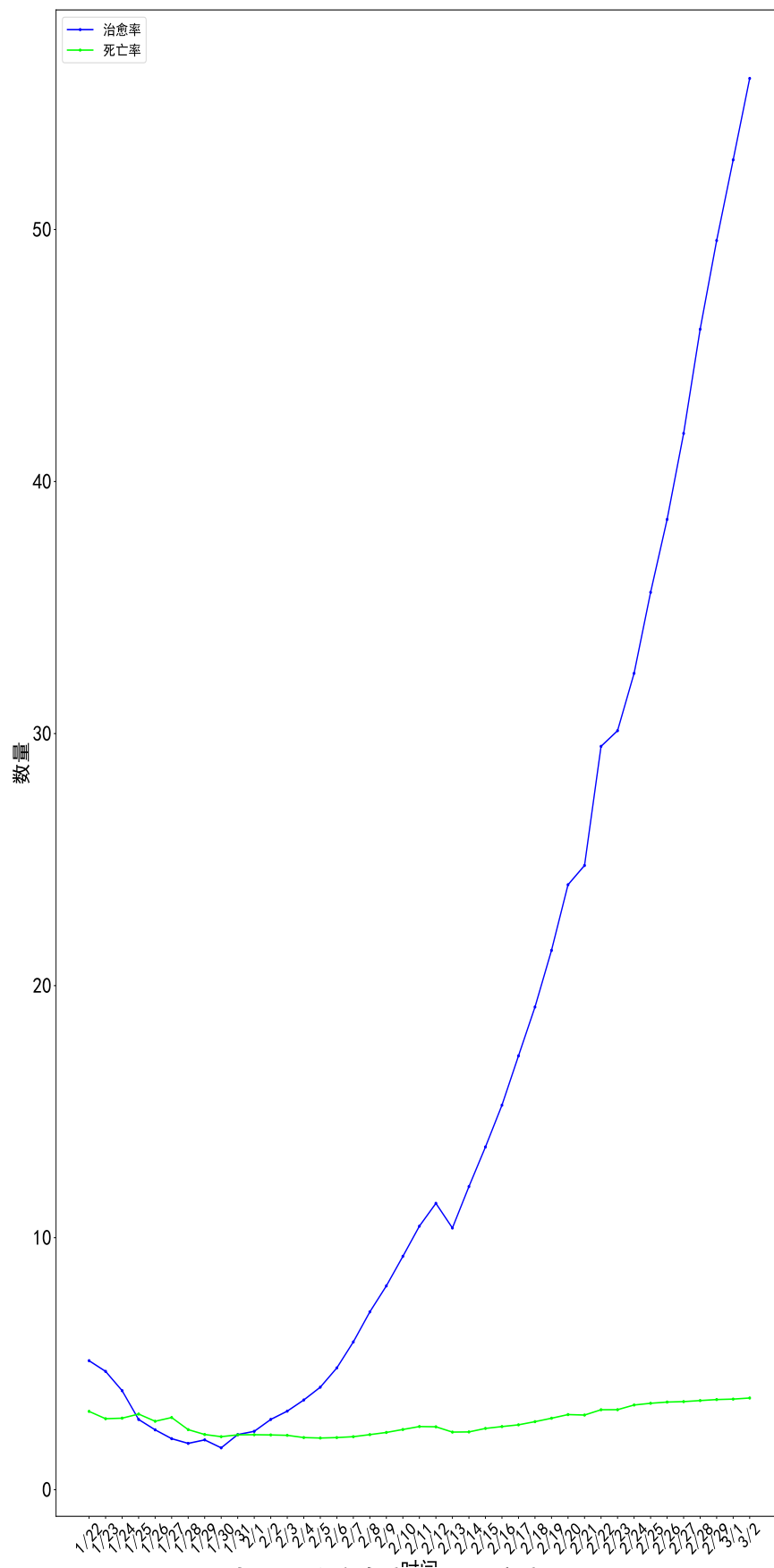
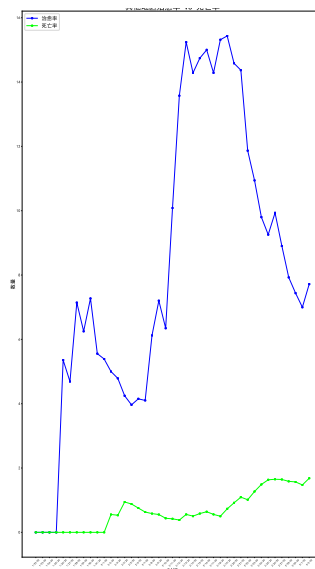


图 4 中国大陆治愈率 VS 死亡率

```

13 plt.xlabel('时间',size = 20)
14 plt.ylabel('数量',size = 20)
15 plt.xticks(rotation = 45,size = 10)
1 plt.yticks(size =15)
1 plt.legend(loc = 'upper left',fontsize = 20)

```

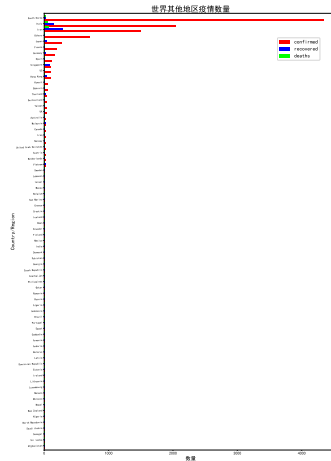


接下来看看其他地区疫情数量。首先还是提出其他地区的数据

```

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['deaths'] = deaths[['last_update']][deaths['Country/Region'] != '
   Mainland China']
4
5 others_countries = others.rename(columns = {last_update:'confirmed'})
6 others_countries = others_countries.set_index('Country/Region')
7 others_countries = others_countries.groupby('Country/Region').sum()
8 #接着画图
9 others_countries.sort_values(by = 'confirmed',ascending = True).plot(
   kind='barh',figsize=(20,30),color = ['red','blue','lime'], width=1,
   rot=2)
10 plt.title('世界其他地区疫情数量', size=30)
11 plt.ylabel('Country/Region',size = 20)
12 plt.xlabel('数量',size = 20)
13 plt.yticks(size=10)
1 plt.xticks(size=15)
1 plt.legend(bbox_to_anchor=(0.95,0.95),fontsize = 20)

```



从图可以看到，韩国，意大利，日本这些地区也有很多新冠肺炎患者。

### 3 绘制疫情地图

这里主要用到两个 python 包，一个是 folium 包，这个包也是笔者最近才发现的绘图包，类似于 R 语言绘图里的 ggplot2，可以添加图层来定义一个 Map 对象，最后以几种方式将 Map 对象展现出来。这里有一个详细教程，感兴趣的可以看看 <https://python-visualization.github.io/folium/>。另一个包就是 plotly 了，这也是一个强大的绘图包，详细教程请看这里 <https://plot.ly/python/plotly-express/>。

首先是 folium 包绘制地图，import folium，只需要导入包就可以了，没下载这个包的记得下载才能使用。我们在前面数据里加入中国大陆的数据，并使用武汉的经纬度。

```

1 import folium
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'})

1 #然后开始正式构建地图
2 #定义一个world_map对象；location的格式为[纬度,经度]；zoom_start表示初始地图的缩放尺寸，
    数值越大放大程度越大；tiles为地图类型，用于控制绘图调用的地图样式，默认为'
    OpenStreetMap'，也有一些其他的内建地图样式，如'Stamen Terrain'、'Stamen Toner'、'
    Mapbox Bright'、'Mapbox Control Room'等；也可以传入'None'来绘制一个没有风格的朴素地
    图，或传入一个URL来使用其它的自选osm。
3 #然后往world_map里添加其他元素，注意这里的for循环和最后的add_to是把经纬度点的信息一个
    一个的加进去
4 world_map = folium.Map(location=[10,-20],zoom_start=2.3,tiles='Stamen
    Toner')
5

```



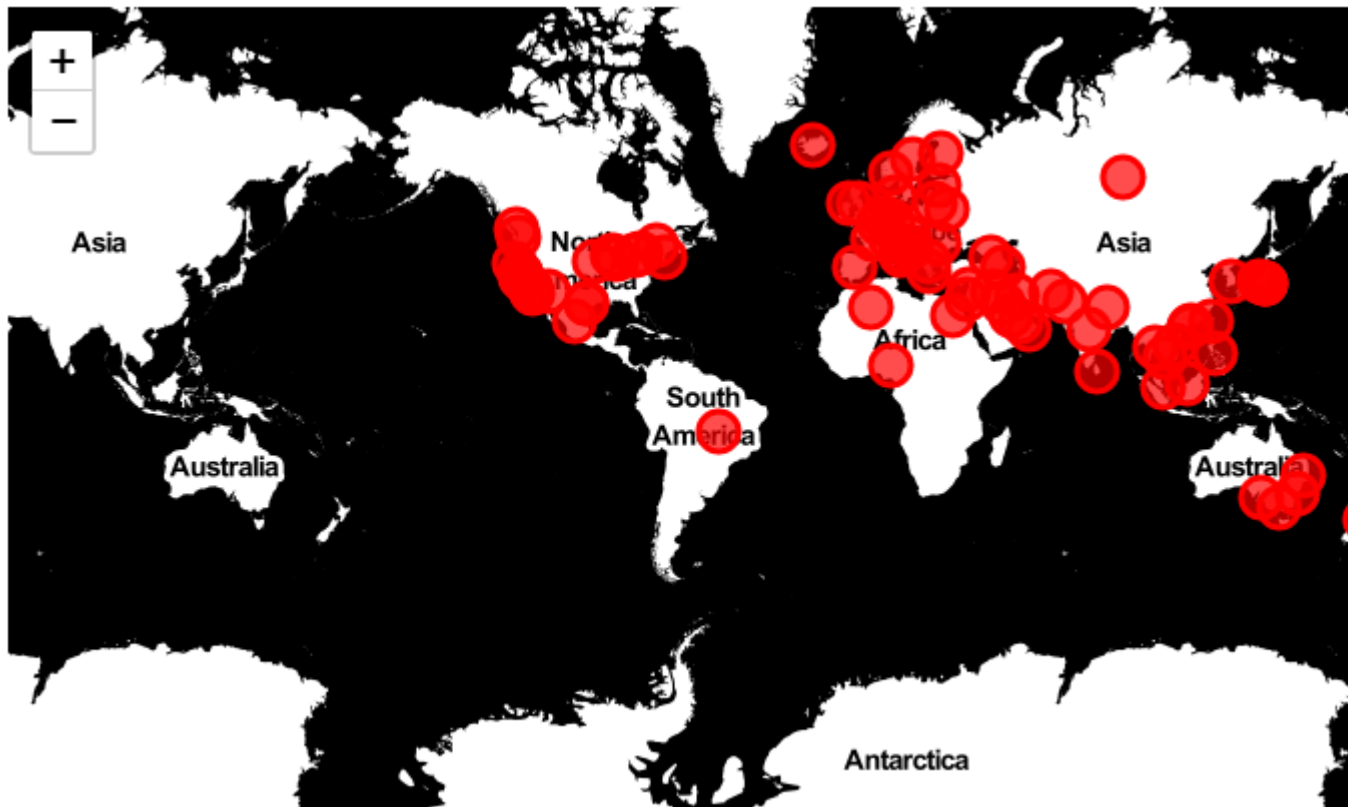


图 5 world\_map

```

6 for lat, lon, value, name in zip(others_countries['Lat'], others_
    countries['Long'], others_countries['confirmed'], others_countries['
    Country/Region']):
7     folium.CircleMarker([lat, lon],
8                           radius=10,
9                           popup = ('<strong>Country</strong>: ' +
10                                str(name).capitalize() + '<br>' +
11                                '<strong>Confirmed Cases</strong>: ' +
12                                str(value) + '<br>'),
13                                color = "red",
14                                fill_color = "red",
15                                fill_opacity = 0.7).add_to(world_map)
1 world_map

```

用 plotly 绘制每日疫情扩散地图

```

1 import plotly.express as px
2 confirmed = confirmed.melt(id_vars = ['Province/State', 'Country/Region'
    , 'Lat', 'Long'], var_name='date', value_name = 'confirmed')
3 confirmed

1 #date列转换成datetime格式的数据
2 confirmed['date_dt'] = pd.to_datetime(confirmed.date, format='%m/%d/%y')
3 confirmed.date = confirmed.date_dt.dt.date
4 confirmed.rename(columns={'Country/Region': 'country', 'Province/State':
    'province'}, inplace=True)
5 confirmed

1 #治愈数据
2 recovered = recovered.melt(id_vars = ['Province/State', 'Country/Region'

```

```

    , 'Lat', 'Long'], var_name='date',value_name = 'recovered')
3 recovered['date_dt'] = pd.to_datetime(recovered.date, format="%m/%d/%y")
4 recovered.date = recovered.date_dt.dt.date
5 recovered.rename(columns={'Country/Region': 'country', 'Province/State':
    'province'}, inplace=True)
6 #死亡数据
7 deaths = deaths.melt(id_vars = ['Province/State', 'Country/Region', 'Lat
    ', 'Long'], var_name='date', value_name = 'deaths')
8 deaths['date_dt'] = pd.to_datetime(deaths.date, format="%m/%d/%y")
9 deaths.date = deaths.date_dt.dt.date
10 deaths.rename(columns={'Country/Region': 'country', 'Province/State': '
    province'}, inplace=True)
11
12 merge_on = ['province', 'country', 'date']
13 all_data = confirmed.merge(deaths[merge_on + ['deaths']], how='left', on
    =merge_on).\
14     merge(recovered[merge_on + ['recovered']], how='left', on=merge_on)
15 all_data

1 Coronavirus_map = all_data.groupby(['date_dt', 'province'])['confirmed',
    'deaths','recovered', 'Lat', 'Long'].max().reset_index()
2 Coronavirus_map['size'] = Coronavirus_map.confirmed.pow(0.5) # 创建实心圆
    大小
3 Coronavirus_map['date_dt'] = Coronavirus_map['date_dt'].dt.strftime('%Y
    -%m-%d')

1 fig = px.scatter_geo(Coronavirus_map, lat='Lat', lon='Long', scope='asia
    ',
2
3     color="size", size='size', hover_name='province',
4     hover_data=['confirmed', 'deaths', 'recovered'],
5     projection="natural earth",animation_frame="date_dt
    ",title='亚洲地区疫情扩散图')
6 fig.update(layout_coloraxis_showscale=False)
1 fig.show()

```

亚洲地区疫情扩散图

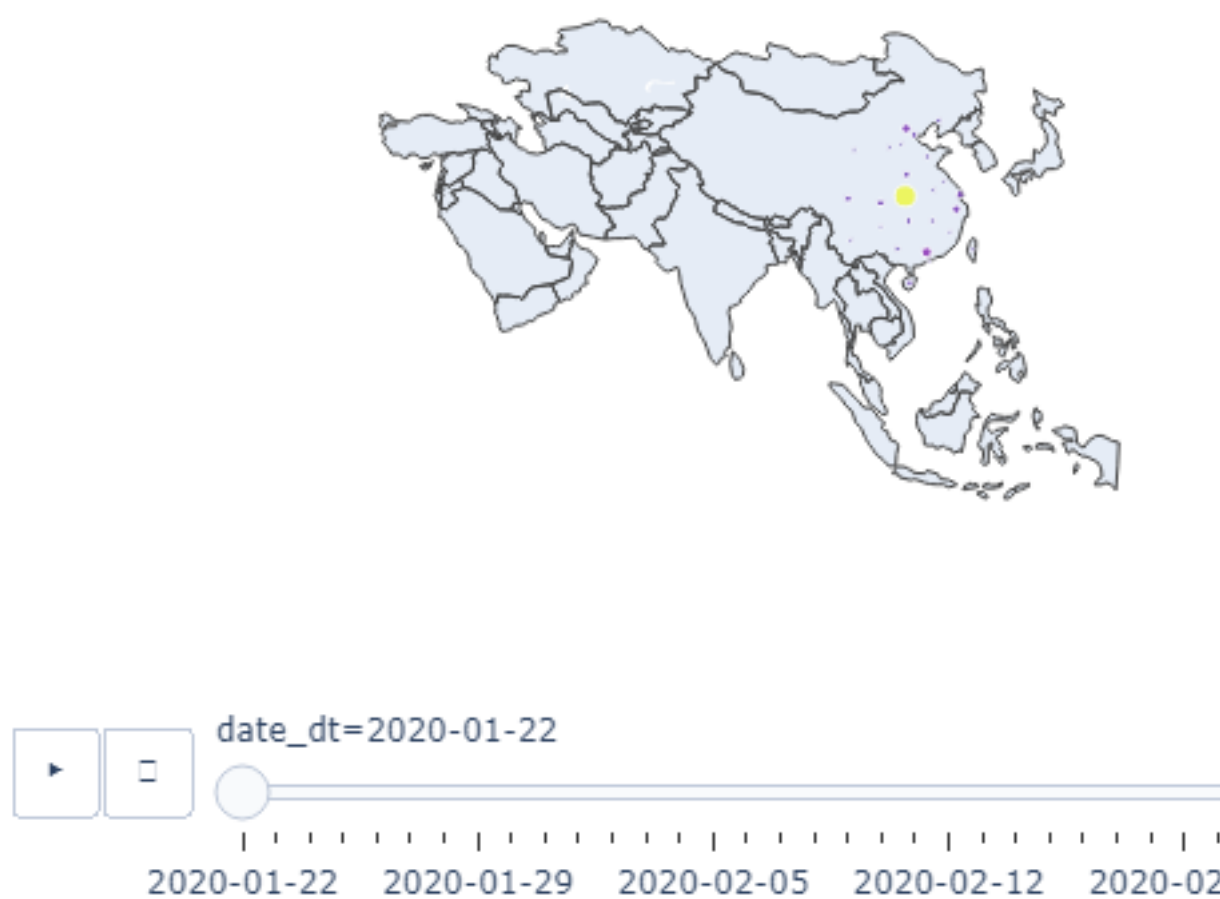


图 6 fig7