The World Epidemic Analysis

wanghonghong

2020-01

1 数据处理及读取

```
import pandas as pd
import numpy as np
```

接下来导入数据，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），基本上每日会更新最新疫情数据。以下数据更新至 2020 年 2 月 28 日。

```
path='./data/COVID-19/csse_covid_19_data/csse_covid_19_time_series/'
confirmed = pd.read_csv(path+'time_series_19-covid-Confirmed.csv')
recovered = pd.read_csv(path+'time_series_19-covid-Recovered.csv')
deaths = pd.read_csv(path+'time_series_19-covid-Deaths.csv')
```

```
print(confirmed.shape)
```

```
print(recovered.shape)
```

```
print(deaths.shape)
```

```
confirmed.head()
```

Province/State	Country/Region	Lat	Long	1/22/20	...	2/28/20
Anhui	Mainland China	31.8257	117.2264	1	...	990
Beijing	Mainland China	40.1824	116.4142	14	...	410
Chongqing	Mainland China	30.0572	107.8740	6	...	576
Fujian	Mainland China	26.0789	117.9874	1	...	296
Gansu	Mainland China	36.0611	103.8343	0	...	91

```
recovered.head()
```

Province/State	Country/Region	Lat	Long	1/22/20	...	2/28/20
Anhui	Mainland China	31.8257	117.2264	0	...	821
Beijing	Mainland China	40.1824	116.4142	0	...	257
Chongqing	Mainland China	30.0572	107.8740	0	...	422
Fujian	Mainland China	26.0789	117.9874	0	...	235
Gansu	Mainland China	36.0611	103.8343	0	...	82

```
deaths.head()
```

Province/State	Country/Region	Lat	Long	1/22/20	...	2/28/20
Anhui	Mainland China	31.8257	117.2264	0	...	6
Beijing	Mainland China	40.1824	116.4142	0	...	7
Chongqing	Mainland China	30.0572	107.8740	0	...	6
Fujian	Mainland China	26.0789	117.9874	0	...	1
Gansu	Mainland China	36.0611	103.8343	0	...	2

2 数据可视化

2.1 世界范围

```
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
```

首先看看哪些地区发生了疫区，一共有 75 个地区都有肺炎疫情。

```
countries = confirmed['Country/Region'].unique()
print(countries)
```

```
print(len(countries))
```

接下来看看世界疫情发展趋势，在此之前，需要计算出每日所有地区新冠肺炎的确诊人数，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.head()
```

接下来就可以画出疫情发展趋势图了。

```
import matplotlib.ticker as tkr
fig,ax = plt.subplots()
ax.plot([i[:-3] for i in all_confirmed.index],all_confirmed.values,color
    ='red',label='确诊',marker='o',markersize=2)
ax.plot([i[:-3] for i in all_confirmed.index],all_recovered.values,color
    ='blue',label='治愈',marker='x',markersize=2)
ax.plot([i[:-3] for i in all_confirmed.index],all_deaths.values,color='
    lime',label='死亡',marker='*',markersize=2)
ax.xaxis.set_major_locator(tkr.MultipleLocator(2.0))
ax.xaxis.set_minor_locator(tkr.MultipleLocator(1.0))
##set the str format of major ticker of X axis
#ax.xaxis.set_major_formatter(tkr.FormatStrFormatter('%m/%d/%y'))
#ax.xaxis.set_ticklabels([i[:-3] for i in all_confirmed.index[::2]],rotation=45)
#all_confirmed.index[::2]#取出奇数位置的元素
plt.xticks(rotation=45)
```

```
plt.yticks()
```

```
ax.set(xlabel='时间',ylabel='数目')
```

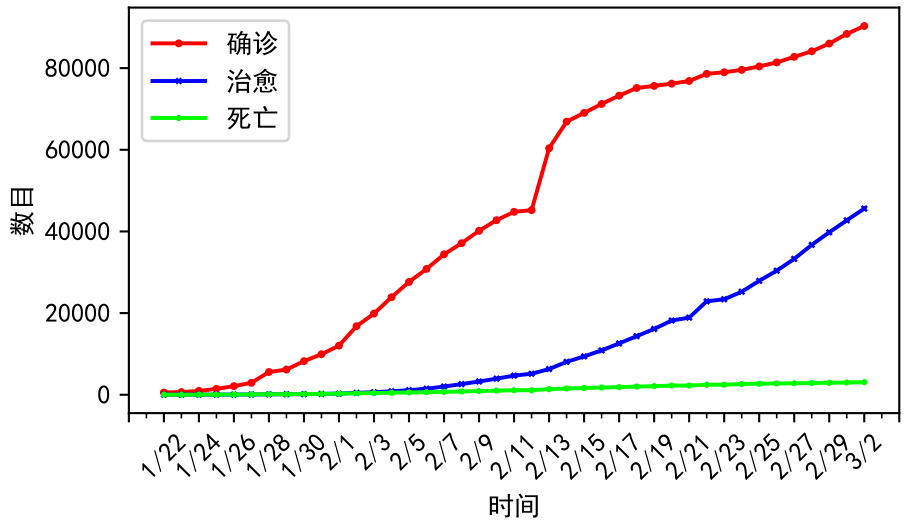

图 1　全球疫情变化趋势

```
plt.legend(loc='upper left',fontsize = 10)
plt.tight_layout()
plt.show()
```

可以看出，目前新冠肺炎确诊病例还在持续增加，不过治愈人数也在持续增长，死亡率很少。

另外确诊人数在 2 月 13 日那天有较大幅度的增长。

下面看看新冠肺炎的死亡率，首先计算死亡率数据，然后再画图。

```
death_rate=(all_deaths/all_confirmed)
import matplotlib.ticker as tkr
fig,ax=plt.subplots()
ax.plot([i[:-3] for i in death_rate.index],death_rate.values,color='lime
    ',label='死亡率',marker='o',markersize=3)
ax.xaxis.set_major_locator(tkr.MultipleLocator(2.0))
ax.xaxis.set_minor_locator(tkr.MultipleLocator(1.0))
## set the str format of major ticker of X axis
#ax.xaxis.set_major_formatter(tkr.FormatStrFormatter('%m/%d'))
#ax.xaxis.set_ticklabels([i[:-3] for i in death_rate.index[::2]],rotation=45)
plt.xticks(rotation=45)
```

```
plt.yticks()
```

```
ax.set(xlabel='时间',ylabel='死亡率')
#plt.title('全球疫情死亡率',size=30)
```

```
plt.tight_layout()
plt.show()
```

2.2 中国大陆

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.

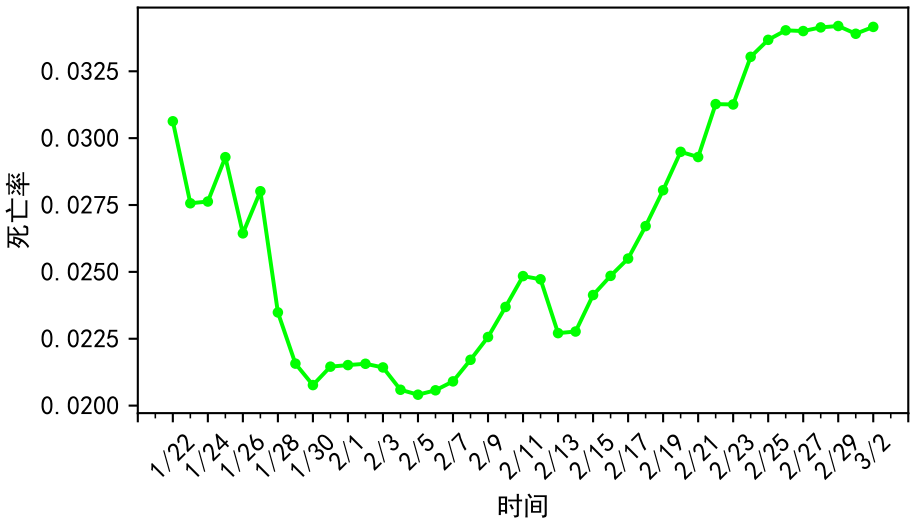

图 2　全球疫情死亡率

2.2.1　数据提取

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 = confirmed.columns[-1] # 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.head()
```

Province/State	confirmed	recovered	deaths
Anhui	990	821	6
Beijing	410	357	7
Chongqing	576	422	6
Fujian	296	235	1
Gansu	91	82	2

2.2.2　中国大陆各个省份确诊数，治愈数，死亡数条形图

下面画出中国大陆每个省份的疫情数量图。

```
#'Country/Region'
fig,ax=plt.subplots()
Mainland_China = China_cases.sort_values(by = 'confirmed',ascending =
    True)
Mainland_China.plot(kind = 'barh',color=['red','blue','lime'],figsize
    =(20,30),ax=ax)
plt.xticks(fontsize=35)
```



```
plt.yticks(fontsize=35)
```

图 3　全国各省市疫情数量

```
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。

2.2.3 中国大陆治愈率，死亡率折线图

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 recovered_rate 和 deaths_rate 里。

```
confirmed_China = confirmed[confirmed['Country/Region'] == 'Mainland China']
#得到中国大陆每个省份每日的确诊人数
confirmed_China = np.sum(confirmed_China.iloc[:,4:])
#得到中国大陆每日共有的确诊人数
recovered_China = recovered[recovered['Country/Region'] == 'Mainland China']
#得到中国大陆每个省份每日的治愈人数
recovered_China = np.sum(recovered_China.iloc[:,4:])
#得到中国大陆每日共有的治愈人数
deaths_China = deaths[deaths['Country/Region'] == 'Mainland China']
#得到中国大陆每个省份每日的死亡人数
deaths_China = np.sum(deaths_China.iloc[:,4:])
#得到中国大陆每日共有的死亡人数
recovered_rate = (recovered_China/confirmed_China)*100
deaths_rate = (deaths_China/confirmed_China)*100
```

接下来画图，显示中国大陆每日的治愈率和死亡率。

```
fig,ax=plt.subplots()
ax.plot([i[:-3] for i in deaths_rate.index],recovered_rate.values,color='blue',label='治愈率',marker='o',markersize=3)
ax.plot([i[:-3] for i in deaths_rate.index],deaths_rate.values,color='lime',label='死亡率',marker='o',markersize=3)
ax.xaxis.set_major_locator(tkr.MultipleLocator(2.0))
ax.xaxis.set_minor_locator(tkr.MultipleLocator(1.0))
## set the str format of major ticker of X axis
#ax.xaxis.set_major_formatter(tkr.FormatStrFormatter('%m/%d/%y'))
#ax.xaxis.set_ticklabels([i[:-3] for i in deaths_rate.index[::2]],rotation=45)
#plt.title('中国大陆治愈率 VS 死亡率',size=30)
ax.set(xlabel='时间',ylabel='数量')
#plt.ylabel("数量")
#plt.xlabel('时间')
```

```
plt.xticks(rotation=45)
```

```
plt.yticks()
```

```
plt.legend(loc = "upper left",fontsize = 10)
plt.tight_layout()
plt.show()
```

虽然在 1 月 25 日到 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。

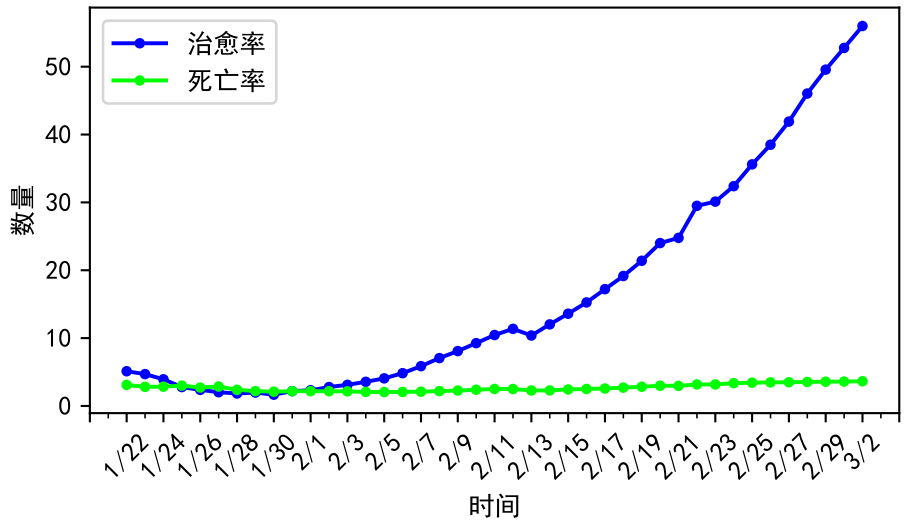

图 4　中国大陆治愈率 VS 死亡率

2.3 除中国大陆以后世界其他地区

希望能够了解世界地区各国的疫情发展情况。

2.3.1　数据提取

首先从全部数据中提取出世界其他地区的数据。里面包含了每个地区最新的确诊数，治愈数，死亡数。

```
others_cases = confirmed[['Country/Region',last_update]][confirmed['
    Country/Region'] != 'Mainland China']
others_cases['recovered'] = recovered[[last_update]][recovered['Country/
    Region'] != 'Mainland China']
others_cases['deaths'] = deaths[[last_update]][deaths['Country/Region']
    != 'Mainland China']
others_cases = others_cases.set_index('Country/Region')
others_cases = others_cases.rename(columns = {last_update:'confirmed'})
others_countries = others_cases.groupby('Country/Region').sum()
#数据是按地区给的，需要用分组聚合统计每个国家的疫情数据
others_countries.head()
```

Country/Region	Lat	long	confirmed	recovered	deaths
Thailand	15.0000	101.0000	41	28	0
Japan	36.0000	138.0000	228	22	4
South Korea	36.0000	128.0000	2337	22	13
Taiwan	23.7000	121.0000	34	6	1
US	47.6062	-122.3321	1	1	0

2.3.2　其他国家确诊数，治愈数，死亡数条形图

```
fig,ax=plt.subplots()
others_countries.sort_values(by='confirmed',ascending=True).plot(kind='
    barh',figsize=(20,30),color=['red','blue','lime'],width=1,rot=2,ax=
    ax)
ax.set_xlabel('数量',fontsize = 35)
```

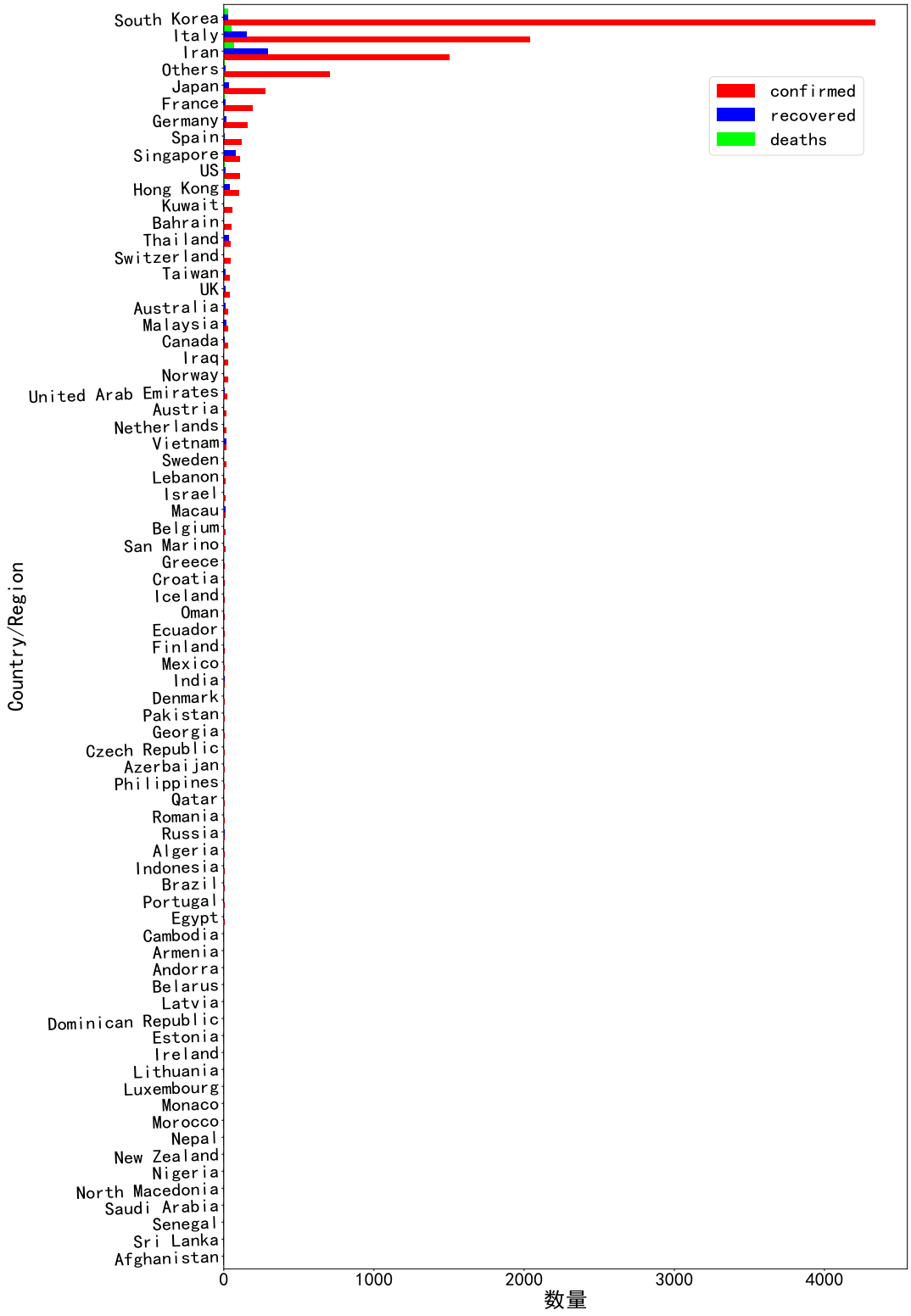

图 5　世界其他地区疫情数量

情况最为严重，确诊人数最多，治愈人数却很少。

2.3.3 其他国家治愈率，死亡率折线图

```
confirmed_others = confirmed[confirmed['Country/Region']!= 'Mainland China']
#世界其他地区每日的确诊人数
confirmed_others = np.sum(confirmed_others.iloc[:,4:])
#世界其他地区每日的确诊人数总和
recovered_others = recovered[recovered['Country/Region']!= 'Mainland China']
#世界其他地区每日的治愈人数
recovered_others = np.sum(recovered_others.iloc[:,4:])
#世界其他地区每日的治愈人数总和
deaths_others = deaths[deaths['Country/Region'] != 'Mainland China']
#世界其他地区每日的死亡人数
deaths_others = np.sum(deaths_others.iloc[:,4:])
#世界其他地区每日的死亡人数总和
recover_rate = (recovered_others/confirmed_others)*100
death_rate = (deaths_others/confirmed_others)*100
```

```
fig,ax=plt.subplots()
ax.plot([i[:-3] for i in death_rate.index],recover_rate.values,color='blue',label='治愈率',marker='o',markersize=3)
ax.plot([i[:-3] for i in death_rate.index],death_rate.values,color='lime',label='死亡率',marker='o',markersize=3)
ax.xaxis.set_major_locator(tkr.MultipleLocator(2.0))
ax.xaxis.set_minor_locator(tkr.MultipleLocator(1.0))
## set the str format of major ticker of X axis
#ax.xaxis.set_major_formatter(tkr.FormatStrFormatter('%m/%d/%y'))
#ax.xaxis.set_ticklabels([i[:-3] for i in death_rate.index[::2]],rotation=45)
#plt.title('世界其他地区治愈率 VS 死亡率',size=30)
ax.set(ylabel='数量',xlabel='时间')
```

```
plt.xticks(rotation=45)
```

```
plt.yticks()
```

```
plt.legend(loc = "upper left",fontsize = 10)
plt.tight_layout()
plt.show()
```

中国大陆的治愈率超过世界其他地区

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/。

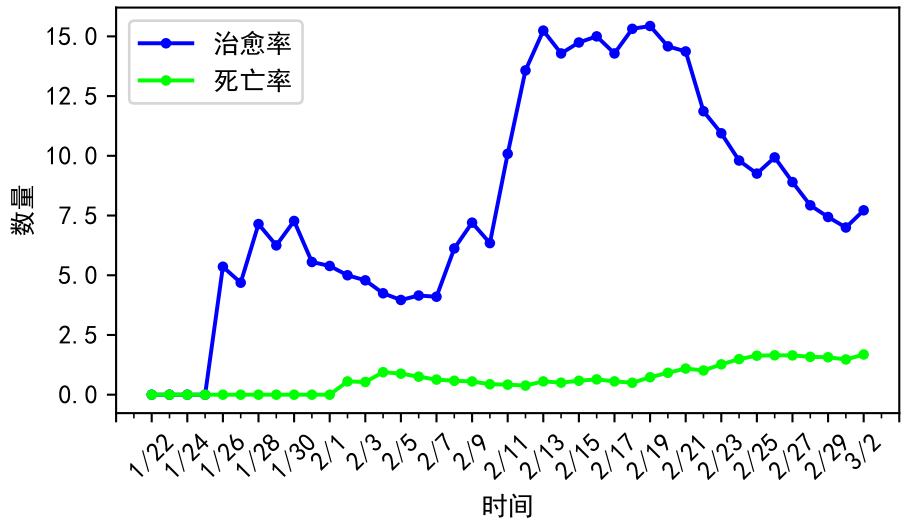

图 6　世界其他地区治愈率 VS 死亡率

(#fig:fig8,)

3.1 folium 包绘制地图

首先是 folium 包绘制地图，导入 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'})
others_countries.loc['94'] = ['Mainland China',30.9756,112.2707,
    confirmed_China[-1],recovered_China[-1],deaths_China[-1]]
#confirmed_China[-1]是中国大陆2月28日的确诊人数,把巴基斯坦的数据更换为中国大陆数
#据。
#others_countries.head().to_csv('./results/other_countries.csv')
```

表 6　各个国家疫情发展情况

X	Country.Region	Lat	Long	confirmed	recovered	death
31	Thailand	15.00	101.0	41	28	0
32	Japan	36.00	138.0	228	22	4
33	South Korea	36.00	128.0	2337	22	13
34	Taiwan	23.70	121.0	34	6	1
35	US	47.61	-122.3	1	1	0

然后开始正式构建地图

```
world_map = folium.Map(location=[10, -20], zoom_start=2.3,tiles='Stamen
    Toner')
```

上面一行是定义一个 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

图 7　全球疫情分布图

```
                                popup = ('<strong>Country</strong>: ' + str(
    name).capitalize() + '<br>'
                                '<strong>Confirmed Cases</strong>: ' + str(
    value) + '<br>'),
                                color='red',
                                fill_color='red',
                                fill_opacity=0.3 ).add_to(world_map)
```

```
world_map
#world_map.save('./results/world_map.html')
#import webbrowser
#webbrowser.open('world_map.html')
```

这里主要说下 popup 参数 popup：str 型或 folium.Popup() 对象输入，用于控制标记部件的具体样式（folium 内部自建了许多样式），默认为 None，即不显示部件。代码使用的是自定义的网页样式，其中表示加粗，表示换行，以便将各个数据显示出来。

然后再运行 world_map，即可出现下面的地图样式，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。

3.2 用 plotly 绘制每日疫情扩散地图。

```
import plotly.express as px
```

如果想绘制每日疫情扩散地图，还需要增加一列，里面记录了每天的日期，因此我们的数据还需要再重新整理下，这里需要用的 melt 函数，它将列名转换为列数据（columns name → column values），重构 DataFrame，

```
confirmed = confirmed.melt(id_vars = ['Province/State','Country/Region',
    'Lat', 'Long'],var_name='date',value_name = 'confirmed')
```

主要参数说明 id_vars: 不需要被转换的列名。value_vars: 需要转换的列名，如果剩下的列全部都要转换，就不用写了。var_name 和 value_name 是自定义设置对应的列名。

还需要把 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_part=confirmed.head().to_csv('./results/confirmed_part.csv')
```

表 7　各个地区疫情确诊情况

X	Province.State	Country.Region	Lat	Long	date	confirmed
0	Anhui	Mainland China	31.83	117.2	1/22/20	1
1	Beijing	Mainland China	40.18	116.4	1/22/20	14
2	Chongqing	Mainland China	30.06	107.9	1/22/20	6
3	Fujian	Mainland China	26.08	118.0	1/22/20	1
4	Gansu	Mainland China	36.06	103.8	1/22/20	0

同时整理出治愈数据和死亡数据

```
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)
#recovered_part=recovered.head().to_csv('./results/recovered_part.csv')

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)
#deaths_part=deaths.head().to_csv('./results/deaths_part.csv')
```

表 8　各个地区疫情治愈情况

X	province	country	Lat	Long	date	recovered	date_dt
0	Anhui	Mainland China	31.83	117.2	2020-01-22	0	2020-01-22
1	Beijing	Mainland China	40.18	116.4	2020-01-22	0	2020-01-22
2	Chongqing	Mainland China	30.06	107.9	2020-01-22	0	2020-01-22
3	Fujian	Mainland China	26.08	118.0	2020-01-22	0	2020-01-22
4	Gansu	Mainland China	36.06	103.8	2020-01-22	0	2020-01-22

表 9　各个地区疫情死亡情况

X	province	country	Lat	Long	date	deaths	date_dt
0	Anhui	Mainland China	31.83	117.2	2020-01-22	0	2020-01-22

1	Beijing	Mainland China	40.18	116.4	2020-01-22	0	2020-01-22
2	Chongqing	Mainland China	30.06	107.9	2020-01-22	0	2020-01-22
3	Fujian	Mainland China	26.08	118.0	2020-01-22	0	2020-01-22
4	Gansu	Mainland China	36.06	103.8	2020-01-22	0	2020-01-22

现在三种数据都有了，我们把它们合并在一张表里面，主要用到 merge 函数

```
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.head()
```

```
confirmed.shape,recovered.shape,deaths.shape
```

由于要演示的是疫情扩散地图，因此笔者用实心圆来表示每个地区的疫情变化，而实心圆的大小则代表了三种数据的大小，所以在我们的数据里要加一列，使用 confirmed 数据的二分之一次方来表示实心圆的大小。

```
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')
#Coronavirus_map_part=Coronavirus_map.head().to_csv('./results/Coronavirus_map_part.csv
    ')
```

表 10　各个地区疫情发展情况

X	date_dt	province	confirmed	deaths	recovered	Lat	Long	size
0	2020-01-22	Montreal, QC	0	0	0	45.50	-73.57	0.000
1	2020-01-22	Anhui	1	0	0	31.83	117.23	1.000
2	2020-01-22	Beijing	14	0	0	40.18	116.41	3.742
3	2020-01-22	Boston, MA	0	0	0	42.36	-71.06	0.000
4	2020-01-22	British Columbia	0	0	0	49.28	-123.12	0.000

最后就是绘图部分，代码也很简单，如果有不懂得参数可以使用 help(px.scatter_geo) 来查看每个参数用法

```
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()
```

图 8　亚洲地区疫情扩散图