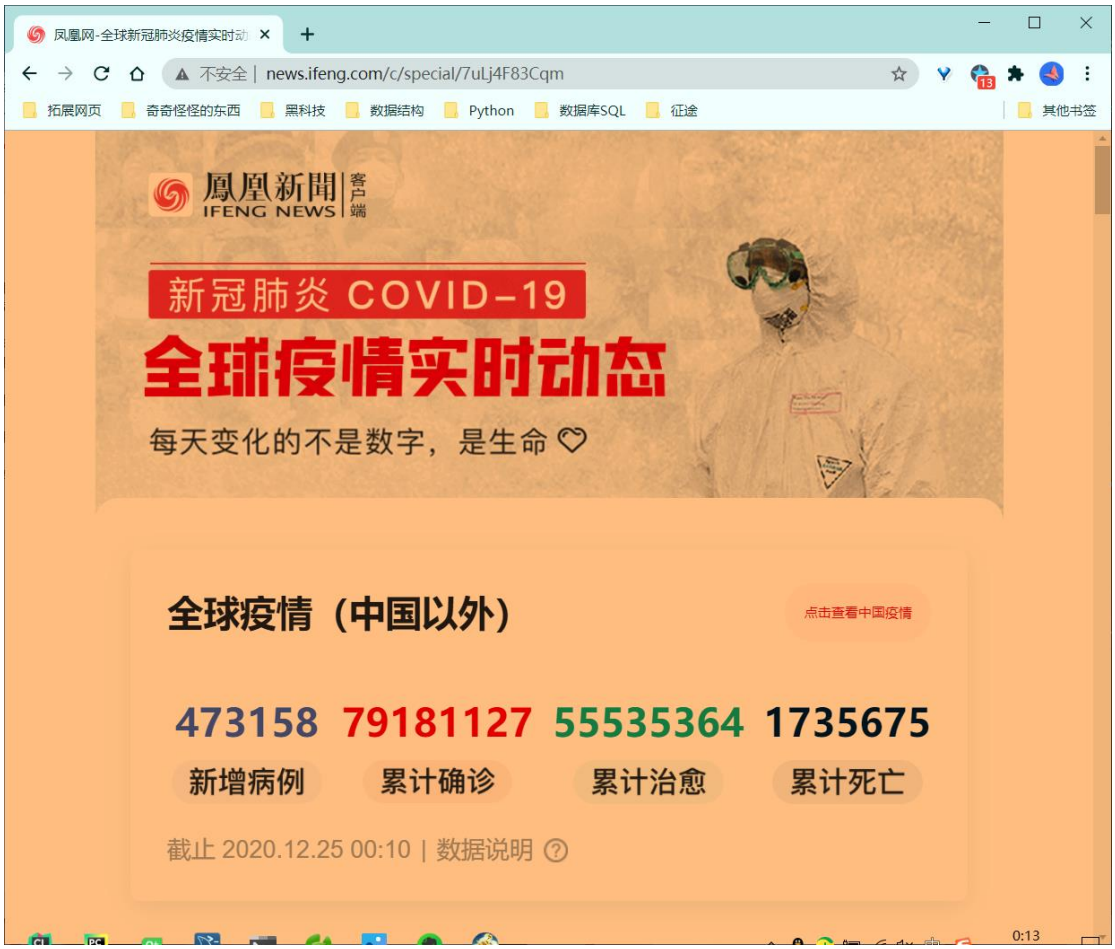
一、数据获取

①**新冠疫情数据获取：**http://news.ifeng.com/c/special/7uLj4F83Cqm



爬虫程序核心代码:

```
1.  class MySelenium(unittest.TestCase):
2.      def setUp(self):
3.          self.browser = webdriver.Chrome()
4.          self.date = datetime.datetime.now().strftime('%Y-%m-%d')
    # 日期
5.          self.country = []  # 国家
6.          self.new_confirm = []  # 今日确诊
7.          self.total_confirm = []  # 累计确诊
8.          self.dead = []  # 累计死亡
9.          self.heal = []  # 累计治愈
10.
11.         # 请求网页
12.         self.browser.get('http://news.ifeng.com/c/special/7uLj4F8
    3Cqm')
13.         # 模拟浏览器手动点击'查看更多'
```

```
14.         self.browser.find_element_by_xpath('//*[@id="list"]/div[2]/div[16]/span').click()
15.
16.     def test(self):
17.         time.sleep(3)  # 让网页完全加载，避免网页没加载完导致获取的数据丢失
18.         html = self.browser.page_source  # 获取网页源码
19.         content = etree.HTML(html)  # 把源码解析为 html dom 文档
20.
21.         # 使用 xpath 去匹配所有的信息
22.         country_list = content.xpath('//div[@class="tr_list_3X3bg3Ov"]/span[1]/text()')
23.         new_confirm_list = content.xpath('//div[@class="tr_list_3X3bg3Ov"]/span[2]/text()')
24.         total_confirm_list = content.xpath('//div[@class="tr_list_3X3bg3Ov"]/span[3]/text()')
25.         heal_list = content.xpath('//div[@class="tr_list_3X3bg3Ov"]/span[4]/text()')
26.         dead_list = content.xpath('//div[@class="tr_list_3X3bg3Ov"]/span[5]/text()')
27.
28.         report = []
29.         # 对全球数据进行聚类划分生成数据行
30.         for i in range(len(country_list)):
31.             self.country.append(country_list[i])
32.             self.new_confirm.append(new_confirm_list[i])
33.             self.total_confirm.append(total_confirm_list[i])
34.             self.heal.append(heal_list[i])
35.             self.dead.append(dead_list[i])
36.
37.             data = [self.date, self.country[i], self.new_confirm[i], self.total_confirm[i], self.heal[i], self.dead[i]]
38.             report.append(data)
39.
40.         # 网页比较特殊，将中国数据单独存放，需跳转页面获取中国的数据
41.         self.browser.find_element_by_xpath('//*[@id="root"]/div/div[2]/div[1]/a/span').click()
42.         html = self.browser.page_source  # 获取网页源码
43.         content = etree.HTML(html)  # 把源码解析为 html dom 文档
44.         c = '中国'
45.         new_c = content.xpath('//div[@class="num4_zVWiof-O"]/span[1]/span[2]/text()')
46.         total_c = content.xpath('//div[@class="num4_zVWiof-O"]/span[2]/text()')
```

```
47.         h = content.xpath('//div[@class="num5_2Fsthu_Y"]/span[2]/
    text()')
48.         d = content.xpath('//div[@class="num6_lDtxv4aj"]/span[2]/
    text()')
49.         self.country.append(c)
50.         self.new_confirm.append(int(new_c[0]))
51.         self.total_confirm.append(total_c[0])
52.         self.heal.append(h[0])
53.         self.dead.append(d[0])
54.         data = [self.date, c, int(new_c[0]), total_c[0], h[0], d[
    0]]
55.         report.append(data)
56.
57.         section_name = ['date', 'country', 'new_confirm', 'total_
    confirm', 'heal', 'dead']
58.         Data_csv = pd.DataFrame(columns=section_name, data=report
    )
59.         filename = 'Data' + str(self.date) + '.csv'
60.         Data_csv.to_csv(filename)
61.
62.     def tearDown(self):
63.         self.browser.quit()
64.         print('Covid-19 data get successfully!')
```

tip：采集的数据区间为【12-4，12-18】共 15 天

②全球各国人口总数获取：https://www.phb123.com/city/renkou/rk.html

首页 > 世界人口排名2018

世界人口排名2020

截止2020年12月24日，全球 230 个国家人口总数为7,585,204,179人，其中中国以 1,400,050,000 人位居第一，成为世界上人口最多的国家，印度以 1,354,051,854 人位居地球第二，第三至第十名分别是：美国、印度尼西亚、巴西、巴基斯坦、尼日利亚、孟加拉国、俄罗斯、等等

更新时间：2020年12月24日

世界人口排名

世界排名	国家	人口数量	增长率	人口密度（公里²）
1	中国	1,400,050,000	0.39%	144.30
2	印度	1,354,051,854	1.11%	411.87
3	美国	326,766,748	0.71%	34.86
4	印度尼西亚	266,794,980	1.06%	140.08
5	巴西	210,867,954	0.75%	24.76

爬虫程序核心代码：

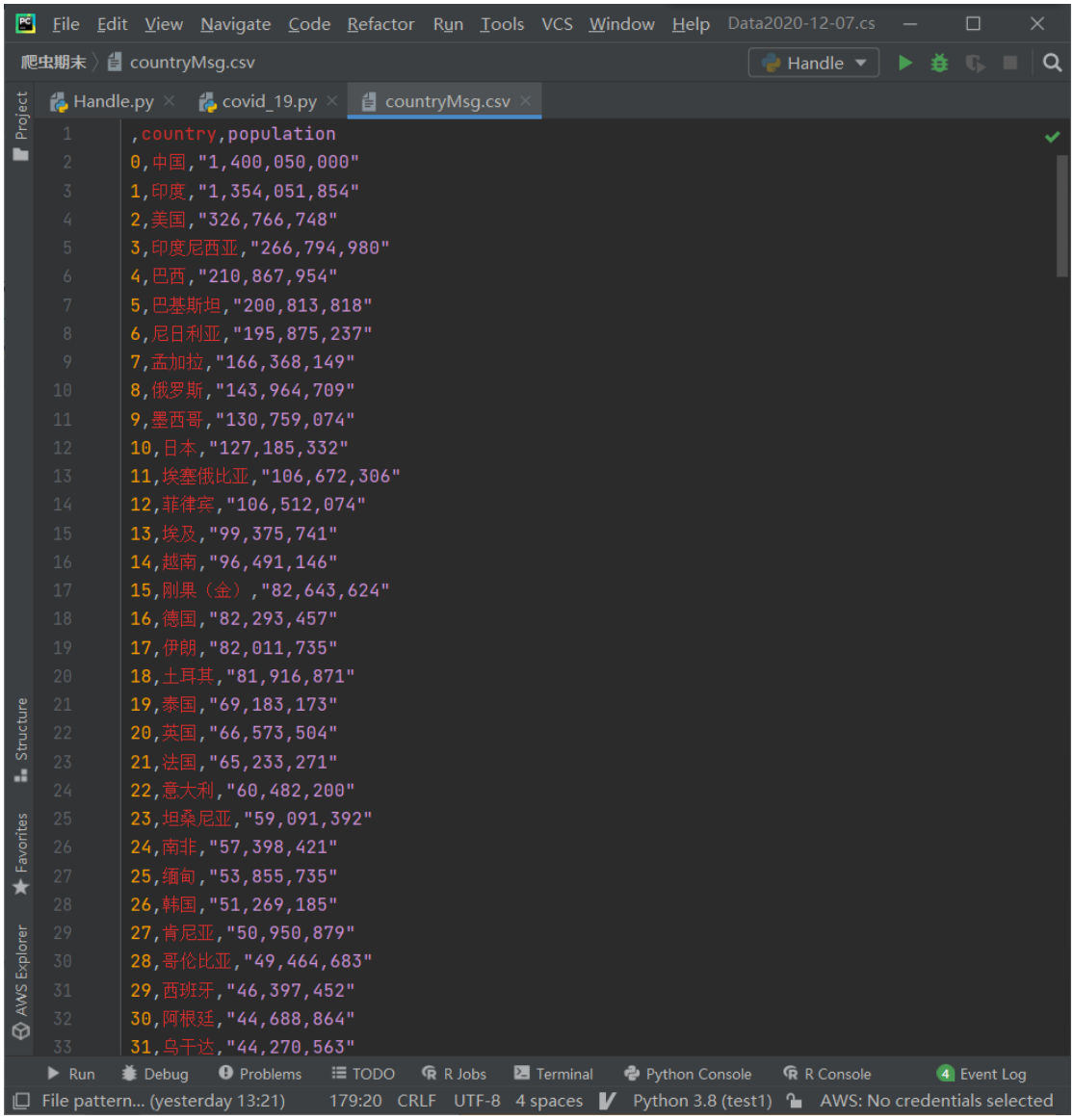
```
1. class getPopulation(unittest.TestCase):
2.     def setUp(self):
3.         self.browser = webdriver.Chrome()
4.         self.country = []  # 国家
5.         self.population = []  # 人口
6.
7.         # 请求网页
8.         self.browser.get('https://www.phb123.com/city/renkou/rk.h
   tml')
9.
10.    def testPopulation(self):
11.        count = 1
12.        rec = []
13.        while count <= 12:  # 获取网页列出的所有国家
14.            html = self.browser.page_source  # 获取网页源码
```

```
            content = etree.HTML(html)  # 把源码解析为 html dom 文档
            # 使用 xpath 去匹配所有的信息
            country_tmp = content.xpath('//span[@class="fl"]/../p/text()')
            popu_tmp = content.xpath('//span[@class="fl"]/../../../td[3]/text()')

            for i in range(len(popu_tmp)):
                data = [country_tmp[i], popu_tmp[i]]
                self.country.append(country_tmp[i])
                self.population.append(popu_tmp[i])
                rec.append(data)

            # 模拟浏览器手动点击'下一页'  tip:网页特殊，翻页可能会导致 xpath 改变，因此需要判断一下
            if count == 1 or count == 9:
                self.browser.find_element_by_xpath('//div[@class="page mt10"]/a[11]').click()
            elif 2 <= count <= 8:
                self.browser.find_element_by_xpath('//div[@class="page mt10"]/a[12]').click()
            elif count == 10:
                self.browser.find_element_by_xpath('//div[@class="page mt10"]/a[10]').click()
            elif count == 11:
                self.browser.find_element_by_xpath('//div[@class="page mt10"]/a[9]').click()
            count += 1

        section_name = ['country', 'population']
        Data_csv = pd.DataFrame(columns=section_name, data=rec)
        Data_csv.to_csv('countryMsg.csv')

    def tearDown(self):
        self.browser.quit()
        print('Population get successfully!')
```

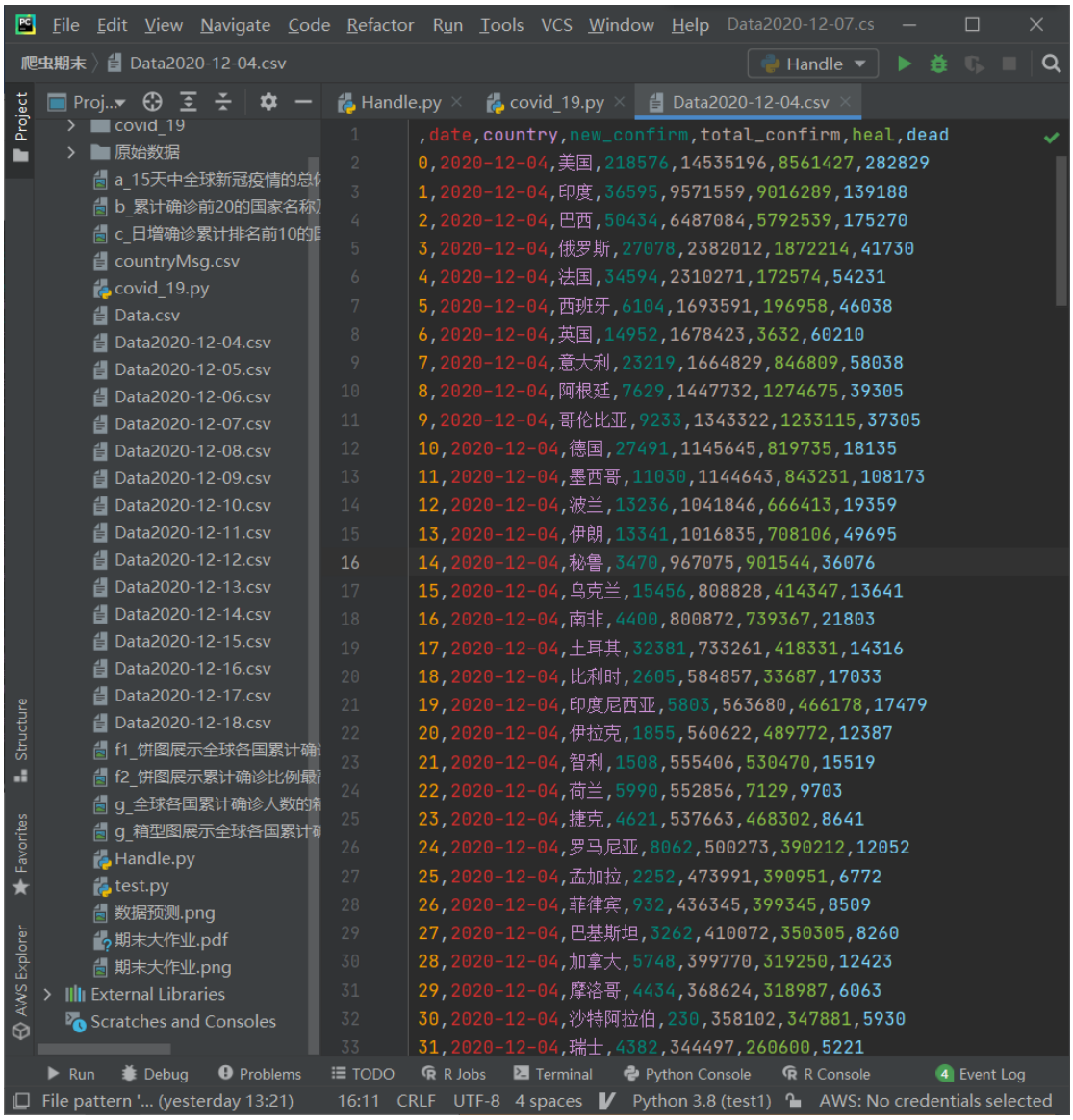
tip：适当人工补全数据中部分缺失的国家（有的国家有疫情数据，但没有人口数据以及命名差异带来的谬误（刚果布与刚果金）。

爬取的数据展示：

①人口总数：


File Edit View Navigate Code Refactor Run Tools VCS Window Help Data2020-12-07.cs
爬虫期末 countryMsg.csv
Handle
Handle.py
covid_19.py
countryMsg.csv
,country,population
0,中国,"1,400,050,000"
1,印度,"1,354,051,854"
2,美国,"326,766,748"
3,印度尼西亚,"266,794,980"
4,巴西,"210,867,954"
5,巴基斯坦,"200,813,818"
6,尼日利亚,"195,875,237"
7,孟加拉,"166,368,149"
8,俄罗斯,"143,964,709"
9,墨西哥,"130,759,074"
10,日本,"127,185,332"
11,埃塞俄比亚,"106,672,306"
12,菲律宾,"106,512,074"
13,埃及,"99,375,741"
14,越南,"96,491,146"
15,刚果（金）,"82,643,624"
16,德国,"82,293,457"
17,伊朗,"82,011,735"
18,土耳其,"81,916,871"
19,泰国,"69,183,173"
20,英国,"66,573,504"
21,法国,"65,233,271"
22,意大利,"60,482,200"
23,坦桑尼亚,"59,091,392"
24,南非,"57,398,421"
25,缅甸,"53,855,735"
26,韩国,"51,269,185"
27,肯尼亚,"50,950,879"
28,哥伦比亚,"49,464,683"
29,西班牙,"46,397,452"
30,阿根廷,"44,688,864"
31,乌干达,"44,270,563"
Project
Structure
Favorites
AWS Explorer
Run Debug Problems TODO R Jobs Terminal Python Console R Console Event Log
File pattern... (yesterday 13:21) 179:20 CRLF UTF-8 4 spaces Python 3.8 (test1) AWS: No credentials selected

②疫情数据：



二、数据预处理

在数据展示前，执行简单的文件读操作，并建立相应的数据结构为之后的数据展示铺垫。

```
1.  # part1.数据预处理
2. df = []  # 将 15 张表的数据全部载入一个列表中，df[i]表示 series 对象
3. world = []  # 单独存储全球疫情数据用于观测全球的趋势
4. for i in range(4, 18 + 1):  # 打开 15 天[4, 18]的数据文件加入列表
5.     if i < 10:
6.         fileNameStr = ('Data' + '2020-12-
   0{}' + '.csv').format(i)
7.     else:
8.         fileNameStr = ('Data' + '2020-12-
   {}' + '.csv').format(i)
9.     df.append(pd.read_csv(fileNameStr, encoding='utf-8'))
```

```
10.
11.     world_new_confirm = sum(df[i - 4]['new_confirm'])
12.     world_total_confirm = sum(df[i - 4]['total_confirm'])
13.     world_heal = sum(df[i - 4]['heal'])
14.     world_dead = sum(df[i - 4]['dead'])
15.     data = ['全球
    ', world_new_confirm, world_total_confirm, world_heal, world_dead
    ]
16.     world.append(data)
17.
18. length = len(df)
19.
20. country_dict = {}  # 将国家与人口存储为字典方便查询
21. # 同时适当人工补全数据中部分缺失的国家（有的国家有疫情数据，但没有人口数
    据以及命名差异带来的谬误（刚果布与刚果金））
22. with open('countryMsg.csv', 'r', encoding='utf-8') as csvfile:
23.     f_csv = csv.reader(csvfile)
24.     for row in f_csv:
25.         country_dict[row[1]] = row[2].replace(',', '')
26. plt.rcParams['font.sans-serif'] = ['SimHei']  # 添加对中文字体的支
    持
```

三、数据展示

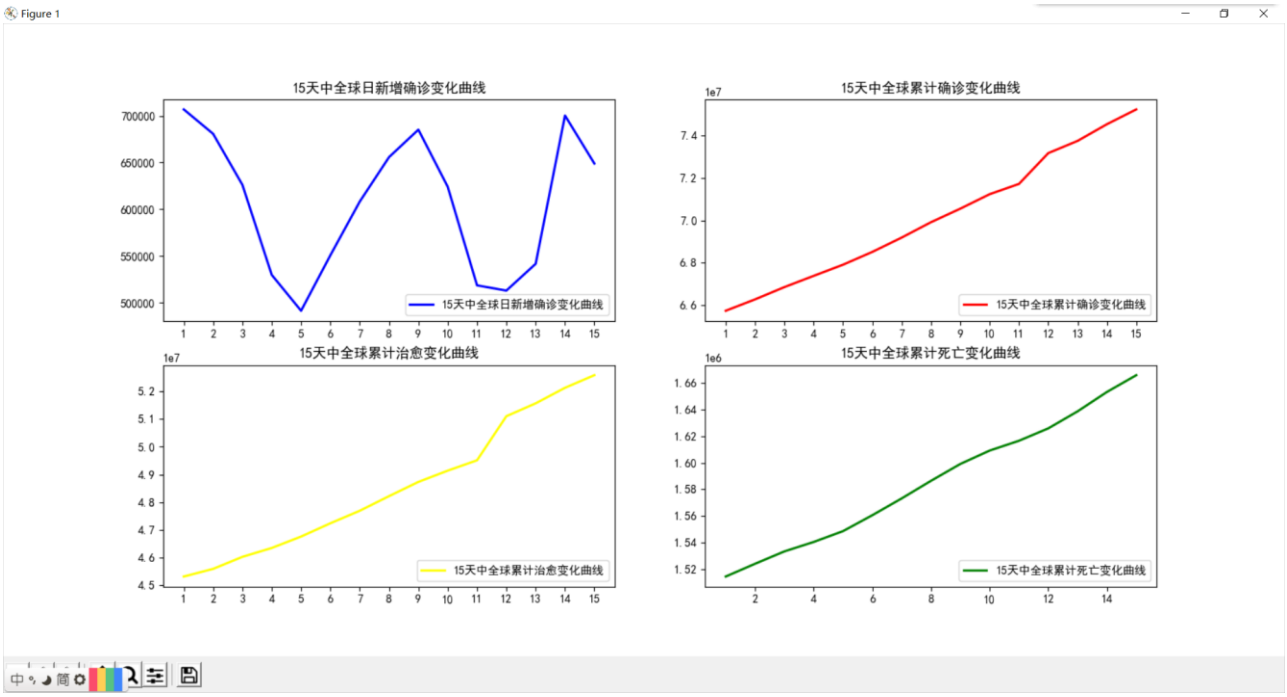
做一些典型的数据展示:

a)15 天中，全球新冠疫情的总体变化趋势(日新增确诊/累计确诊/累计治愈/累计死亡)

```
1. fig1 = plt.figure()
2. # a1.日新增确诊折线图
3. ax1 = plt.subplot(2, 2, 1)
4. x = np.linspace(1, length, length)  # 在 1-15 区间内生成 15 个数据
5. ax1.set_xticks(np.arange(1, length + 1))  # 设置 x 轴的刻度
6. data = []  # 提取出日新增数据
7. for i in world:
8.     data.append(i[1])
9.
10. plt.plot(x, np.array(data), color='blue', linewidth=2, linestyle=
    '-', label='15 天中全球日新增确诊变化曲线')
11. plt.legend(loc='lower right')
12. plt.title(label='15 天中全球日新增确诊变化曲线')
13.
```

```
# a2.累计确诊折线图
ax2 = plt.subplot(2, 2, 2)
x = np.linspace(1, length, length)  # 在 1-15 区间内生成 15 个数据
ax2.set_xticks(np.arange(1, length + 1))  # 设置 x 轴的刻度
data = []  # 提取出累计确诊数据
for i in world:
    data.append(i[2])

plt.plot(x, np.array(data), color='red', linewidth=2, linestyle='-', label='15 天中全球累计确诊变化曲线')
plt.legend(loc='lower right')
plt.title(label='15 天中全球累计确诊变化曲线')

# a3.累计治愈折线图
ax3 = plt.subplot(2, 2, 3)
x = np.linspace(1, length, length)  # 在 1-15 区间内生成 15 个数据
ax3.set_xticks(np.arange(1, length + 1))  # 设置 x 轴的刻度
data = []  # 提取出累计治愈数据
for i in world:
    data.append(i[3])

plt.plot(x, np.array(data), color='yellow', linewidth=2, linestyle='-', label='15 天中全球累计治愈变化曲线')
plt.legend(loc='lower right')
plt.title(label='15 天中全球累计治愈变化曲线')

# a4.累计死亡折线图
ax4 = plt.subplot(2, 2, 4)
x = np.linspace(1, length, length)  # 在 1-15 区间内生成 15 个数据
ax3.set_xticks(np.arange(1, length + 1))  # 设置 x 轴的刻度
data = []  # 提取出累计死亡数据
for i in world:
    data.append(i[4])

plt.plot(x, np.array(data), color='green', linewidth=2, linestyle='-', label='15 天中全球累计死亡变化曲线')
plt.legend(loc='lower right')
plt.title(label='15 天中全球累计死亡变化曲线')

fig1.savefig('a_15 天中全球新冠疫情的总体变化趋势.png')
```

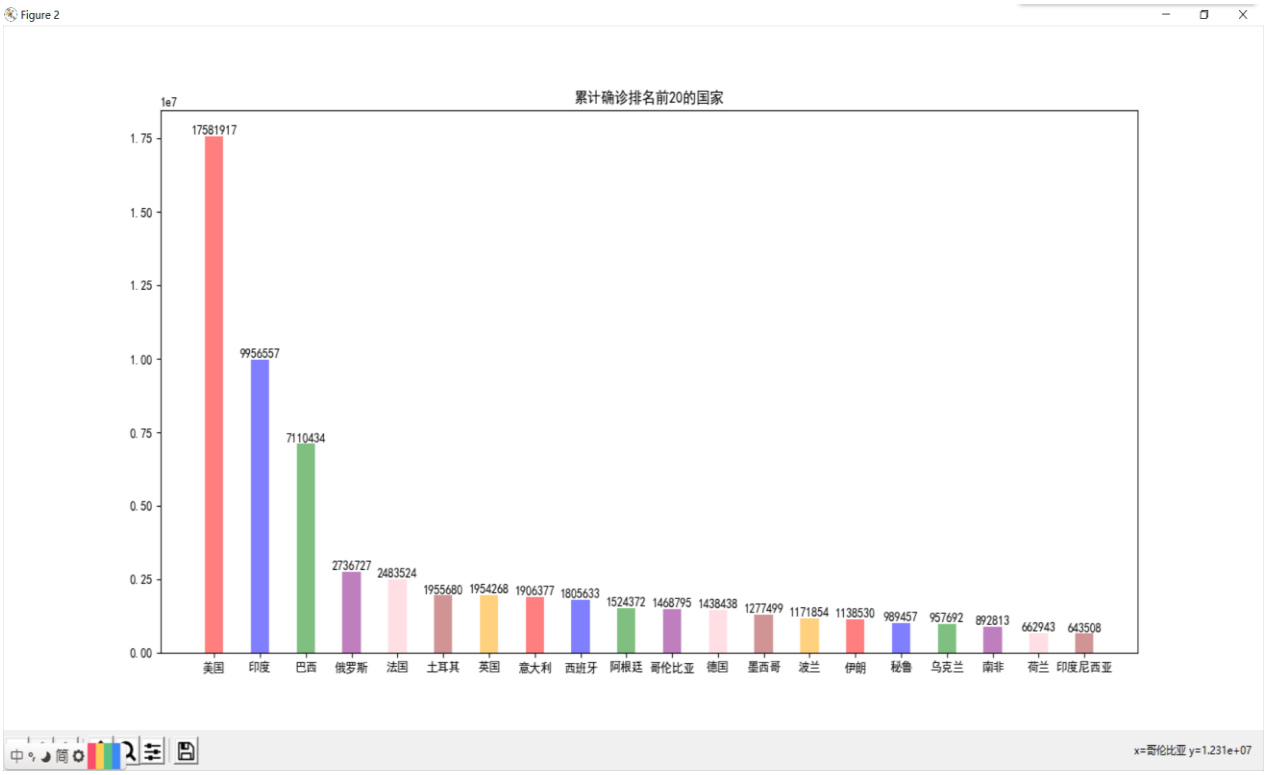
结果展示:



b)累计确诊数排名前 20 的国家名称及其数量

```
1.  df[length - 1].sort_values('total_confirm', ascending=False)  # 取第 15 天的累计确诊数据作为画图依据
2. x = []
3. y = []
4. for i in range(0, 20):
5.     x.append(df[length - 1]['country'][i])
6.     y.append(df[length - 1]['total_confirm'][i])
7.
8. fig2, ax = plt.subplots()
9. color_list = ['red', 'blue', 'green', 'purple', 'pink', 'brown', 'orange']  # 用于直方图中不同国家的区分
10.plt.bar(x, y, width=0.4, alpha=0.5, color=color_list)
11.for a, b in zip(x, y):
12.    plt.text(a, b + 0.2, '%d' % b, ha='center', va='bottom', fontsize=10)
13.plt.title('累计确诊排名前 20 的国家')
14.print("\n====== 累计确诊前 20 的国家 ======")
15.for i in range(0, 20):
16.    print(i + 1, x[i], y[i])
17.fig2.savefig('b_累计确诊前 20 的国家名称及其数量.png')
```

结果展示:

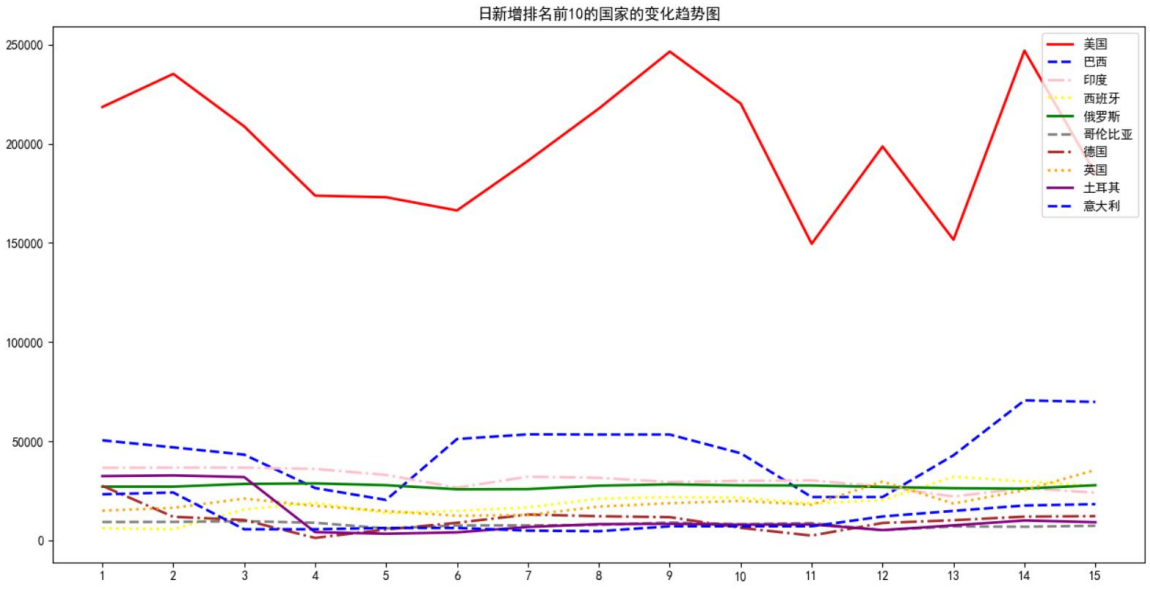


c)15 天中，每日新增确诊数排名前 10 的国家的日新增确诊数据曲线图

```
# 最后一天的累计确诊减去第一天的累计确诊即为 15 内的新增确诊，从中取前 10 做分析
data_list = []
for i in range(len(df[length-1]['total_confirm'])):
    data_list.append([df[length-1]['country'][i], df[length-1]['total_confirm'][i]-df[0]['total_confirm'][i]])
data_list.sort(key=lambda x: -x[1])

Data = []
for i in range(0, 10+1):
    data = []
    for k in range(len(df[0]['new_confirm'])):
        if df[0]['country'][k] == data_list[i][0]:
            for j in range(0, length):
                data.append(df[j]['new_confirm'][k])
            break
    Data.append([df[0]['country'][k], data])

fig4, ax4 = plt.subplots()
x = np.linspace(1, length, length)  # 在 1-15 区间内生成 15 个数据
ax4.set_xticks(np.arange(1, length + 1))  # 设置 x 轴的刻度
```

```
20.
21.plt.plot(x, Data[0][1], color='red', linewidth=2, linestyle='-
   ', label=Data[0][0])
22.plt.plot(x, Data[1][1], color='blue', linewidth=2, linestyle='--
   ', label=Data[1][0])
23.plt.plot(x, Data[2][1], color='pink', linewidth=2, linestyle='-.'
   , label=Data[2][0])
24.plt.plot(x, Data[3][1], color='yellow', linewidth=2, linestyle=':
   ', label=Data[3][0])
25.plt.plot(x, Data[4][1], color='green', linewidth=2, linestyle='-
   ', label=Data[4][0])
26.plt.plot(x, Data[5][1], color='grey', linewidth=2, linestyle='--
   ', label=Data[5][0])
27.plt.plot(x, Data[6][1], color='brown', linewidth=2, linestyle='-.
   ', label=Data[6][0])
28.plt.plot(x, Data[7][1], color='orange', linewidth=2, linestyle=':
   ', label=Data[7][0])
29.plt.plot(x, Data[8][1], color='purple', linewidth=2, linestyle='-
   ', label=Data[8][0])
30.plt.plot(x, Data[9][1], color='blue', linewidth=2, linestyle='--
   ', label=Data[9][0])
31.plt.title('日新增排名前 10 的国家的变化趋势图')
32.plt.legend(loc='upper right')
33.
34.fig4.savefig('c_日增确诊累计排名前 10 的国家的日新增变化曲线.png')
```

结果展示:



d)累计确诊人数占总人口比例最高的 10 个国家(确诊/总人口)

```
infection_rate = []  # 取第 15 天的累计确诊数据作为画图依据
for i in range(len(df[length - 1]['total_confirm'])):
    nation = df[length - 1]['country'][i]
    infection_rate.append([nation, df[length-1]['total_confirm'][i] / int(country_dict[nation]), df[length-1]['total_confirm'][i], int(country_dict[nation])])
infection_rate.sort(key=lambda x: -x[1])  # 按确诊比例降序排列并输出其中最大的前十国家
print("\n====== 累计确诊人数占国家总人口比例最高的 10 个国家 ======")
for i in range(0, length):
    print(infection_rate[i][0], infection_rate[i][1], infection_rate[i][2], infection_rate[i][3])
```

```
====== 累计确诊人数占国家总人口比例最高的10个国家 ======
安道尔 0.09702025911920263 7466 76953
卢森堡 0.0733143493116457 43279 590321
黑山 0.068330740171546 42995 629219
圣马力诺 0.06106028548439968 2049 33557
巴林 0.05727083656404336 89743 1566993
捷克 0.05669551304675184 602404 10625250
美国 0.053805710365609175 17581917 326766748
比利时 0.053490192954414394 615058 11498519
卡塔尔 0.05252873166548478 141557 2694849
亚美尼亚 0.05159650897431353 151392 2934152
格鲁吉亚 0.05153858419387525 201368 3907131
斯洛文尼亚 0.04902943409280917 102043 2081260
巴拿马 0.0480339536320652 199947 4162618
瑞士 0.04675906018164253 399511 8544034
克罗地亚 0.04489141451067198 186963 4164783
```

e)新冠患病致死率最低的 10 个国家(死亡/确诊)

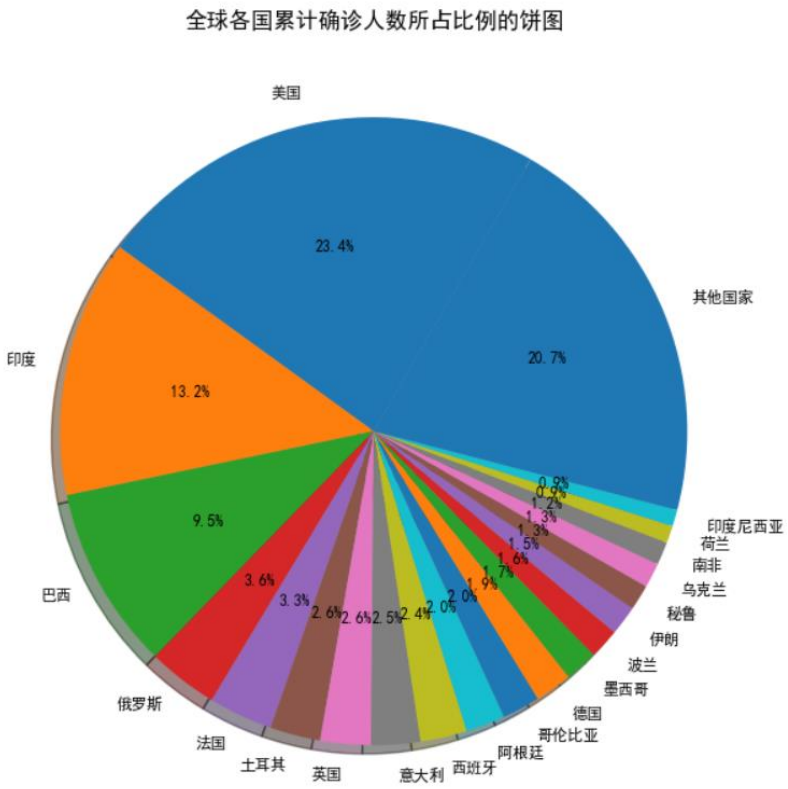
```
death_rate = []  # 取第 15 天的累计死亡人数作为依据
for i in range(len(df[length - 1]['dead'])):
    nation = df[length - 1]['country'][i]
```

```
    death_rate.append([nation, df[length-1]['dead'][i] / df[length-1]['total_confirm'][i], df[length-1]['dead'][i], df[length-1]['total_confirm'][i]])
death_rate.sort(key=lambda x: x[1])  # 按死亡率升序排列并输出其中最低的 30 个国家
print("\n====== 死亡率最低的 30 个国家 ======")  # 考虑到爬取的数据中有大量的'零患死亡'国家，适当延伸，列出 30 个国家，使数据有分析意义
for i in range(0, 30):
    print(death_rate[i][0], death_rate[i][1], death_rate[i][2], death_rate[i][3])
```

```
====== 死亡率最低的30个国家 ======
蒙古 0.0 0 923
厄立特里亚 0.0 0 741
不丹 0.0 0 440
柬埔寨 0.0 0 362
塞舌尔 0.0 0 202
圣文森特和格林纳丁斯 0.0 0 100
多米尼克 0.0 0 88
格林纳达 0.0 0 85
老挝 0.0 0 41
东帝汶 0.0 0 31
圣基茨和尼维斯 0.0 0 28
梵蒂冈 0.0 0 27
所罗门群岛 0.0 0 17
马绍尔群岛 0.0 0 4
瓦努阿图 0.0 0 1
新加坡 0.0004967709885742673 29 58377
布隆迪 0.0013315579227696406 1 751
卡塔尔 0.0017095586936711007 242 141557
博茨瓦纳 0.0029199323805132933 38 13014
阿联酋 0.003312862755838328 629 189866
马尔代夫 0.003577284245043971 48 13418
巴林 0.003888882698372018 349 89743
摩纳哥 0.004285714285714286 3 700
斯里兰卡 0.0045214344250713535 160 35387
马来西亚 0.004846689778196627 432 89133
冰岛 0.005010737294201861 28 5588
塞浦路斯 0.005250154416306362 85 16190
几内亚 0.005932517612161661 80 13485
科特迪瓦 0.006108763549513136 133 21772
加纳 0.006180792859410303 331 53553
```

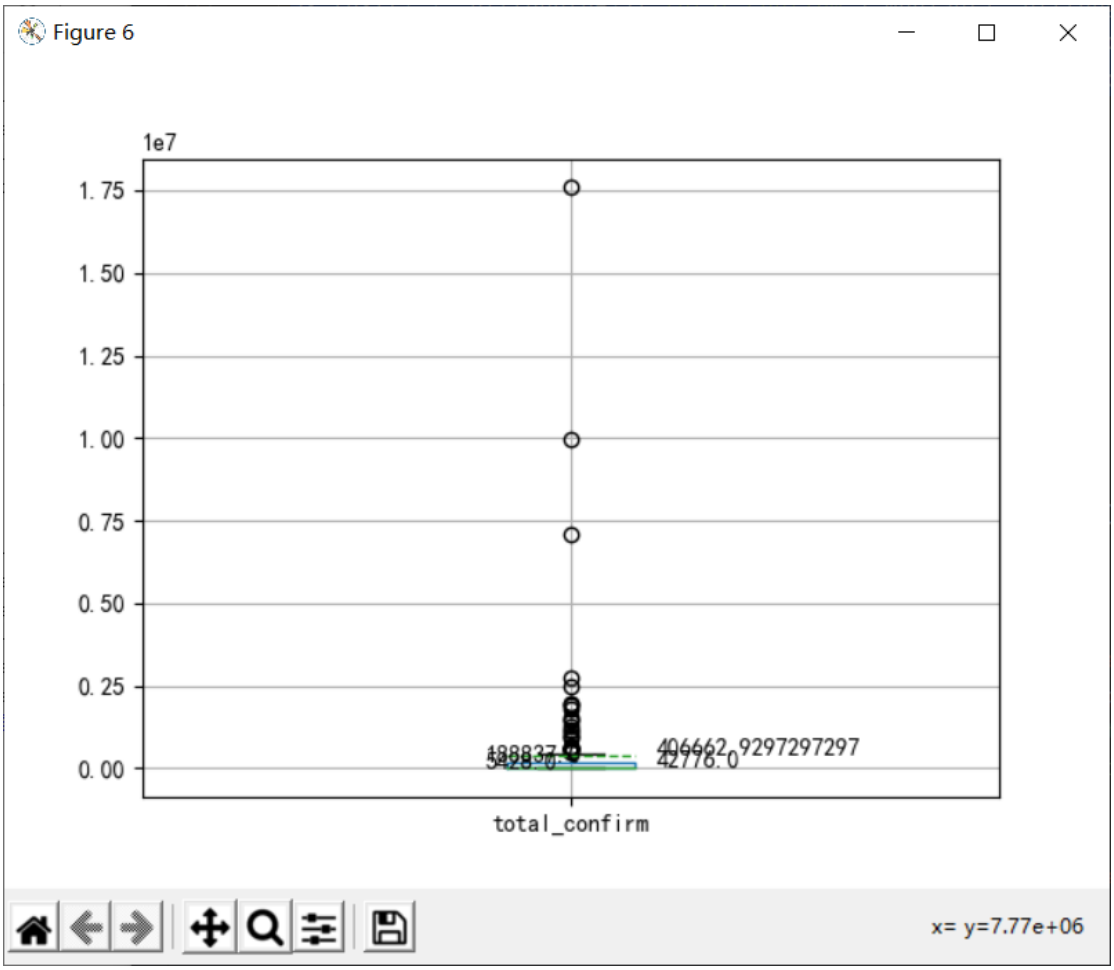
f)用饼图展示各个国家的累计确诊人数的比例（数据量较大，仅展示代表性数据）

```
fig3 = plt.figure()
plt.title('全球各国累计确诊人数所占比例的饼图')
section_label = []
section_size = []
total_num = world[length-1][2]

for i in range(1, 20+1):
    section_label.append(df[length-1]['country'][i-1])
    section_size.append(df[length-1]['total_confirm'][i-1])
    total_num -= df[length-1]['total_confirm'][i-1]
else:
    section_label.append('其他国家')
    section_size.append(total_num)

patches, texts, autotexts = plt.pie(section_size, labels=section_label, labeldistance=1.1, autopct="%1.1f%%", shadow=True, startangle=60, pctdistance=0.6)

proptease = fm.FontProperties()
proptease.set_size('small')
plt.setp(texts, fontproperties=proptease)
plt.setp(autotexts, fontproperties=proptease)
fig3.savefig('f_饼图展示全球各国累计确诊人数的比例.png')
```



g)展示全球各个国家累计确诊人数的箱型图，要有平均值

```
1. fig5 = plt.figure()
2. ax = df[length-
   1].boxplot(column=['total_confirm'], meanline=True, showmeans=Tru
   e, vert=True)
3. ax.text(1.1, df[length-1]['total_confirm'].mean(), df[length-
   1]['total_confirm'].mean())
4. ax.text(1.1, df[length-1]['total_confirm'].median(), df[length-
   1]['total_confirm'].median())
5. ax.text(0.9, df[length-
   1]['total_confirm'].quantile(0.25), df[length-
   1]['total_confirm'].quantile(0.25))
6. ax.text(0.9, df[length-
   1]['total_confirm'].quantile(0.75), df[length-
   1]['total_confirm'].quantile(0.75))
7. fig5.savefig('g_箱型图展示全球各国累计确诊人数.png')
```



h) 展示治愈率最高的前 30 个国家

```
heal_rate = []  # 取第 15 天的累计治愈人数作为依据
for i in range(len(df[length - 1]['heal'])):
    nation = df[length - 1]['country'][i]
    heal_rate.append([nation, df[length-1]['heal'][i] / df[length-1]['total_confirm'][i], df[length-1]['heal'][i], df[length-1]['total_confirm'][i]])
heal_rate.sort(key=lambda x: -x[1])  # 按治愈率升序排列并输出其中最低的 30 个国家
print("\n====== 治愈率最高的 30 个国家 ======")
for i in range(0, 30):
    print(heal_rate[i][0], heal_rate[i][1], heal_rate[i][2], heal_rate[i][3])
```

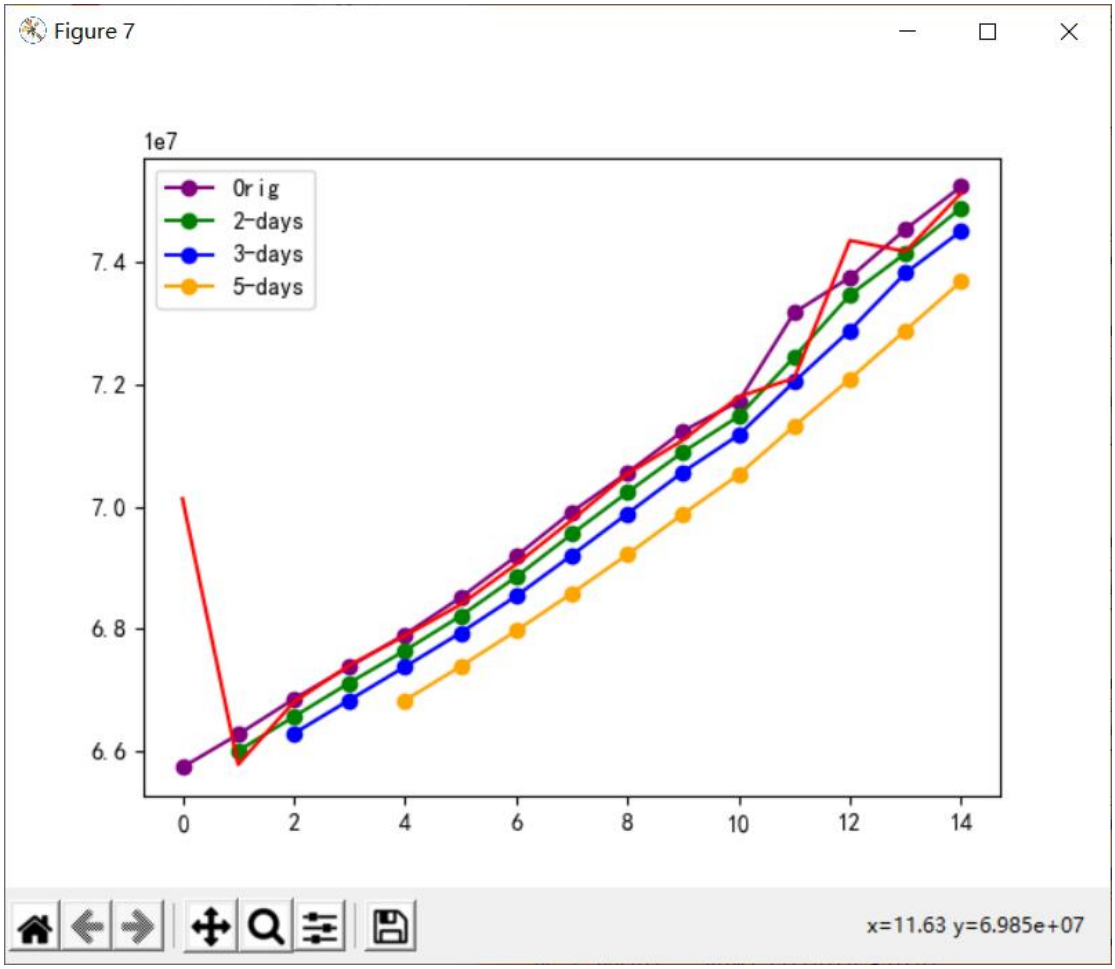
```
====== 治愈率最高的30个国家 ======
马绍尔群岛 1.0 4 4
瓦努阿图 1.0 1 1
新加坡 0.997858745738904 58252 58377
加蓬 0.9839965859383335 9223 9373
卡塔尔 0.9836532280282855 139243 141557
科特迪瓦 0.9829597648355686 21401 21772
巴林 0.9788618610922301 87846 89743
吉布提 0.9766031195840554 5635 5770
加纳 0.9759677329001176 52266 53553
沙特阿拉伯 0.9746169379445018 351365 360516
文莱 0.9736842105263158 148 152
科威特 0.9722878960813087 143113 147192
几内亚比绍 0.9718022067838169 2378 2447
赤道几内亚 0.9712313003452244 5064 5214
冰岛 0.9704724409448819 5423 5588
佛得角 0.9682663884541819 11137 11502
东帝汶 0.967741935483871 30 31
新西兰 0.9676190476190476 2032 2100
```

四、数据分析与预测

```
# part3.数据预测：针对全球累计确诊数，利用前 10 天的数据做后 5 天的预测，并与实际数据进行对比
world_data = []
for i in world:
    world_data.append(i[2])
world_data = DataFrame({'total_confirm':world_data})

fig6, ax = plt.subplots()
# 归一化
scaler = MinMaxScaler()
x_reshape = world_data['total_confirm'].values.reshape(-1,1)
total_confirm = scaler.fit_transform(x_reshape)

# 生成滑动窗口为 2 的预测值
predict_2 = world_data['total_confirm'].rolling(window=2, center=False).mean()
print("2-days mean error:", (world_data['total_confirm'] - predict_2).mean())                  # 平均误差
```

```
print("2-days mean absolute deviation:", abs(world_data['total_confirm'] - predict_2).mean())  # 平均绝对误差
print("2-days standard deviation:", (world_data['total_confirm'] - predict_2).std(), '\n')     # 均方误差

# 生成滑动窗口为 3 的预测值
predict_3 = world_data['total_confirm'].rolling(window=3, center=False).mean()
print("3-days mean error:", (world_data['total_confirm'] - predict_3).mean())
print("3-days mean absolute deviation:", abs(world_data['total_confirm'] - predict_3).mean())
print("3-days standard deviation:", (world_data['total_confirm'] - predict_3).std(), '\n')

# 生成滑动窗口为 5 的预测值
predict_5 = world_data['total_confirm'].rolling(window=5, center=False).mean()
print("5-days mean error:", (world_data['total_confirm'] - predict_5).mean())
print("5-days mean absolute deviation:", abs(world_data['total_confirm'] - predict_5).mean())
print("5-days standard deviation:", (world_data['total_confirm'] - predict_5).std(), '\n')

ax.plot(np.arange(15), world_data['total_confirm'],color='purple', marker='o')
ax.plot(np.arange(15), predict_2, color='green', marker='o')
ax.plot(np.arange(15), predict_3, color='blue', marker='o')
ax.plot(np.arange(15), predict_5, color='orange', marker='o')
ax.legend(["Orig", "2-days", "3-days", "5-days"])

arima = ARIMA(world_data['total_confirm'], order=(2, 0, 0))
result = arima.fit(disp=False)
print(result.aic, result.bic, result.hqic)
plt.plot(result.fittedvalues + 72.4, color='red')
```

```
41.fig6.savefig('数据预测.png')
```



五、结果分析

1)全球应对新冠疫情最好的 10 个国家

根据先前给出的治愈率最高的 30 个国家的数据, 可以发现处于前 10 的分别是[新加坡，加蓬，卡塔尔，科特迪瓦，巴林，吉布提，加纳，沙特阿拉伯，文莱，科威特]，在有一定规模的确诊人数时，有较高的治愈率(99.72%~99.78%)，显然这些国家的抗疫确实做得不错。

2)数据预测分析

使用滑动窗口的预测方法做数据预测分析，分别以 2,3,5 为窗口做预测，之后再采用自回归移动平均模型 ARIMA 做预测。通过对比红线与紫线的数据点与曲线走势，可以发现 ARIMA 数据预测分析比较合理，差距最大的第一个点是因为没有先前的数据做参考因此出现偏差。而对于使用滑动窗口的预测分析方法, 曲线整体走势一致, 但数据值有稳定的差值。