全球抗“疫”：用Python带你了解世界疫情

王梦圆

2020-02

# 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。数据主要是是三个文件，包含了疫情的确诊数（confirmed），治愈数（recoved），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。

import numpy as np  
import pandas as pd  
#confirmed = pd.read\_csv('https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse\_covid\_19\_data/csse\_covid\_19\_time\_series/time\_series\_19-covid-Confirmed.csv')  
#recovered = pd.read\_csv('https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse\_covid\_19\_data/csse\_covid\_19\_time\_series/time\_series\_19-covid-Recovered.csv')  
#deaths =pd.read\_csv('https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse\_covid\_19\_data/csse\_covid\_19\_time\_series/time\_series\_19-covid-Deaths.csv')  
confirmed = pd.read\_csv('./time\_series\_19-covid-Confirmed.csv')  
recovered = pd.read\_csv('./time\_series\_19-covid-Recovered.csv')  
deaths = pd.read\_csv('./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)是查看数据前五行；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。

confirmed.head(5)

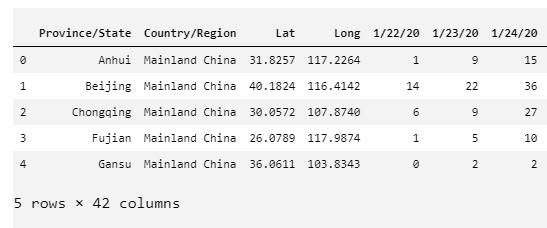


表1

recovered.head(5)

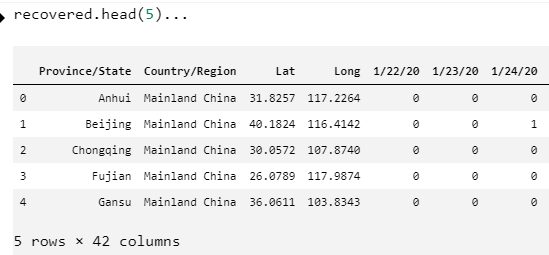


表2

deaths.head(5)

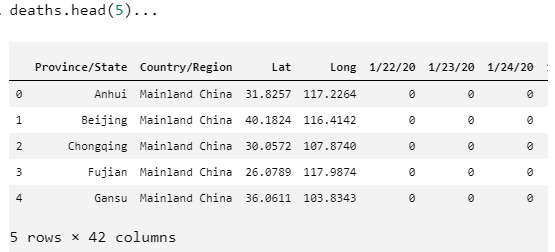


表3

print(confirmed.shape)

print(recovered.shape)

print(deaths.shape)

# 数据可视化

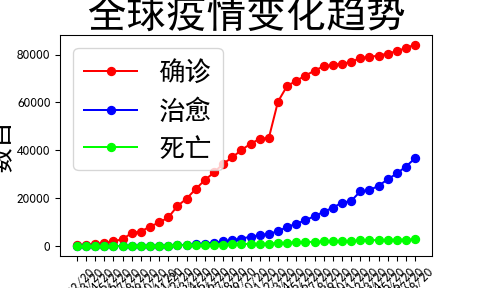
import matplotlib.pyplot as plt  
plt.rcParams['font.sans-serif'] = ['SimHei']#用来正常显示中文标签  
plt.rcParams['axes.unicode\_minus'] = False#用来正常显示负号  
  
countries = confirmed['Country/Region'].unique()  
print(countries)#可以看出一共49个国家/地区都有新冠状肺炎病例  
  
#计算出每日所有地区新冠肺炎的确诊数，治愈数，死亡数

all\_confirmed = np.sum(confirmed.iloc[:,4:])  
all\_recovered = np.sum(recovered.iloc[:,4:])  
all\_deaths = np.sum(deaths.iloc[:,4:])  
all\_confirmed

plt.plot(all\_confirmed,color = 'red',label = '确诊',marker = 'o')  
plt.plot(all\_recovered,color = 'blue',label = '治愈',marker = 'o')  
plt.plot(all\_deaths,color = 'lime',label = '死亡',marker = 'o')  
plt.xticks(rotation = 45,size = 10)

plt.yticks(size = 10)

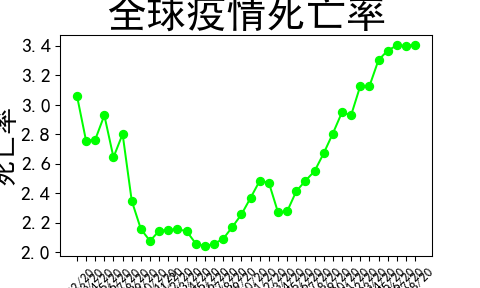
plt.xlabel('时间',size = 20)  
plt.ylabel('数目',size = 20)  
plt.title('全球疫情变化趋势',size = 30)  
plt.legend(loc = 'upper left',fontsize = 20)  
plt.show()  
#可以看出，目前新冠肺炎确诊病例还在持续增加，不过令人高兴的是治愈数也在持续增长，死亡数很少



#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  
death\_rate = (all\_deaths/all\_confirmed)\*100  
plt.plot(death\_rate,color = 'lime',label = '死亡',marker = 'o')  
plt.xticks(rotation = 45,size = 10)

plt.yticks(size = 15)

plt.xlabel('时间',size = 20)  
plt.ylabel('死亡率',size = 20)  
plt.title('全球疫情死亡率',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。  
last\_update = '2/28/20' #设置最新数据日期  
China\_cases = confirmed[['Province/State',last\_update]][confirmed['Country/Region']=='Mainland China']  
China\_cases['recovered'] = recovered[[last\_update]][recovered['Country/Region']=='Mainland China']  
China\_cases['deaths'] = deaths[[last\_update]][deaths['Country/Region']=='Mainland China']  
China\_cases = China\_cases.set\_index('Province/State')  
China\_cases = China\_cases.rename(columns = {last\_update:'confirmed'})  
China\_cases

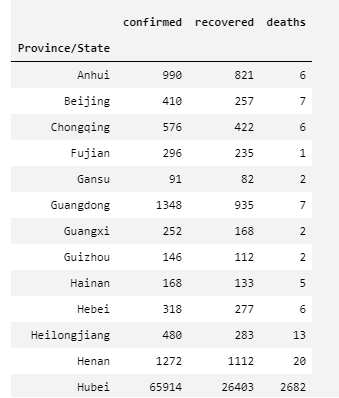
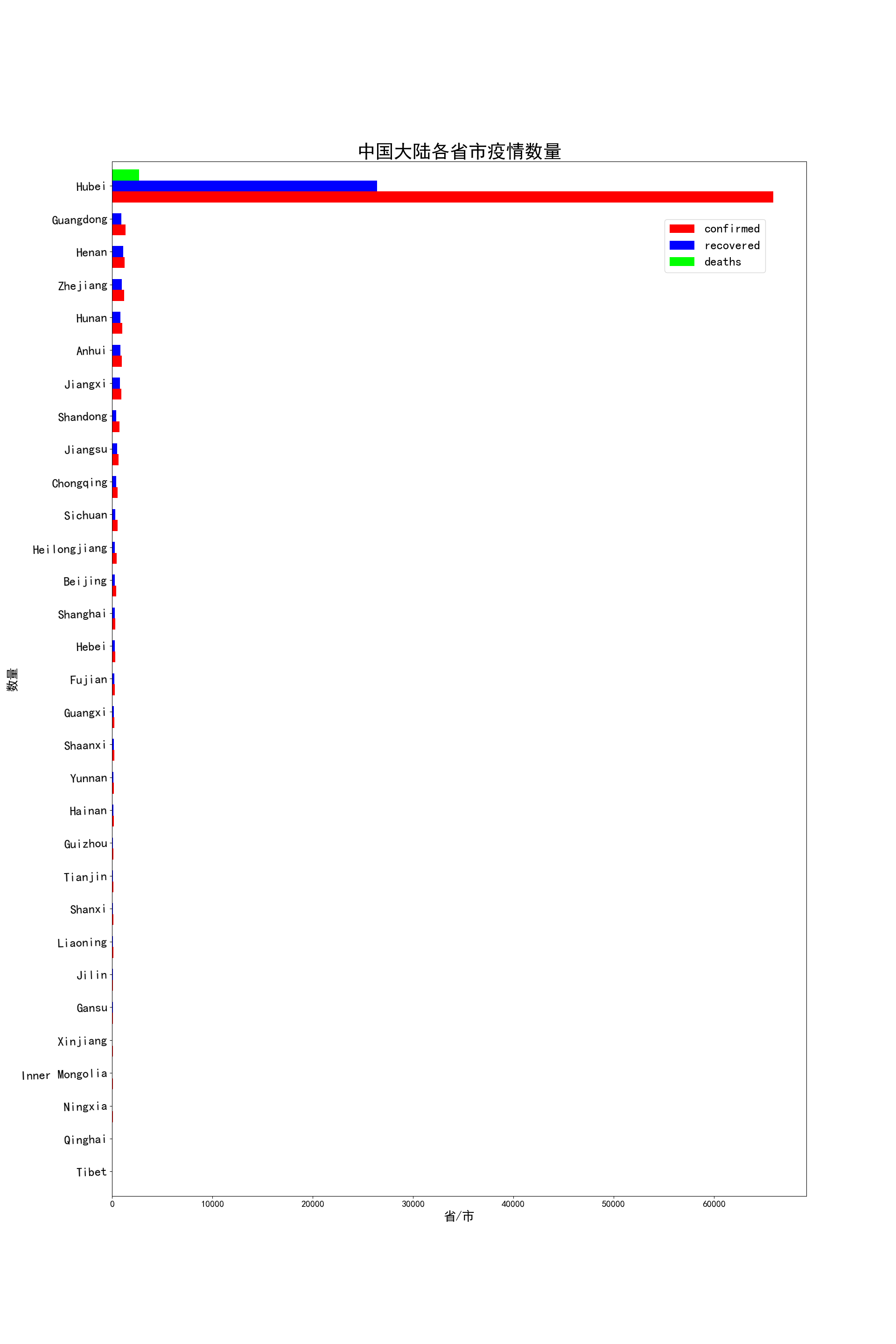


表4

#下面画出中国大陆每个省份的疫情数量图  
Mianland\_China = China\_cases.sort\_values(by='confirmed',ascending=True)  
Mianland\_China.plot(kind='barh',figsize=(20,30),color = ['red','blue','lime'],width = 1,rot = 2)  
plt.title('中国大陆各省市疫情数量',size = 30)  
plt.xlabel('省/市',size = 20)  
plt.ylabel('数量',size = 20)  
plt.xticks(size = 15)

plt.yticks(size = 20)

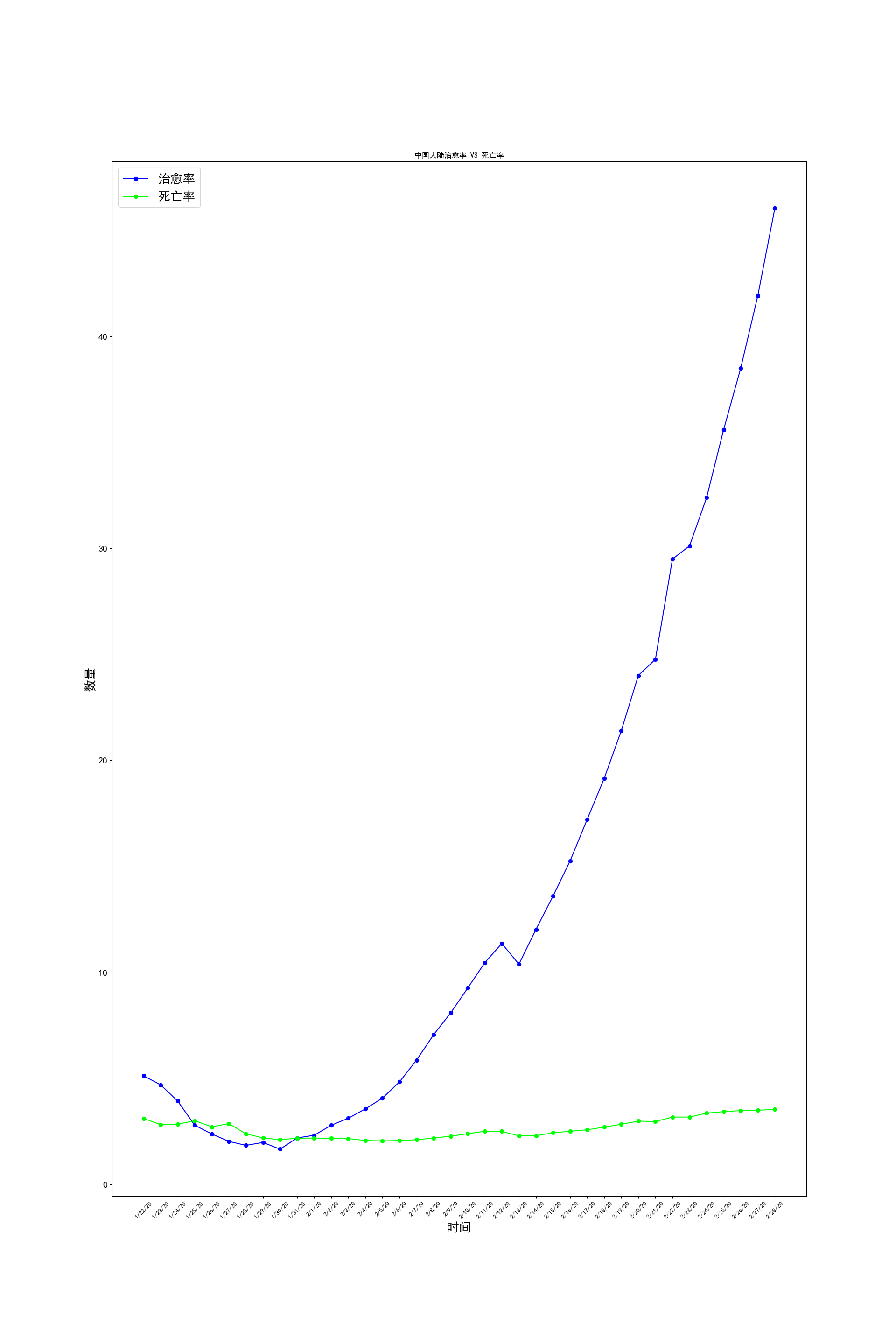
plt.legend(bbox\_to\_anchor = (0.95,0.95),fontsize = 20)  
#可以看到，湖北省三项数据高居第一位，且远远高于其他省份。

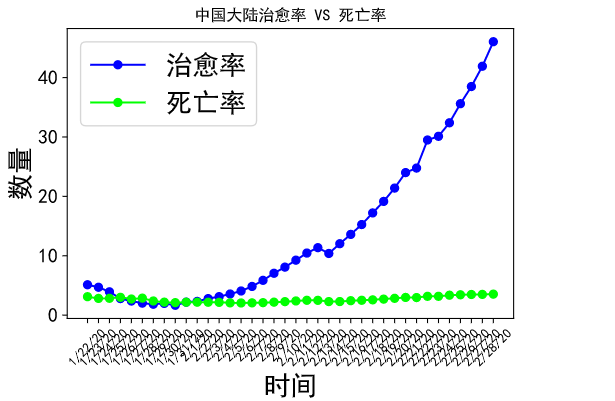


#下面看看中国大陆的治愈率和死亡率数据，数据使用下面的代码即可计算出来，最终结果在recover\_rate和death\_rate里。  
confirmed\_China = confirmed[confirmed['Country/Region']=='Mainland China']  
confirmed\_China = np.sum(confirmed\_China.iloc[:,4:])  
recovered\_China = recovered[recovered['Country/Region']=='Mainland China']  
recovered\_China = np.sum(recovered\_China.iloc[:,4:])  
deaths\_China = deaths[deaths['Country/Region']=='Mainland China']  
deaths\_China = np.sum(deaths\_China.iloc[:,4:])  
recover\_rate = (recovered\_China/confirmed\_China)\*100 #中国地区的治愈率  
deaths\_rate = (deaths\_China/confirmed\_China)\*100#中国各地区的死亡率  
#接下来画图  
plt.plot(recover\_rate,color = 'blue',label = '治愈率',marker = 'o')  
plt.plot(deaths\_rate,color = 'lime',label = '死亡率',marker = 'o')  
plt.title('中国大陆治愈率 VS 死亡率')  
plt.xlabel('时间',size = 20)  
plt.ylabel('数量',size = 20)  
plt.xticks(rotation = 45,size = 10)

plt.yticks(size =15)

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



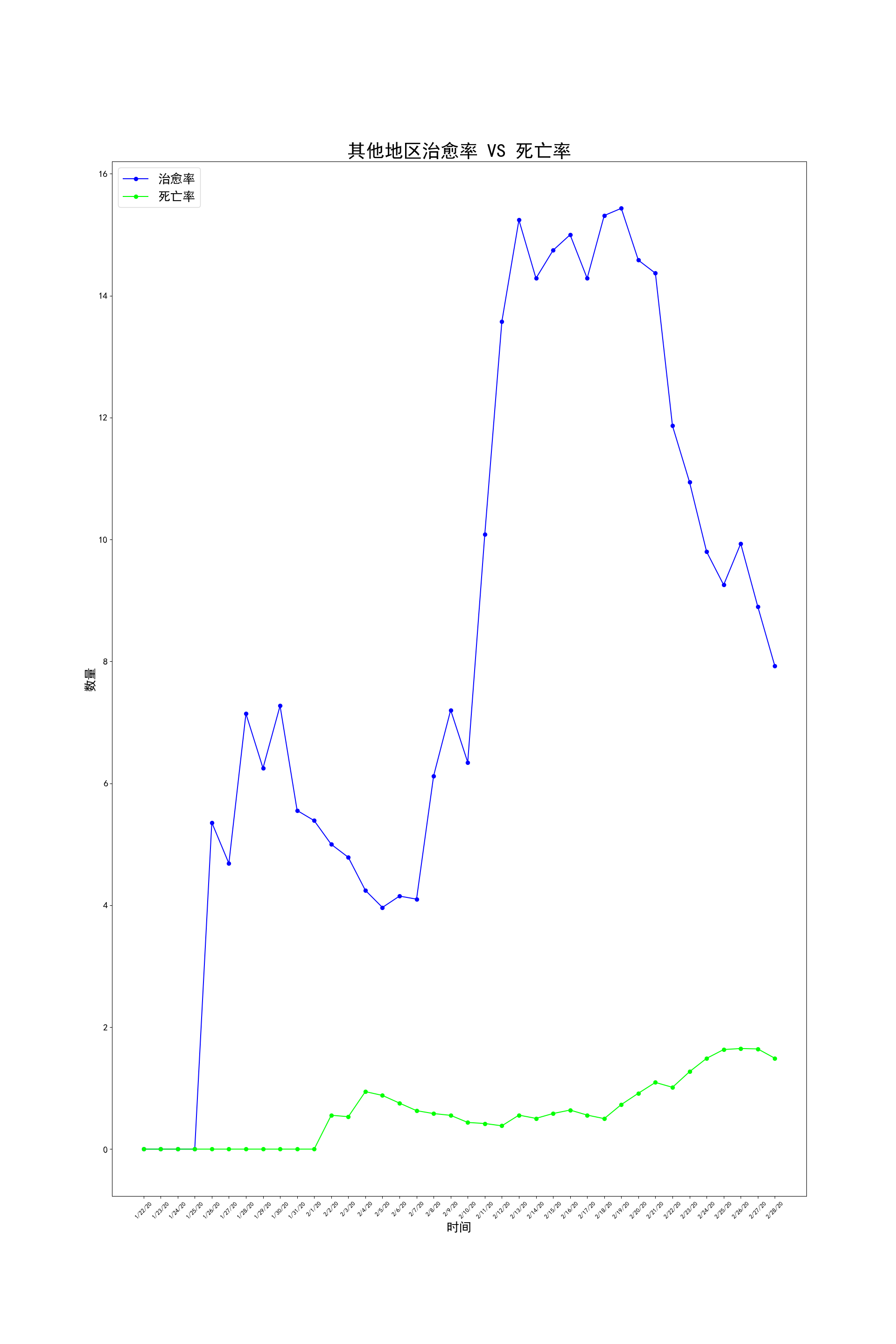
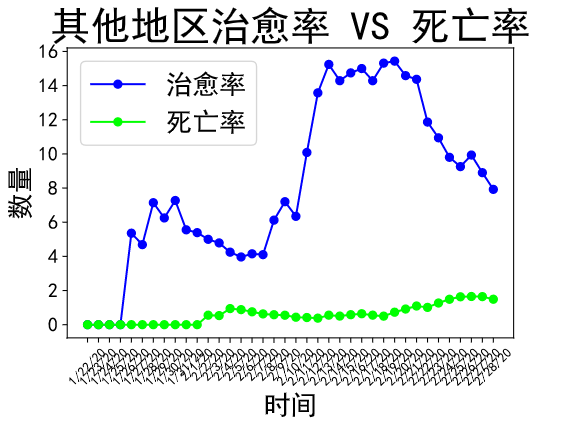


中国大陆治愈率VS死亡率

#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。  
confirmed\_others = confirmed[confirmed['Country/Region'] != 'Mainland China']  
confirmed\_others = np.sum(confirmed\_others.iloc[:,4:])  
recovered\_others = recovered[recovered['Country/Region'] != 'Mainland China']  
recovered\_others = np.sum(recovered\_others.iloc[:,4:])  
deaths\_others = deaths[deaths['Country/Region']!='Mainland China']  
deaths\_others = np.sum(deaths\_others.iloc[:,4:])  
recover\_rate\_others = (recovered\_others/confirmed\_others)\*100 #其他地区的治愈率  
deaths\_rate\_others = (deaths\_others/confirmed\_others)\*100#其他各地区的死亡率  
#接下来画图  
plt.plot(recover\_rate\_others,color = 'blue',label = '治愈率',marker = 'o')  
plt.plot(deaths\_rate\_others,color = 'lime',label = '死亡率',marker = 'o')  
plt.title('其他地区治愈率 VS 死亡率',size = 30)  
plt.xlabel('时间',size = 20)  
plt.ylabel('数量',size = 20)  
plt.xticks(rotation = 45,size = 10)

plt.yticks(size =15)

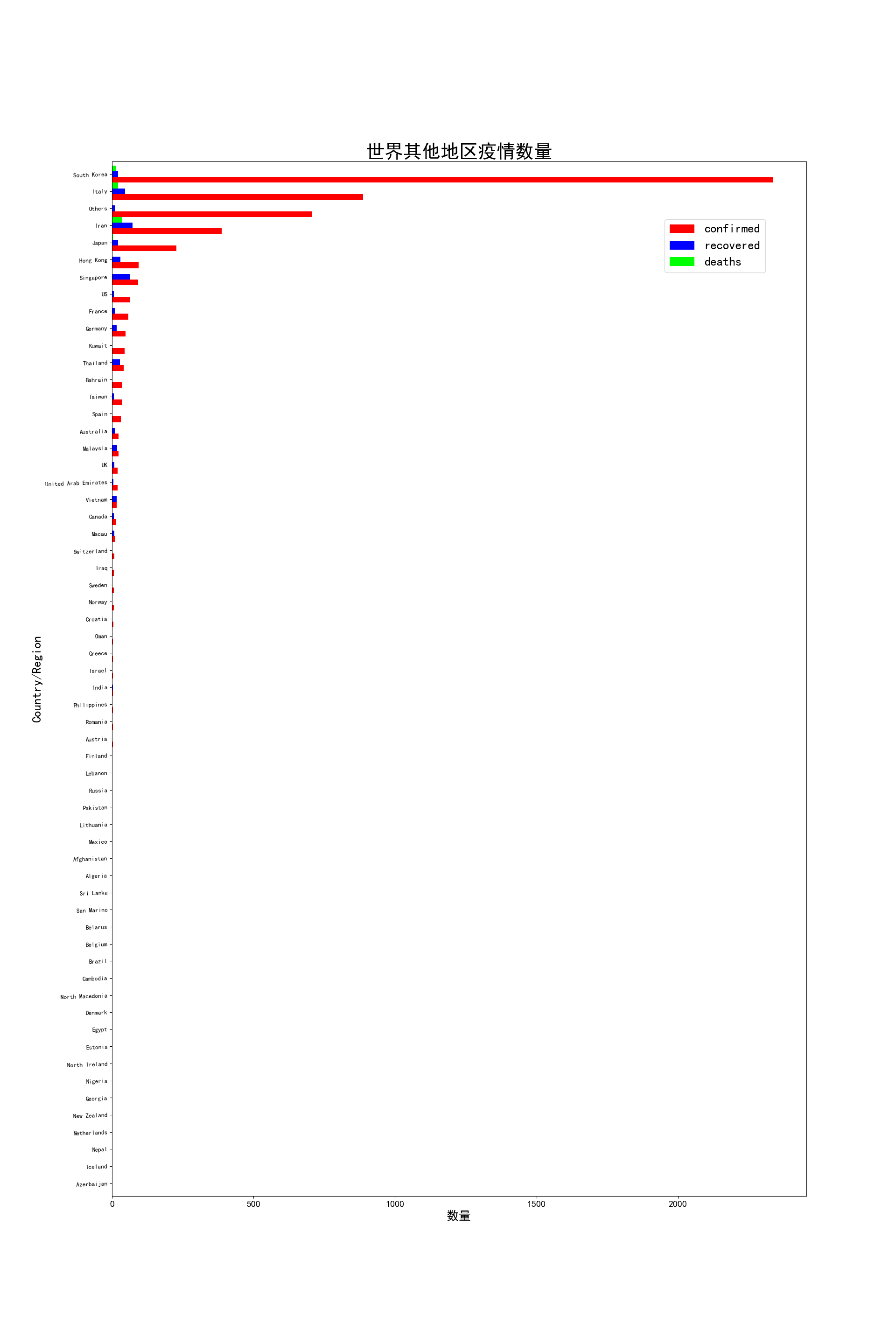
plt.legend(loc = 'upper left',fontsize = 20)

#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  
others = confirmed[['Country/Region',last\_update]][confirmed['Country/Region'] != 'Mainland China']  
others['recovered'] = recovered[[last\_update]][recovered['Country/Region'] != 'Mainland China']  
others['deaths'] = deaths[[last\_update]][deaths['Country/Region'] != 'Mainland China']  
  
others\_countries = others.rename(columns = {last\_update:'confirmed'})  
others\_countries = others\_countries.set\_index('Country/Region')  
others\_countries = others\_countries.groupby('Country/Region').sum()  
#接着画图  
others\_countries.sort\_values(by = 'confirmed',ascending = True).plot(kind='barh',figsize=(20,30),color = ['red','blue','lime'], width=1,rot=2)  
plt.title('世界其他地区疫情数量', size=30)  
plt.ylabel('Country/Region',size = 20)  
plt.xlabel('数量',size = 20)  
plt.yticks(size=10)

plt.xticks(size=15)

plt.legend(bbox\_to\_anchor=(0.95,0.95),fontsize = 20)  
plt.show()  
#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。



# 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。

import folium  
others = confirmed[['Country/Region','Lat','Long',last\_update]][confirmed['Country/Region'] != 'Mainland China']  
others['recovered'] = recovered[[last\_update]][recovered['Country/Region'] != 'Mainland China']  
others['death'] = deaths[[last\_update]][deaths['Country/Region'] != 'Mainland China']  
others\_countries = others.rename(columns = {last\_update:'confirmed'})

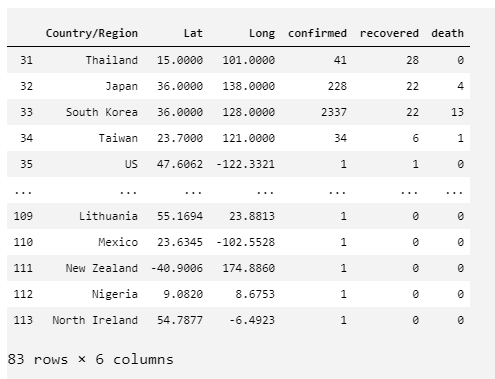
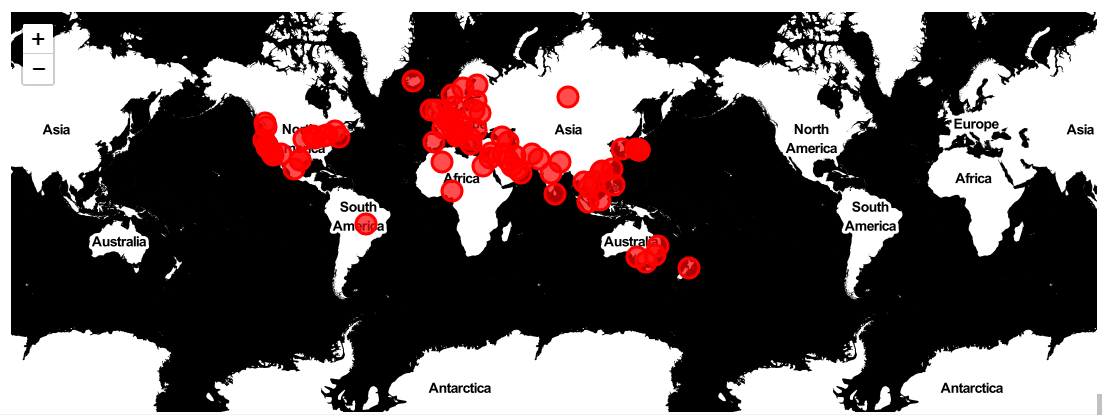


表5

#然后开始正式构建地图  
#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。  
#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  
world\_map = folium.Map(location=[10,-20],zoom\_start=2.3,tiles='Stamen Toner')  
  
for lat, lon, value, name in zip(others\_countries['Lat'],others\_countries['Long'],others\_countries['confirmed'],others\_countries['Country/Region']):  
 folium.CircleMarker([lat, lon],  
 radius=10,  
 popup = ('<strong>Country</strong>: ' + str(name).capitalize()+'<br>' '<strong>Confirmed Cases</strong>: ' + str(value) + '<br>'),  
 color = "red",  
 fill\_color = "red",  
 fill\_opacity = 0.7).add\_to(world\_map)

world\_map



world\_map

用plotly绘制每日疫情扩散地图

import plotly.express as px  
confirmed = confirmed.melt(id\_vars = ['Province/State', 'Country/Region', 'Lat', 'Long'], var\_name='date',value\_name = 'confirmed')  
confirmed

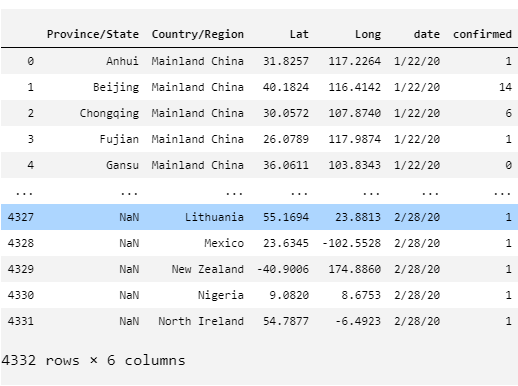


表6

#date列转换成datetime格式的数据  
confirmed['date\_dt'] = pd.to\_datetime(confirmed.date,format='%m/%d/%y')  
confirmed.date = confirmed.date\_dt.dt.date  
confirmed.rename(columns={'Country/Region':'country','Province/State':'province'},inplace=True)  
confirmed

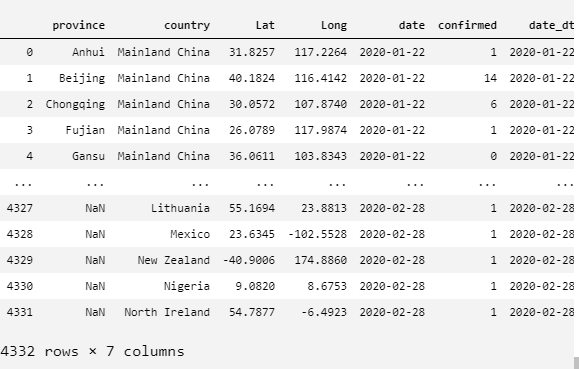


表7

#治愈数据  
recovered = recovered.melt(id\_vars = ['Province/State', 'Country/Region', 'Lat', 'Long'], var\_name='date',value\_name = 'recovered')  
recovered['date\_dt'] = pd.to\_datetime(recovered.date, format="%m/%d/%y")  
recovered.date = recovered.date\_dt.dt.date  
recovered.rename(columns={'Country/Region': 'country', 'Province/State': 'province'}, inplace=True)  
#死亡数据  
deaths = deaths.melt(id\_vars = ['Province/State', 'Country/Region', 'Lat', 'Long'], var\_name='date', value\_name = 'deaths')  
deaths['date\_dt'] = pd.to\_datetime(deaths.date, format="%m/%d/%y")  
deaths.date = deaths.date\_dt.dt.date  
deaths.rename(columns={'Country/Region': 'country', 'Province/State': 'province'}, inplace=True)  
  
merge\_on = ['province', 'country', 'date']  
all\_data = confirmed.merge(deaths[merge\_on + ['deaths']], how='left', on=merge\_on).\  
 merge(recovered[merge\_on + ['recovered']], how='left', on=merge\_on)  
all\_data

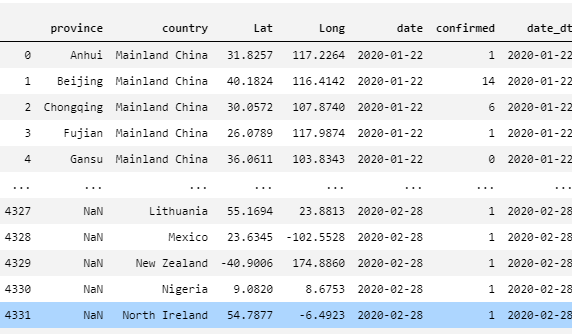


表8

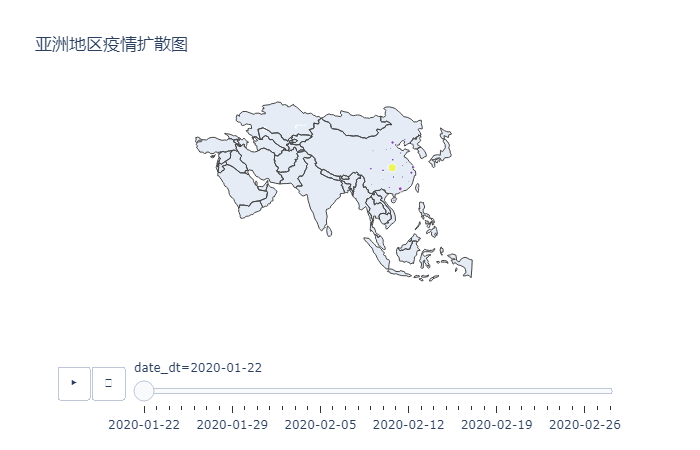
Coronavirus\_map = all\_data.groupby(['date\_dt', 'province'])['confirmed', 'deaths','recovered', 'Lat', 'Long'].max().reset\_index()  
Coronavirus\_map['size'] = Coronavirus\_map.confirmed.pow(0.5) # 创建实心圆大小  
Coronavirus\_map['date\_dt'] = Coronavirus\_map['date\_dt'].dt.strftime('%Y-%m-%d')



表9

fig = px.scatter\_geo(Coronavirus\_map, lat='Lat', lon='Long', scope='asia',  
 color="size", size='size', hover\_name='province',  
 hover\_data=['confirmed', 'deaths', 'recovered'],  
 projection="natural earth",animation\_frame="date\_dt",title='亚洲地区疫情扩散图')  
fig.update(layout\_coloraxis\_showscale=False)

fig.show()



亚洲地区疫情扩散图