

Python 期末大作业

班级：2018211313 班

学号：2018211366

姓名：蒋潇逸

版本：1.0

更新：December 29, 2020

目录

1	爬虫爬取新冠疫情以及人口数据	2
1.1	疫情数据网页首页截图	2
1.2	爬取疫情数据	2
1.3	人口数据网页首页截图	6
1.4	爬取人口数据	7
2	数据分析以及展示	9
2.1	全球新冠疫情的总体变化趋势	9
2.2	累计确诊数排名前 20 的国家	11
2.3	每日新增确诊数排名前 10 个国家的每日新增确诊数据的曲线图	13
2.4	累计确诊人数占国家总人口比例最高的 10 个国家	16
2.5	死亡率最低的 10 个国家	18
2.6	各个国家的累计确诊人数的比例	20
2.7	全球各个国家累计确诊人数的箱型图	22
2.8	全球累计确诊人数前 10 名国家分布	24

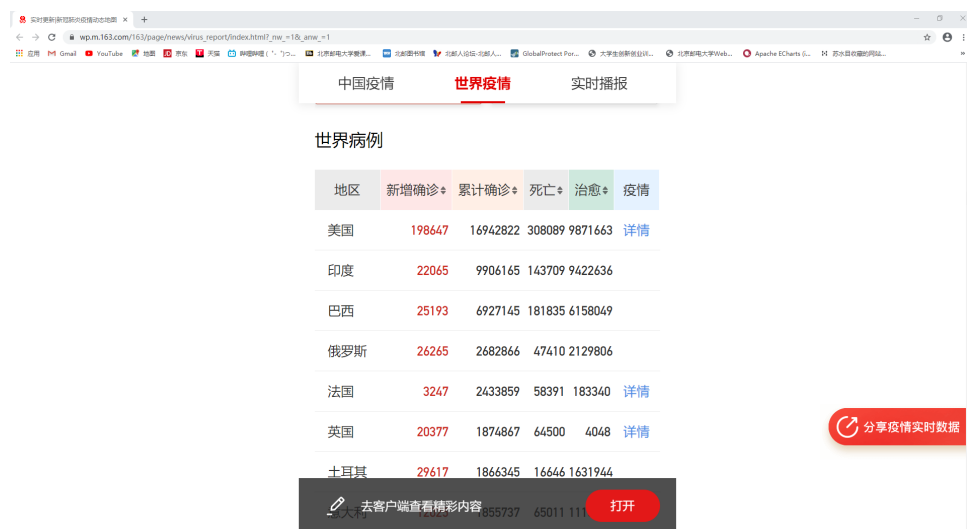
3 应对新冠疫情最好的 10 个国家 25

4 预测分析 28

1 爬虫爬取新冠疫情以及人口数据

1.1 疫情数据网页首页截图

本次大作业爬虫的网址为 https://wp.m.163.com/163/page/news/virus_report/index.html?_nw_=1&_anw_=1 , 该网址的首页截图如下图1所示。



The screenshot shows a web browser displaying the 163.com COVID-19 report page. The page has tabs for '中国疫情' (China), '世界疫情' (World), and '实时播报' (Real-time). Under '世界疫情', there is a table titled '世界病例' (World Cases) with columns: '地区' (Region), '新增确诊' (Newly Confirmed), '累计确诊' (Cumulative Confirmed), '死亡' (Deaths), '治愈' (Recovered), and '疫情' (Epidemic). The table lists data for the United States, India, Brazil, Russia, France, the United Kingdom, and Turkey. A red button '分享疫情实时数据' (Share real-time COVID-19 data) is visible on the right.

地区	新增确诊	累计确诊	死亡	治愈	疫情
美国	198647	16942822	308089	9871663	详情
印度	22065	9906165	143709	9422636	
巴西	25193	6927145	181835	6158049	
俄罗斯	26265	2682866	47410	2129806	
法国	3247	2433859	58391	183340	详情
英国	20377	1874867	64500	4048	详情
土耳其	29617	1866345	16646	1631944	

图 1: 爬取疫情数据网页截图

1.2 爬取疫情数据

利用 selenium 库爬取动态网页的数据，程序流程图以及核心代码如下：

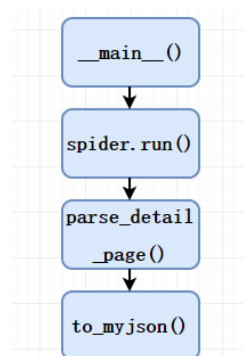


图 2: 程序流程图

```

1 import datetime
2 import json
3
4 from lxml import etree
5 from selenium import webdriver
6 from selenium.webdriver.common.by import By
7 from selenium.webdriver.support import expected_conditions as EC
8 from selenium.webdriver.support.ui import WebDriverWait
9 from selenium.webdriver.chrome.options import Options
10
11
12 class COVID_19(object):
13     driver_path = r'D:\Program Files\Python38\chromedriver.exe'
14     # 浏览器驱动路径
15
16     def __init__(self):
17         # 为了将Chrome不弹出界面，实现无界面爬取
18         chrome_options = Options()
19         chrome_options.add_argument('--headless')
20         chrome_options.add_argument('--disable-gpu')
21         self.driver = webdriver.Chrome(executable_path=COVID_19.
22                                         driver_path, options=chrome_options)
23         self.url = 'https://wp.m.163.com/163/page/news/virus_report/
24                     index.html?_nw_=1&_anw_=1'
25         # 将生成的json文件命名成当天日期
26         curr_time = datetime.datetime.now()
27         data = curr_time.date()
28         file_name = str(data) + 'MyData.json'
29         try: # 打开json文件
30             self.file = open(file_name, "w", encoding="utf-8")
31         except Exception as err:
32             print(err)
33
34     def run(self):
35         self.driver.get(self.url)
36         source = self.driver.page_source
37         # 等到页面的最底层的一个div加载出来之后再进行数据的爬取
38         WebDriverWait(driver=self.driver, timeout=10).until(
39             EC.presence_of_element_located((By.XPATH, '/html/body/div

```

```

        [3]/div[5]/div'))
38     # 调用parse_detail_page()函数
39     self.parse_detail_page(source)
40     self.driver.close()
41     self.file.close()    # 关闭文件
42
43     def parse_detail_page(self, source):
44         html = etree.HTML(source)
45         # 数据更新时间
46         DateTime = html.xpath('//*[@id="map_block"]/div/div[4]/div
            [2]/div[7]/text()')[0].strip()
47         DateTime = DateTime.replace("截至", "")
48         # 世界新型冠状病毒肺炎数据
49         Country = "世界"
50         str_temp = html.xpath('//*[@id="overseas_card"]/div[2]/div
            [1]/div[1]/p/span/text()')[0].strip()
51         MyNewAdd = str_temp.replace("+", "")    # 新增人数
52         MyTotal = html.xpath('//*[@id="overseas_card"]/div[2]/div[1]/
            div[1]/div/text()')[0].strip()    # 总人数
53         MyDead = html.xpath('//*[@id="overseas_card"]/div[2]/div[1]/
            div[3]/div/text()')[0].strip()    # 死亡人数
54         MyCure = html.xpath('//*[@id="overseas_card"]/div[2]/div[1]/
            div[4]/div/text()')[0].strip()    # 治愈人数
55         data = {
56             "Country": Country,
57             "NewAdd": MyNewAdd,
58             "Total": MyTotal,
59             "Dead": MyDead,
60             "Cure": MyCure,
61             "Time": DateTime,
62         }
63         print(data)    # 输出
64         # 调用to_myjson()函数
65         self.to_myjson(data)    # 把该数据写入json文件中
66         print('+ ' * 40)
67         # 各个国家新型冠状病毒肺炎数据
68         AllData = html.xpath('//*[@id="overseas_list"]/div')
69         for item in AllData:
70             # 疫情数据

```

```

71         Country = item.xpath('div/div[1]/text()')[0].strip()
72         NewAdd = item.xpath('div/div[2]/text()')[0].strip()
73         Total = item.xpath('div/div[3]/text()')[0].strip()
74         Dead = item.xpath('div/div[4]/text()')[0].strip()
75         Cure = item.xpath('div/div[5]/text()')[0].strip()
76         data = {
77             "Country": Country,
78             "NewAdd": NewAdd,
79             "Total": Total,
80             "Dead": Dead,
81             "Cure": Cure,
82             "Time": DateTime,
83         }
84         print(data)
85         # 调用to_myjson()函数
86         self.to_myjson(data) # 把该数据写入json文件中
87         print('+ ' * 40)
88
89     def to_myjson(self, data):
90         dict_item = dict(data) # 生成字典对象
91         json_str = json.dumps(dict_item, ensure_ascii=False) + "\n"
92         # 生成json串
93         self.file.write(json_str) # 将json串写入到文件中
94
95 if __name__ == '__main__':
96     spider = COVID_19()
97     spider.run() # 启动爬虫程序

```

爬取的部分数据如下：

	Country	NewAdd	Total	Dead	Cure	Time
1	世界	531705	63491019	1474794	44019339	2020-12-1 8:01
3	美国	161568	13919870	274332	8222879	2020-12-1 8:01
4	印度	38772	9431691	137139	8847600	2020-12-1 8:01
5	巴西	21138	6335878	173120	5597802	2020-12-1 8:01
6	俄罗斯	26046	2275936	39491	1763493	2020-12-1 8:01
7	法国	4443	2275016	52819	167913	2020-12-1 8:01
8	西班牙	6659	1664945	45069	196958	2020-12-1 8:01
9	英国	12420	1633733	58545	3570	2020-12-1 8:01
10	意大利	16376	1601554	55576	757507	2020-12-1 8:01
11	阿根廷	5726	1424533	38730	1257227	2020-12-1 8:01
12	哥伦比亚	8430	1316806	36766	1210489	2020-12-1 8:01
13	墨西哥	5668	1107071	105655	818397	2020-12-1 8:01
14	德国	6894	1069912	16694	749219	2020-12-1 8:01
15	波兰	5736	990811	17150	577514	2020-12-1 8:01
16	秘鲁	2162	962530	35923	893061	2020-12-1 8:01
17	伊朗	13321	962070	48246	668151	2020-12-1 8:01
18	南非	2302	790004	21535	731242	2020-12-1 8:01
19	乌克兰	10238	752343	12731	360360	2020-12-1 8:01
20	土耳其	31219	638847	13746	404727	2020-12-1 8:01
21	比利时	746	577345	16645	0	2020-12-1 8:01
22	伊拉克	2114	552549	12258	482674	2020-12-1 8:01
23	智利	1313	551743	15410	526604	2020-12-1 8:01
24	印度尼西亚	4617	538883	16945	450518	2020-12-1 8:01
25	荷兰	26	531930	9453	6947	2020-12-1 8:01
26	捷克	3575	523298	8295	451607	2020-12-1 8:01
27	罗马尼亚	3826	475362	11331	353188	2020-12-1 8:01

1.3 人口数据网页首页截图

本次大作业爬取国家人口数量的网址为 <https://www.phb123.com/city/renkou/rk.html> ,该网址的首页截图如下图3所示。

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
6		巴基斯坦	200,813,818	1.93%	227.70
7		尼日利亚	195,875,237	2.61%	212.04
8		孟加拉国	166,368,149	1.03%	1127.38
9		俄罗斯	143,964,709	0.00%	8.42
10		墨西哥	130,759,074	1.24%	66.57
11		日本	127,185,332	0.00%	336.53
12		埃塞俄比亚	106,672,306	2.46%	97.38
13		菲律宾	106,512,074	1.52%	311.12

相关文章

1. 中国十大养老机构排名: 静安健康上榜, 第
2. 2020年全国主要城市人口排名 重庆人口最多
3. 芬兰十大城市排名: 奥卢上榜, 第一名人口
4. 中国人口密度省份排名, 全国各省市人口密
5. 2019年600万净资产家庭分布城市 全国494万
6. 2019城市人口吸引力排行 深圳居首, 东莞
7. 中国肥胖率最高的省份排行 全国胖子最多
8. 2018各省城市低保人数排名 四川首94.57
9. 世界各国民均寿命排名2019, 附男女平均
10. 31省份常住人口排行榜 东三省以及北京

图 3: 爬取人口数量网页截图

1.4 爬取人口数据

利用 selenium 库爬取动态网页的数据，由于该网页的数据不在同一页中，故需要模拟点击事件实现翻页，核心代码如下：

```
1 import json
2 import time
3
4 from lxml import etree
5 from selenium import webdriver
6 from selenium.webdriver.common.by import By
7 from selenium.webdriver.support import expected_conditions as EC
8 from selenium.webdriver.support.ui import WebDriverWait
9 from selenium.webdriver.chrome.options import Options
10
11
12 class PeopleNum(object):
13     driver_path = r'D:\Program Files\Python38\chromedriver.exe'
14
15     def __init__(self):
16         # 为了将Chrome不弹出界面，实现无界面爬取
17         chrome_options = Options()
18         chrome_options.add_argument('--headless')
19         chrome_options.add_argument('--disable-gpu')
20         self.driver = webdriver.Chrome(executable_path=PeopleNum.
21                                     driver_path, options=chrome_options)
22         # 网页地址
23         self.url = 'https://www.phb123.com/city/renkou/rk.html'
24         file_name = 'PeopleNum.json'
25         try: # 打开json文件
26             self.file = open(file_name, "w", encoding="utf-8")
27         except Exception as err:
28             print(err)
29
30     def run(self):
31         self.driver.get(self.url)
32         # 利用循环跳转页面爬取数据
33         while True:
34             source = self.driver.page_source
35             # 等到该页面最后一个a元素加载出来后再进行数据的爬取
```

```

35         WebDriverWait(driver=self.driver, timeout=30).until(
36             EC.presence_of_element_located((By.XPATH, '/html/body
           /div[6]/div[1]/div[3]/div[2]/a[last()]')))
37         # 调用parse_detail_page()函数
38         self.parse_detail_page(source)
39         next_btn = self.driver.find_element_by_xpath('/html/body/
           div[6]/div[1]/div[3]/div[2]/a[last()-1]')
40         if "on" in next_btn.get_attribute('class'):
41             break # 已经在最后一页 退出
42         else:
43             next_btn.click() # 点击下一页实现跳转
44             time.sleep(1)
45         self.driver.close()
46         self.file.close() # 关闭文件
47
48     def parse_detail_page(self, source):
49         html = etree.HTML(source)
50         # 利用xpath爬取数据
51         AllData = html.xpath('/html/body/div[6]/div[1]/div[3]/table/
           tbody/tr')
52         del AllData[0]
53         for item in AllData:
54             Country = item.xpath('td[2]/a/p/text()')[0].strip()
55             PeopleNum = item.xpath('td[3]/text()')[0].strip()
56             data = {
57                 "Country": Country,
58                 "PeopleNum": PeopleNum,
59             }
60             print(data)
61             # 调用to_myjson()函数
62             self.to_myjson(data)
63             print('+ ' * 40)
64
65     def to_myjson(self, data):
66         dict_item = dict(data) # 生成字典对象
67         json_str = json.dumps(dict_item, ensure_ascii=False) + "\n"
68         # 生成json串
69         self.file.write(json_str) # 将json串写入到文件中

```



```

70
71 if __name__ == '__main__':
72     spider = PeopleNum()
73     spider.run()

```

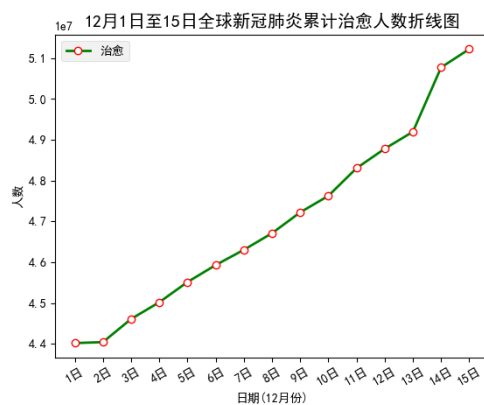
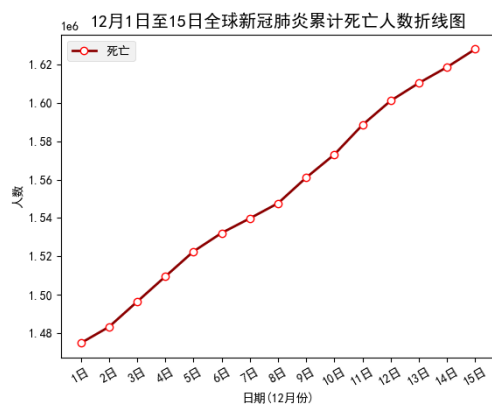
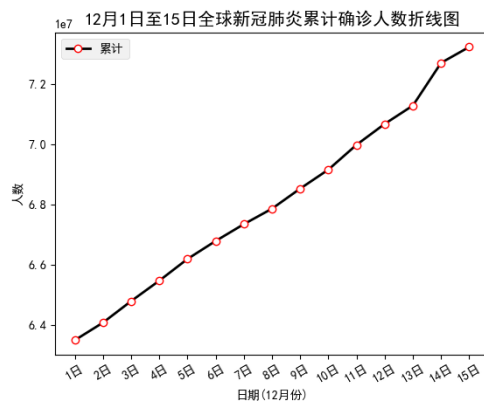
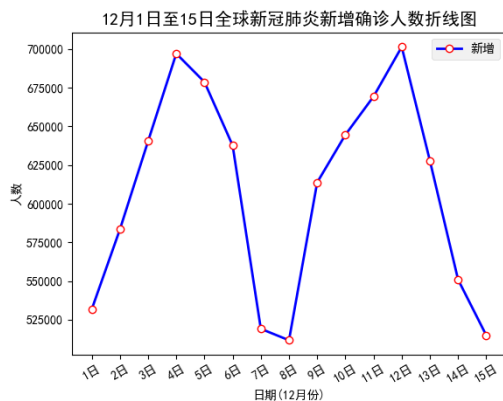
爬取的部分数据如下：

	A	B	C	D
1	Country	PeopleNum		
2	中国	1,400,050,000		
3	印度	1,354,051,854		
4	美国	326,766,748		
5	印度尼西亚	266,794,980		
6	巴西	210,867,954		
7	巴基斯坦	200,813,818		
8	尼日利亚	195,875,237		
9	孟加拉国	166,368,149		
10	俄罗斯	143,964,709		
11	墨西哥	130,759,074		
12	日本	127,185,332		
13	埃塞俄比亚	106,672,306		
14	菲律宾	106,512,074		
15	埃及	99,375,741		
16	越南	96,491,146		
17	刚果民主共和国	82,643,624		
18	德国	82,293,457		
19	伊朗	82,011,735		
20	土耳其	81,916,871		
21	泰国	69,183,173		
22	英国	66,573,504		
23	法国	65,233,271		
24	意大利	60,482,200		
25	坦桑尼亚	59,091,392		
26	南非	57,398,471		

2 数据分析以及展示

2.1 全球新冠疫情的总体变化趋势

画出从 12 月 1 日至 15 日，全球新冠疫情每日新增人数、累计确诊人数、累计死亡人数、累计治愈人数的曲线图如下。从折线图可以看出，全球新冠疫情每日新增人数基本在 60 万上下波动，处于一个稳定的状态，可见现在全球的疫情状况还是比较严峻的。疫情累计确诊人数和死亡人数大致是线性趋势，说明了目前新冠疫情得到了一定的控制，并没有呈现出疫情初期快速增长的趋势。累计治愈人数也表现出线性增长的趋势，其中 13 日至 14 日累计治愈人数有了飞跃性的增加，但是在 14 日至 15 日，增加趋势与之前相差不大，后来查看数据爬取的时间发现，14 日的数据是在晚上 10 点爬取的，而其他几日的的数据都是在下午或者早上爬取，故出现了时间上的误差，导致人数飞跃，同样的情况发生在累计确诊人数的折线图上，13 日至 14 日的数据也是有一个飞跃。折线图以及代码如下所示。



```

1 import pandas as pd
2 import matplotlib.pyplot as plt
3
4 plt.rcParams['font.sans-serif'] = ['SimHei'] # 用来正常显示中文标签
5 fileNameStr = 'total.csv' # 打开文件
6 df = pd.read_csv(fileNameStr, encoding='utf-8')
7 df_total = df[df['Country'] == '世界'] # 选出世界数据
8 # 将全球的新增累计死亡治愈时间数据都转换成列表
9 NewAdd = df_total.NewAdd.tolist()
10 Total = df_total.Total.tolist()
11 Dead = df_total.Dead.tolist()
12 Cure = df_total.Cure.tolist()
13 Time = df_total.Time.tolist()
14 # 将时间变成整数 2020-12-03->3
15 for i in range(0, len(Time)):
16     temp = str(Time[i]).split()[0]
17     Time[i] = int(temp.replace("2020-12-", ""))
18 ax = plt.axes() # 创建坐标对象
19 plt.style.use('bmh') # 设置图像风格

```

```

20 plt.rcParams['font.sans-serif'] = ['SimHei'] # 用来正常显示中文标签
21 plt.rcParams['axes.unicode_minus'] = False # 用来正常显示负号
22 ax.set_title("12月1日至15日全球新冠肺炎累计治愈人数折线图")
23 # 直方图标题
24 ax.set_ylabel('人数')
25 ax.set_xlabel('日期(12月份)')
26 # 转换成整形画图
27 NewAdd = list(map(int, NewAdd))
28 Total = list(map(int, Total))
29 Dead = list(map(int, Dead))
30 Cure = list(map(int, Cure))
31 # plt.plot(Time, NewAdd, marker='o', mec='r', mfc='w', color="blue",
32 #          linewidth=2, linestyle="--", label="新增")
33 # plt.plot(Time, Total, marker='o', color="black", mec='r', mfc='w',
34 #          linewidth=2, linestyle="--", label="累计")
35 # plt.plot(Time, Dead, marker='o', mec='r', mfc='w', color="darkred",
36 #          linewidth=2, linestyle="--", label="死亡")
37 plt.plot(Time, Cure, marker='o', mec='r', mfc='w', color="green",
38 #        linewidth=2, linestyle="--", label="治愈")
39 date = [] # 设置x轴的标签
40 for i in range(len(Time)):
41     date.append(str(Time[i]) + '日')
42 plt.xticks(Time, date, rotation=30)
43 ax.legend()
44 plt.show()

```

2.2 累计确诊数排名前 20 的国家

根据 15 日的疫情累计确诊人数排序选出排名前 20 名的国家，并画出南丁格尔图如图4。从图中可以看出，排名前 4 的国家的确诊人数相差很大，递减趋势明显，而第 4 名至第 20 名国家的确诊人数递减趋势明显放缓。第 1 名美国的确诊人数是第 10 名阿根廷的 10 倍还多，而阿根廷的确诊人数仅是第 20 名印度尼西亚的 2 倍多。这说明了各个国家的疫情严重程度不同，与其他国家相比，美国、印度、巴西这三个国家的累计确诊人数非常多，疫情情况也格外严重。

累计确诊数排名前20的国家

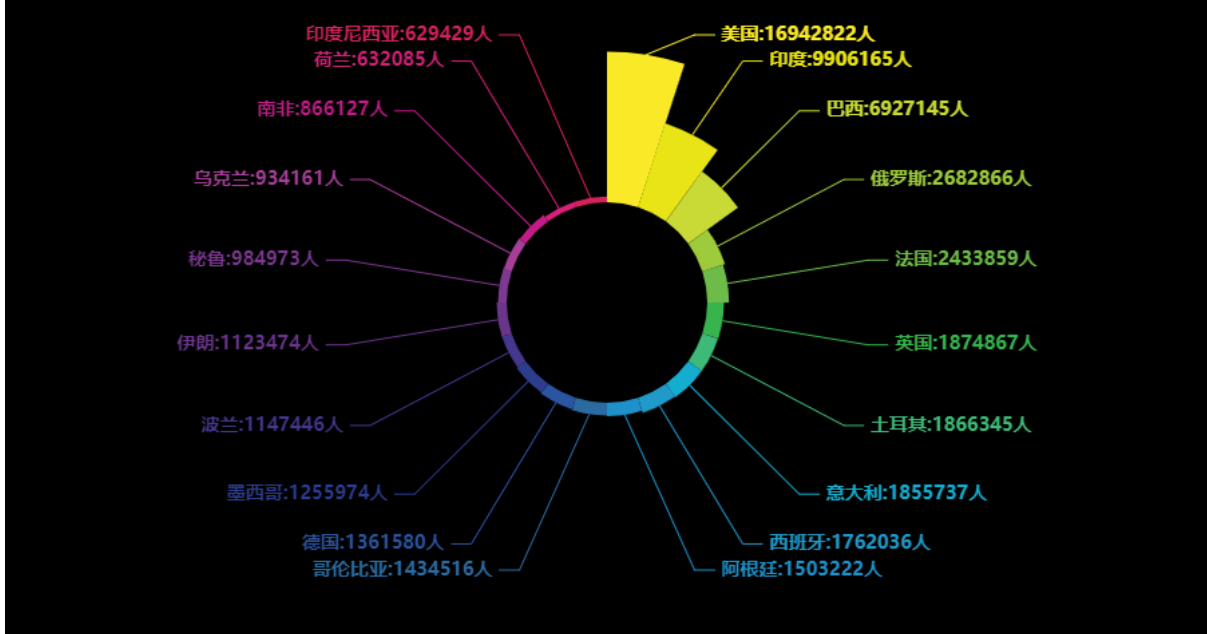


图 4: 累计确诊数排名前 20 的国家的南丁格尔图

```

1  # 累计确诊数排名前 20 的国家的南丁格尔图
2  import pandas as pd
3  from pyecharts.charts import Pie
4  from pyecharts import options as opts
5  from pyecharts.globals import ThemeType
6  # 打开CSV文件
7  fileNameStr = 'total.csv'
8  df = pd.read_csv(fileNameStr, encoding='utf-8')
9  # 在12-15中排序比较
10 df_total = df[df['Time'] == '2020-12-15 18:25']
11 # 对总确诊人数排序
12 df_total = df_total.sort_values(by='Total', ascending=False)
13 country = []
14 total_num = []
15 # 选取前20名的国家加到队列中
16 for i in range(0, 20):
17     country.append(df_total.iloc[i + 1, 0])
18     total_num.append(df_total.iloc[i + 1, 2])
19 # 颜色
20 color_series=[ '#FAE927', '#E9E416', '#C9DA36', '#9ECB3C', '#6DBC49',
21                '#37B44E', '#3DBA78', '#14ADCF', '#209AC9', '#1E91CA',
22                '#2C6BA0', '#2B55A1', '#2D3D8E', '#44388E', '#6A368B',
23                '#7D3990', '#A63F98', '#C31C88', '#D52178', '#D5225B']
    
```

```

24 # 创建数据框
25 df = pd.DataFrame({'country': country, 'total_num': total_num})
26 # 降序排序
27 df.sort_values(by='total_num', ascending=False, inplace=True)
28 # 提取数据
29 country = df['country'].values.tolist()
30 total_num = df['total_num'].values.tolist()
31 # 实例化Pie类
32 pie1 = Pie(init_opts=opts.InitOpts(theme=ThemeType.MACARONS))
33 # 设置颜色
34 pie1.set_colors(color_series)
35 # 添加数据, 设置饼图的半径, 是否展示成南丁格尔图
36 pie1.add("", [list(z) for z in zip(country, total_num)],
37             radius=["30%", "75%"],
38             center=["50%", "50%"],
39             rosetype="area")
40 # 设置全局配置项, 标题图例不显示
41 pie1.set_global_opts(title_opts=opts.TitleOpts(
42                     title='累计确诊数排名前20的国家'),
43                     legend_opts=opts.LegendOpts(is_show=False),
44                     toolbox_opts=opts.ToolboxOpts())
45 # 设置系列配置项, 在图中显示国家: xxx人
46 pie1.set_series_opts(label_opts=opts.LabelOpts(
47                     is_show=True, font_size=14,
48                     formatter="{b}:{c}人",
49                     font_weight="bold",
50                     font_family="Microsoft YaHei"))
51 # 生成html文档
52 pie1.render('累计确诊数排名前20的国家.html')

```

2.3 每日新增确诊数排名前 10 个国家的每日新增确诊数据的曲线图

画出该曲线图需要先对各个国家每日新增确诊数求和并排序, 得出前 10 名的国家为美国, 巴西, 印度, 土耳其, 俄罗斯, 意大利, 英国, 德国, 法国和乌克兰, 新增确诊数排名第 1 第 2 国家的新增确诊曲线图见图5, 具体代码如下:

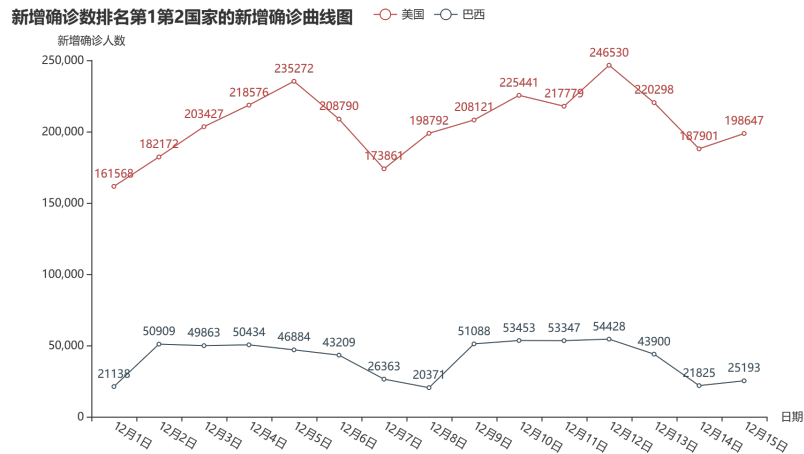


图 5: 新增确诊数排名第 1 第 2 国家的新增确诊曲线图

```

1 import pandas as pd
2 from pyecharts.render import make_snapshot
3 from snapshot_phantomjs import snapshot
4 import pyecharts.options as opts
5 from pyecharts.charts import Line
6
7 # 打开CSV文件
8 fileNameStr = 'total.csv'
9 df = pd.read_csv(fileNameStr, encoding='utf-8')
10 # '-'表示该国家还未更新数据, 将其人数设为0
11 for i in range(len(df['NewAdd'])):
12     if df['NewAdd'][i] == '-':
13         df['NewAdd'][i] = 0
14 print(df["NewAdd"])
15 # 将NewAdd列转成int类型便于处理
16 df["NewAdd"] = df["NewAdd"].astype('int')
17 # 计算出15天内, 各个国家每日新增确诊数的总和并写入new_add.csv文件中
18 print(df.groupby("Country")["NewAdd"].sum().to_csv("new_add.csv"))
19 df_add = pd.read_csv('new_add.csv', encoding='utf-8')
20 # 将15天内各个国家的新增人数排序
21 df_add = df_add.sort_values(by='NewAdd', ascending=False)
22 country = []
23 # i从1开始除去世界的的数据, 取出前10名的国家
24 for i in range(1, 11):
25     country.append(df_add.iloc[i][0])
26 print(country)
27 new_add = []

```

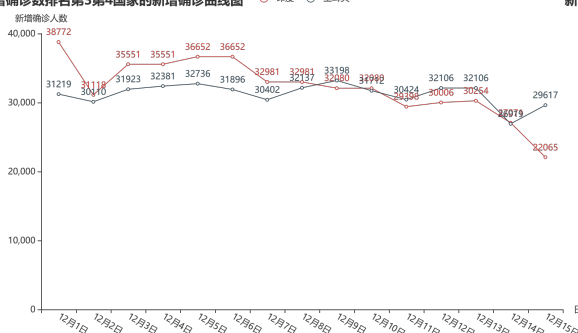
```

28 # 根据城市找到其对应的每日新增确诊数据
29 for i in range(0, 10):
30     new_add.append(df[df['Country'] == country[i]].NewAdd.tolist())
31 print(new_add)
32 # 时间为1号到15号
33 Time = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
34 line = (
35     Line()
36     .add_xaxis(['12月{}日'.format(Time[i]) for i in range(15)]) #
        12.1-12.15
37     .add_yaxis(country[0], new_add[0]) # 画出排名第一的国家的曲线图
38     .add_yaxis(country[1], new_add[1]) # 画出排名第二的国家的曲线图
39     .set_global_opts(xaxis_opts=opts.AxisOpts(axislabel_opts=opts.
        LabelOpts(rotate=-30), name='日期'), # 设置x轴标签旋转角度
40                     yaxis_opts=opts.AxisOpts(name='新增确诊人数'),
41                     title_opts=opts.TitleOpts(title='新增确诊数排名
        第1第2国家的新增确诊曲线图'))
42 line.render('新增确诊数排名第1第2国家的新增确诊曲线图.html')
43 make_snapshot(snapshot, line.render(), "新增确诊数排名第1第2国家的新
        增确诊曲线图.png")

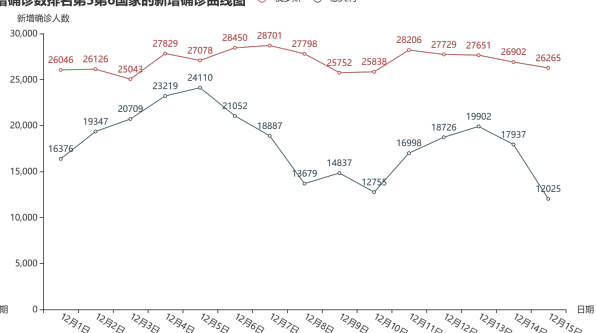
```

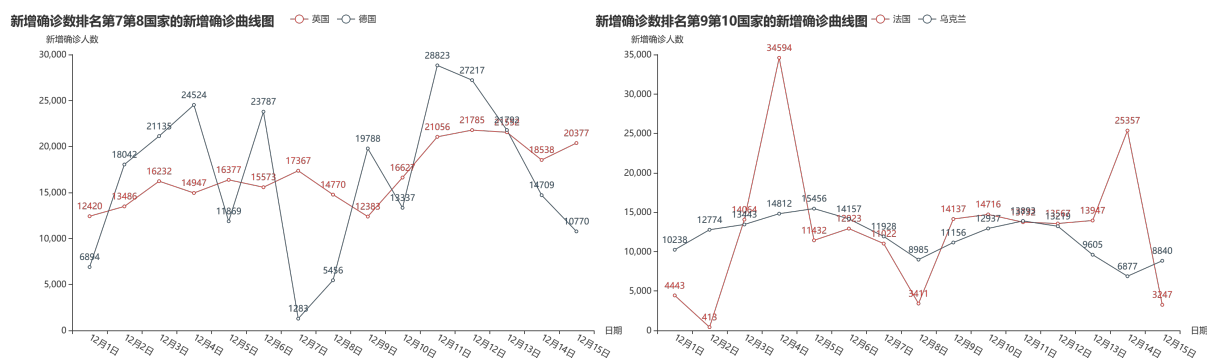
其余3到10名国家的曲线图代码与上述代码基本一致，故不再列出，新增确诊数排名第3到第10国家的新增确诊曲线图如下。从曲线图可以看出前10名国家中大部分国家的新增人数比较稳定，例如俄罗斯这15天以来新增确诊人数大致在26000人这个数值上下浮动；而有部分国家的新增确诊人数波动较为明显，例如法国和德国，德国这15日内最少新增确诊人数为1283人，最大为28823人，二者相差20倍多；还有一部分国家的新增人数呈现下降趋势，例如印度，说明印度的防疫工作做的越来越好。

新增确诊数排名第3第4国家的新增确诊曲线图



新增确诊数排名第5第6国家的新增确诊曲线图





2.4 累计确诊人数占国家总人口比例最高的 10 个国家

要计算累计确诊人数占国家总人口的比例，需要合并疫情数据文件和国家人口数量文件。本程序利用 `merge()` 函数将两者连接，算出比例并排序输出前 10 名的国家如下图6所示，柱状图如图7所示。可以看到比例最高的国家是安道尔，累计确诊人数占国家总人口比例高达 9.5929%，大致每 10 人中就有 1 名新冠疫情的感染者，可见这次疫情的传染性之强以及做好防护的重要性。

	Country	PeopleNum	NewAdd	Total	Dead	Cure	Time	rate
170	安道尔	76953	44	7382	79	6706	2020-12-15 18:25	0.095929
152	卢森堡	590321	1145	41900	410	33367	2020-12-15 18:25	0.070978
151	黑山	629219	377	41803	585	31138	2020-12-15 18:25	0.066436
180	圣马力诺	33557	2	1934	51	1608	2020-12-15 18:25	0.057633
139	巴林	1566993	125	89268	348	87332	2020-12-15 18:25	0.056968
73	比利时	11498519	1074	609211	18054	33687	2020-12-15 18:25	0.052982
130	卡塔尔	2694849	311	141272	241	138919	2020-12-15 18:25	0.052423
2	美国	326766748	198647	16942822	308089	9871663	2020-12-15 18:25	0.051850
127	亚美尼亚	2934152	438	149120	2529	127452	2020-12-15 18:25	0.050822
121	格鲁吉亚	3907131	3837	194900	1883	164786	2020-12-15 18:25	0.049883

图 6: 累计确诊人数占国家总人口比例最高的 10 个国家

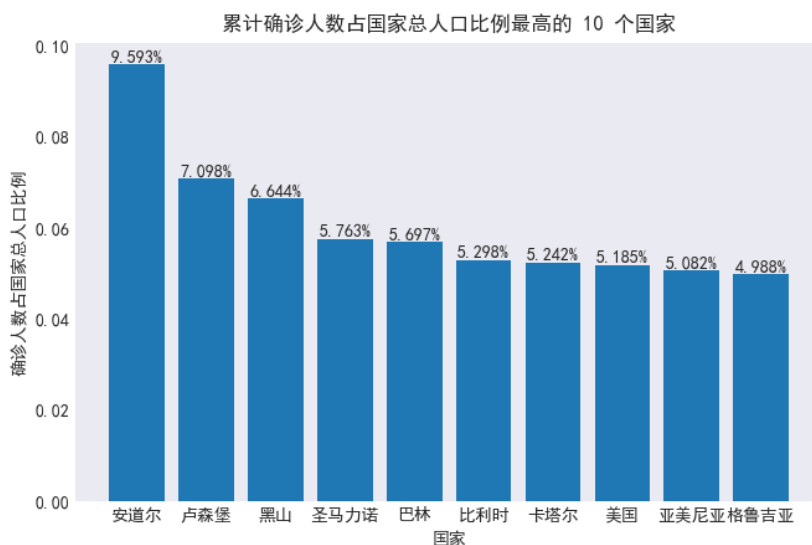


图 7: 累计确诊人数占国家总人口比例最高的 10 个国家柱状图

代码如下

```
1 import pandas as pd
2 import numpy as np
3 import matplotlib.pyplot as plt
4
5 # 打开疫情数据CSV文件
6 fileNameStr = 'total.csv'
7 df_result = pd.read_csv(fileNameStr, encoding='utf-8')
8 # 打开人口数量CSV文件
9 fileNameStr = 'PeopleNum.csv'
10 df_people_num = pd.read_csv(fileNameStr, encoding='utf-8')
11 # 选出12月15日的疫情数据
12 df_15 = df_result[df_result['Time'] == "2020-12-15 18:25"]
13 # 连接合并国家的疫情数据与国家人口数据
14 df_all = pd.merge(df_people_num, df_15, how='inner', on=['Country'])
15 # 将人口数据的逗号去掉 例如 123,456 变成 123456
16 for i in range(len(df_all['PeopleNum'])):
17     df_all['PeopleNum'][i] = str(df_all['PeopleNum'][i]).replace(",",
18                                     "")
19 # 转成int
20 df_all['PeopleNum'] = df_all['PeopleNum'].astype('int')
21 # 计算累计确诊人数占国家总人口比例
22 df_all['rate'] = round(df_all['Total'] / df_all['PeopleNum'], 6)
23 # 按照累计确诊人数占国家总人口比例排名, 降序
24 df_all = df_all.sort_values(by='rate', ascending=False)
25 # 输出累计确诊人数占国家总人口比例最高的 10 个国家
26 pd.set_option('display.max_columns', None) # 输出完整列
27 pd.set_option('display.width', None) # 宽度无限
28 print(df_all[0:10])
29 plt.style.use('seaborn-dark') # 设置图像风格
30 fig, ax = plt.subplots()
31 plt.rcParams['font.sans-serif'] = ['SimHei'] # 用来正常显示中文标签
32 ax.set_title("累计确诊人数占国家总人口比例最高的 10 个国家")
33 # 设置坐标轴名称
34 plt.xlabel('国家')
35 plt.ylabel('确诊人数占国家总人口比例')
36 plt.bar(df_all[0:10].Country, df_all[0:10].rate)
37 for a, b in zip(df_all[0:10].Country, df_all[0:10].rate): # 在直方图
    上显示数字
```

```

37     plt.text(a, b + 0.0015, f'{b*100:.3f}%', ha='center', va='center'
    , fontsize=10)
38 plt.savefig("C:\\Users\\Lenovo\\Desktop\\1.jpg") # 保存图片
39 plt.show()

```

2.5 死亡率最低的 10 个国家

本程序先把数据转成整形，根据累计死亡人数/累计确诊人数计算出死亡率，排序输出结果如图8所示。观察结果可知死亡率最低的国家往往是感染人数不多的偏远的小国家，有个别国家甚至才感染了 1 个人，这种情况下的死亡率不确定性太大，故加入确诊人数超过 200 人的条件下，比较各个国家的死亡率，结果如图9所示，柱状图如图10所示。值得一提的是，新加坡在感染人数达到 58341 人的情况下，死亡率只有 0.000497，排名第 2，可见新加坡的医疗水平很先进。卡塔尔在死亡率方面也表现优异，141292 人感染新冠疫情，仅有 241 人死亡。

	Country	NewAdd	Total	Dead	Cure	Time	dead_rate
3119	圣皮埃尔岛和密克隆岛	-	1	0	1	2020-12-15 18:25	0.0
3077	法罗群岛	-	187	0	187	2020-12-15 18:25	0.0
3075	蒙古	-	197	0	98	2020-12-15 18:25	0.0
3072	柬埔寨	3	362	0	312	2020-12-15 18:25	0.0
3094	不丹	-	66	0	21	2020-12-15 18:25	0.0
3096	法属波利尼西亚	-	60	0	60	2020-12-15 18:25	0.0
3098	厄立特里亚	-	41	0	39	2020-12-15 18:25	0.0
3100	纳米比亚	-	31	0	17	2020-12-15 18:25	0.0
3102	圣文森特和格林纳丁斯	-	27	0	25	2020-12-15 18:25	0.0
3103	东帝汶	-	24	0	24	2020-12-15 18:25	0.0

图 8: 死亡率最低的 10 个国家

	Country	NewAdd	Total	Dead	Cure	Time	dead_rate
3072	柬埔寨	3	362	0	312	2020-12-15 18:25	0.000000
2998	新加坡	16	58341	29	58210	2020-12-15 18:25	0.000497
2969	卡塔尔	311	141272	241	138919	2020-12-15 18:25	0.001706
3071	法属留尼汪岛	-	487	1	447	2020-12-15 18:25	0.002053
3066	法属圭亚那	0	1255	3	534	2020-12-15 18:25	0.002390
2956	阿联酋	1226	187267	622	165023	2020-12-15 18:25	0.003321
3023	马尔代夫	11	13379	48	12731	2020-12-15 18:25	0.003588
2991	巴林	125	89268	348	87332	2020-12-15 18:25	0.003898
3006	斯里兰卡	332	33478	154	24309	2020-12-15 18:25	0.004600
2992	马来西亚	1772	86618	422	71681	2020-12-15 18:25	0.004872

图 9: 确诊人数超过 200 人死亡率最低的 10 个国家

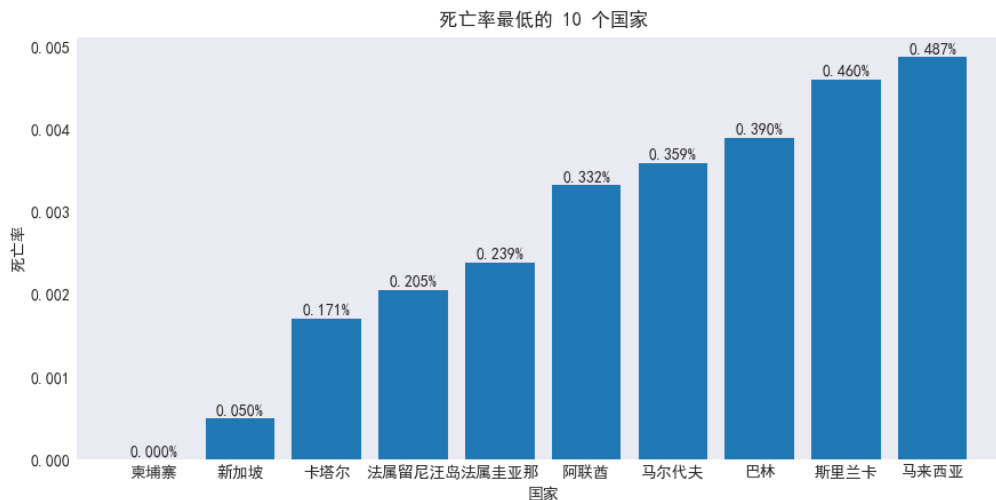


图 10: 确诊人数超过 200 人死亡率最低的 10 个国家柱状图

```

1 import pandas as pd
2 import matplotlib.pyplot as plt
3
4 # 打开疫情数据CSV文件
5 fileNameStr = 'total.csv'
6 df_result = pd.read_csv(fileNameStr, encoding='utf-8')
7 # 选出12月15日的疫情数据
8 df_15 = df_result[df_result['Time'] == "2020-12-15 18:25"]
9 # 计算死亡率（累计死亡人数/累计确诊人数）
10 df_15['dead_rate'] = round(df_15['Dead'] / df_15['Total'], 6)
11 # 对死亡率排序，降序
12 df_top = df_15.sort_values(by='dead_rate', ascending=True)
13 print(df_top)
14 pd.set_option('display.max_columns', None) # 显示所有列
15 pd.set_option('display.max_rows', None) # 显示所有行
16 pd.set_option('display.width', None) # 宽度无限
17 country = [] # 存储国家
18 dead_rate = [] # 存储死亡率
19 j = 0 # 计数
20 # 取出前10名
21 for i in range(0, 10):
22     while 1:
23         # 判断若确诊人数大于200则满足条件
24         if int(df_top.iloc[j][2]) > 200:
25             country.append(df_top.iloc[j][0])

```

```

26         j = j + 1
27         break
28     j = j + 1
29 print(country)
30 # 存储对应的死亡率
31 for i in range(0, 10):
32     dead_rate.append(float(df_top[df_top['Country'] == country[i]].
33                             dead_rate))
34 # print(df_top[0:10])
35 # print(df_top[df_top['Total'] > 200])
36 print(dead_rate)
37 plt.style.use('seaborn-dark') # 设置图像风格
38 fig, ax = plt.subplots()
39 plt.rcParams['font.sans-serif'] = ['SimHei'] # 用来正常显示中文标签
40 ax.set_title("死亡率最低的 10 个国家")
41 # 设置坐标轴名称
42 plt.xlabel('国家')
43 plt.ylabel('死亡率')
44 plt.bar(country, dead_rate)
45 for a, b in zip(country, dead_rate): # 在直方图上显示数字
46     plt.text(a, b + 0.0001, f'{b * 100:.3f}%', ha='center', va='
47         center', fontsize=10)
48 plt.savefig("C:\\Users\\Lenovo\\Desktop\\2.jpg") # 保存图片
49 plt.show()

```

2.6 各个国家的累计确诊人数的比例

若用饼图展示 100 多个国家的累计确诊人数的比例，由于国家数过多，会导致饼图的效果不佳，故此处仅展示前 25 名累计确诊人数的国家，并将后 25 名的国家合并为其他国家。结果如图 11。从图中可以看出，累计确诊人数前 3 名的国家差不多占全球累计确诊人数的 45%，说明不同国家的疫情严重性不同，世界范围内的疫情分布不均匀，美国、印度、巴西这三个国家的疫情状况格外严峻。

各个国家的累计确诊人数的比例

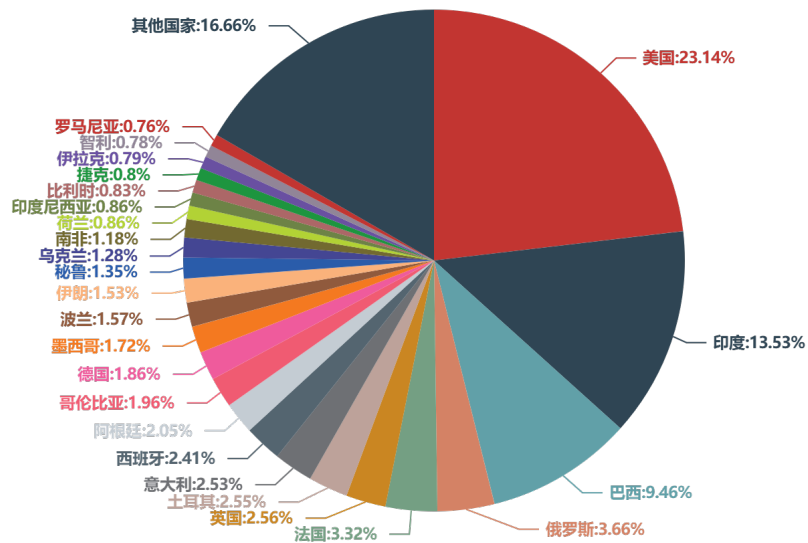


图 11: 各个国家的累计确诊人数的比例

```
1 import pandas as pd
2 from pyecharts.render import make_snapshot
3 from snapshot_phantomjs import snapshot
4 from pyecharts.charts import Pie
5 from pyecharts import options as opts
6
7 # 打开CSV文件
8 fileNameStr = 'total.csv'
9 df_result = pd.read_csv(fileNameStr, encoding='utf-8')
10 # 选出12月15日的疫情数据
11 df_15 = df_result[df_result['Time'] == "2020-12-15 18:25"]
12 # 对确诊人数排序
13 df_15 = df_15.sort_values(by='Total', ascending=False)
14 # 转换成列表
15 count = df_15.Total.tolist()
16 section_names = df_15.Country.tolist()
17 # 把世界的的数据删除, 只留下各个国家的数据
18 del count[0]
19 del section_names[0]
20 # 将排名后25名的国家设为其他国家, 把他们的人数求和
21 other = 0
22 for i in range(25, len(count)):
23     other += count[i]
```

```

24 # 删除25名之后的国家
25 del count[25:]
26 # 设为其他国家
27 count.append(other)
28 del section_names[25:]
29 section_names.append("其他国家")
30 data_pair = [list(z) for z in zip(section_names, count)]
31 pie = Pie()
32 # 画出饼图
33 pie.add(
34     series_name="比例",
35     data_pair=data_pair,
36     center=["50%", "50%"],
37 )
38 pie.set_global_opts(title_opts=opts.TitleOpts(title="各个国家的累计确
    诊人数的比例"), legend_opts=opts.LegendOpts(is_show=False))
39 pie.set_series_opts(label_opts=opts.LabelOpts(is_show=True, formatter=
    "{b}:{d}% ", font_weight="bold", font_family="Microsoft YaHei"))
40 pie.render(path="Bing1.html")
41 make_snapshot(snapshot, pie.render(), "各个国家的累计确诊人数的比例.
    png")

```

2.7 全球各个国家累计确诊人数的箱型图

全球各个国家累计确诊人数的箱型图如图12. 各国累计确诊人数的平均数是 353771 人，中位数是 17587 人，箱型图的异常值集中在较大值一侧，说明个别国家的疫情状况不容乐观，确诊人数远超其他国家。从箱形图可得，有一半的国家的确诊人数集中在 17587 以下，有 75% 的国家的确诊人数在 151144.5 以下，可以看出对于各国累计确诊人数的数据，主要集中在较小一侧。大体上看，确诊人数越少，国家聚集的数量越多，但是该数据集的平均数是 353771 人，差不多是上四分位数的两倍还多，说明了部分国家的确诊人数非常多，和其他国家的确诊人数不在一个量级。

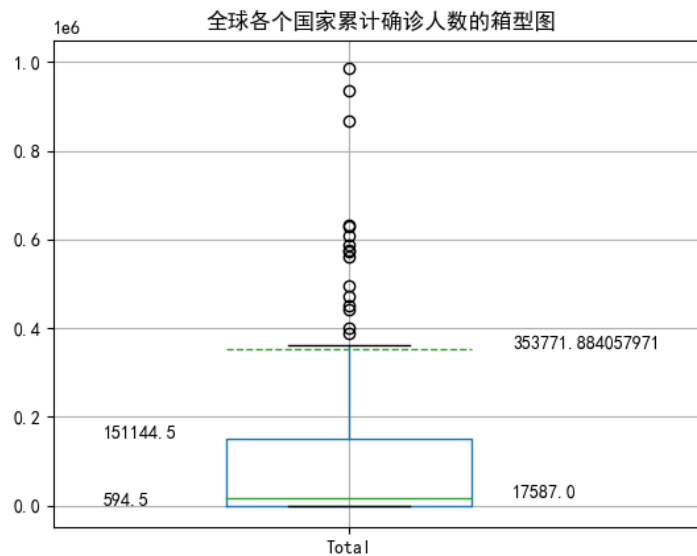


图 12

```

1 import pandas as pd
2 import matplotlib.pyplot as plt
3
4 plt.rcParams['font.sans-serif'] = ['SimHei'] # 用来正常显示中文标签
5 # 打开CSV文件
6 fileNameStr = 'total.csv'
7 df = pd.read_csv(fileNameStr, encoding='utf-8')
8 df_15 = df[df['Time'] == "2020-12-15 18:25"]
9 # 第15天的全球数据位于第2912行
10 df_15.drop([2912], inplace=True) # 除去全球的数据只留下各个国家的累
    计确诊人数
11 # 添加文本注释
12 ax = df_15.boxplot(column=['Total'], meanline=True, showmeans=True,
    vert=True) # 修改True的设置
13 ax.set_title('全球各个国家累计确诊人数的箱型图')
14 ax.text(1.1, df_15['Total'].mean(), df_15['Total'].mean())
15 ax.text(1.1, df_15['Total'].median(), df_15['Total'].median())
16 ax.text(0.85, df_15['Total'].quantile(0.25), df_15['Total'].quantile
    (0.25))
17 ax.text(0.85, df_15['Total'].quantile(0.75), df_15['Total'].quantile
    (0.75))
18 plt.show()

```

2.8 全球累计确诊人数前 10 名国家分布

根据 2.2 的累计确诊排名数据，画出全球累计确诊人数前 10 名国家分布，如图13所示。可以看出累计确诊人数多的国家主要分布在欧洲，南美洲，北美洲，且该国家多是发达的人流量较多的国家。在比例上，欧洲在全球累计确诊人数前 10 名国家中占了 5 个，是比例最大的一个州，主要爆发于西欧地区。从人数上看，北美洲的确诊人数较多。

全球累计确诊人数前10名国家分布

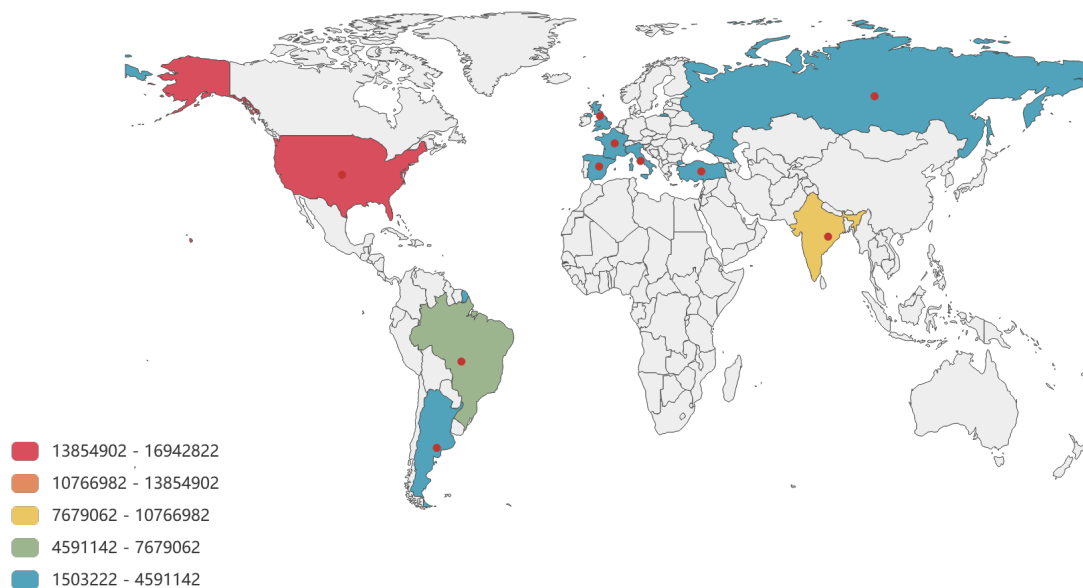


图 13: 全球累计确诊人数前 10 名国家分布

代码如下

```
1 from pyecharts.charts import Map
2 from pyecharts import options as opts
3 from pyecharts.render import make_snapshot
4 from snapshot_phantomjs import snapshot
5
6 # 全球累计确诊人数前10名国家数据
7 country = ['United States', 'India', 'Brazil', 'Russia', 'France',
8            'United Kingdom', 'Turkey', 'Italy', 'Spain', 'Argentina']
9 data_world = [16942822, 9906165, 6927145, 2682866, 2433859,
10              1874867, 1866345, 1855737, 1762036, 1503222]
11 world = (
12     Map()
13     .add('',
14         [list(z) for z in zip(country, data_world)], # 数据
```



```

15         'world') # 地图类型
16     .set_global_opts(
17         title_opts=opts.TitleOpts(title='全球累计确诊人数前10名国家分
18         布'),
19         visualmap_opts=opts.VisualMapOpts(
20             max_=16942822,
21             min_=1503222,
22             is_pieewise=True) # 定义图例为分段型，默认为连续的图例
23     )
24     .set_series_opts(label_opts=opts.LabelOpts(is_show=False))
25 world.render(path='全球累计确诊人数前10名国家分布.html') # 生成地图
26 make_snapshot(snapshot, world.render(), "全球累计确诊人数前10名国家分
    布.png")

```

3 应对新冠疫情最好的 10 个国家

为了给出应对新冠疫情最好的 10 个国家，我们考虑如下四个因素 (各个因素的权重在表中给出)

- 死亡率。死亡率可以表示一个国家应对新冠疫情的医疗水平。
- 累计确诊人数占国家总人口，该因素可以体现疫情在该国家的爆发程度，从而看出该国家对于疫情的防控是否有效。
- 治愈率。治愈率是治愈人数除以累计确诊人数，从该因素可以看出该国家对于治疗新冠肺炎的方法是否正确，是否正确认识到该病毒的感染原理。
- 新增人数占国家总人口。该因素可以看出目前该国家的疫情控制程度，也可以看出该国家采取的一系列防疫措施是否取得一定的成效。

根据这四个因素计算每个国家的总得分，并结合实际情况给出应对新冠疫情最好的 10 个国家。计算得分的过程如下，设这四个因素的权重分别为 w_1, w_2, w_3, w_4 ，参与排名的国家总数为 M ，考虑某个国家在这四个因素的排名分别为 n_1, n_2, n_3, n_4 ，该排名是利用 `rank()` 函数，为各组分配的一个平均排名，则 $\frac{M-n_1}{M} * w_1 + \frac{M-n_2}{M} * w_2 + \frac{M-n_3}{M} * w_3 + \frac{M-n_4}{M} * w_4$ 即为这个国家的得分。

表 1: 考虑因素及其权重

死亡率	累计确诊人数占国家总人口	治愈率	新增人数占国家总人口
0.2	0.35	0.2	0.25

计算各个国家的得分，结果如下：

	Country	PeopleNum	NewAdd	...	add_rate	add_rate_rank	score
16	泰国	69183173	9	...	0.000000	29.5	0.799864
75	贝宁	11485674	0	...	0.000000	29.5	0.794293
114	新西兰	4749598	0	...	0.000000	29.5	0.786277
48	科特迪瓦	24905843	25	...	0.000001	60.5	0.769429
47	马达加斯加	26262810	0	...	0.000000	29.5	0.758832
69	几内亚	13052608	26	...	0.000002	64.0	0.731793
12	越南	96491146	5	...	0.000000	29.5	0.730027
88	塔吉克斯坦	9107211	37	...	0.000004	74.5	0.702853
43	莫桑比克	30528673	48	...	0.000002	64.0	0.701630
138	几内亚比绍	1907268	0	...	0.000000	29.5	0.696060
0	中国	1400050000	105	...	0.000000	29.5	0.689538
45	加纳	29463643	256	...	0.000009	91.0	0.688315
6	尼日利亚	195875237	199	...	0.000001	60.5	0.683832
71	卢旺达	12501156	88	...	0.000007	87.5	0.678125
142	赤道几内亚	1313894	0	...	0.000000	29.5	0.677853
103	新加坡	5791901	16	...	0.000003	68.5	0.673234
40	乌兹别克斯坦	32364996	147	...	0.000005	80.0	0.671739

考虑到部分国家地处偏远，人流量不大，感染人数只有几百个甚至只有几十个，这些国家虽然得分较高，但是不能体现该国家应对新冠疫情的表现，故本文结合实际情况和国家得分给出应对新冠疫情最好的 10 个国家如下：中国，新加坡，越南，新西兰，泰国，马达加斯加，尼日利亚，莫桑比克，乌兹别克斯坦，塔吉克斯坦。

- 中国。疫情最早爆发在中国，而且是在春运，这一人流量最多的时间节点爆发，但是中国抗击疫情能力非常突出，经过全国人民的努力，疫情得到了有效的控制。中国在累计确诊人数占国家总人口方面排名 17，新增人数占国家总人口排名 29.5，这些数据都说明了中国已经基本控制住了疫情，虽然在死亡率排名 171 不是很理想，但是鉴于中国是最早爆发地，没有新冠病毒的治疗经验，因此死亡率较高在接受范围之内。
- 新加坡。新加坡的医疗水平在这次疫情期间体现的淋漓尽致，死亡率排名 18，治愈率达到 99.77%，排名第 8，很大程度上保障了确诊病人的生命健康。低死亡率和治愈率都体现出新加坡在应对这次疫情方面表现优异。
- 越南。越南在累计确诊人数占国家总人口方面表现优异，平均排名第 10，在总人口 96491146 的情况下，只有 1402 人累计确诊，同时新增人数占国家总人口排名 29.5，说明了越南有效的控制住了疫情的爆发。
- 新西兰。新西兰在治愈率方面同样表现优异，治愈率 96.14%，排名 26；在新增人数占国家总人口平均排名 29.5；表现相对最差的死亡率方面，排名为 62，为中上水平，新西兰在这四个考虑因素下，没有特别明显的短板。
- 泰国。泰国在累计确诊人数占国家总人口方面表现优异，平均排名第 15，在总人口 69183173 的情况下，只有 4246 人累计确诊，同时新增人数占国家总人口排名

29.5, 12月15日仅仅新增9名感染者, 说明了泰国也有效的控制住了疫情的爆发。

- 马达加斯加。马达加斯加在治愈率和新增人数占国家总人口表现优异, 治愈率96.61%, 排名23, 同时新增人数占国家总人口排名29.5。
- 尼日利亚。尼日利亚在四个方面表现均衡, 累计确诊人数占国家总人口排名39, 其他方面排名均为60多, 说明有效的控制住了疫情。
- 莫桑比克。莫桑比克在死亡率和累计确诊人数占国家总人口表现优异, 死亡率只有1.4727%, 说明该国家对于疫情的治疗非常有效, 治愈率和新增人数占国家总人口方面表现也不赖, 排名60。
- 乌兹别克斯坦。乌兹别克斯坦的治愈率和死亡率的排名也很高, 治愈率96.3863%, 死亡率0.8134%, 这两项排名分别为24和38。
- 塔吉克斯坦。塔吉克斯坦在新增人数占国家总人口方面表现优异, 12月15日, 增加确诊人数仅为37人, 同样的治愈率和死亡率也十分理想, 治愈率95.53%, 死亡率0.69%, 排名靠前。

为各个国家打分代码如下:

```
1 import pandas as pd
2
3 # 打开疫情数据CSV文件
4 fileNameStr = 'total.csv'
5 df_result = pd.read_csv(fileNameStr, encoding='utf-8')
6 # 打开人口数量CSV文件
7 fileNameStr = 'PeopleNum.csv'
8 df_people_num = pd.read_csv(fileNameStr, encoding='utf-8')
9 # 选出12月15日的疫情数据
10 df_15 = df_result[df_result['Time'] == "2020-12-15 18:25"]
11 # 连接合并国家的疫情数据与国家人口数据
12 df_all = pd.merge(df_people_num, df_15, how='inner', on=['Country'])
13 # 将人口数据的逗号去掉 例如 123,456 变成 123456
14 for i in range(len(df_all['PeopleNum'])):
15     df_all['PeopleNum'][i]=str(df_all['PeopleNum'][i]).replace(",","")
16 # 转成int
17 df_all['PeopleNum'] = df_all['PeopleNum'].astype('int')
18 # 计算累计确诊人数占国家总人口比例
19 df_all['total_rate'] = round(df_all['Total']/df_all['PeopleNum'], 6)
20 # 计算各个国家死亡率并利用rank()函数平均排名
21 df_all['dead_rate'] = round(df_all['Dead'] / df_all['Total'], 6)
22 df_all["dead_rate_rank"] = df_all["dead_rate"].rank(method="average")
```

```

23 # 计算各个国家累计确诊人数占国家总人口并利用rank()函数平均排名
24 df_all['total_rate'] = round(df_all['Total']/df_all['PeopleNum'], 6)
25 df_all["total_rate_rank"] = df_all["total_rate"].rank(method="average")
26 # 计算各个国家治愈率并利用rank()函数平均排名
27 df_all['cure_rate'] = round(df_all['Cure'] / df_all['Total'], 6)
28 df_all["cure_rate_rank"] = df_all["cure_rate"].rank(ascending=False,
    method="average")
29 for i in range(len(df_all['NewAdd'])):
30     if df_all['NewAdd'][i] == '-':
31         df_all['NewAdd'][i] = 0
32 df_all["NewAdd"] = df_all["NewAdd"].astype('int')
33 # 计算各个国家新增人数占国家总人口的比率并利用rank()函数平均排名
34 df_all['add_rate'] = round(df_all['NewAdd'] / df_all['PeopleNum'], 6)
35 df_all["add_rate_rank"] = df_all["add_rate"].rank(method="average")
36 # 计算分数 一个有184个国家
37 df_all["score"] = (184 - df_all["dead_rate_rank"]) / 184 * 0.2 + (184
    - df_all["cure_rate_rank"]) / 184 * 0.2 + (184 - df_all["
    total_rate_rank"]) / 184 * 0.35 + (184 - df_all["add_rate_rank"])
    / 184 * 0.25
38 df_all.sort_values(by='score', ascending=False)
39 # 输出累计确诊人数占国家总人口比例最高的 10 个国家
40 pd.set_option('display.max_columns', None) # 输出完整列
41 pd.set_option('display.width', None) # 宽度无限
42 pd.set_option('display.max_rows', None) # 显示所有行
43 df_all = df_all.sort_values(by='score', ascending=False) # 分数排名
44 df_all.to_csv('score.csv')
45 # 输出累计确诊人数大于1000的国家
46 print(df_all[df_all['Total'] > 1000])

```

4 预测分析

画出全球 12 月 1 日至 10 日的新冠肺炎累计确诊人数的散点图，如图14. 由该散点图可以看出，全球新冠肺炎累计确诊人数大致呈现线性增长，故采用线性拟合的方法做后 5 天的预测. 预测结果函数为 $y = 628699.6x + 62910931.8$ ，其中 x 为日期，即 12 月的几日， y 为累计确诊人数，如图15所示.

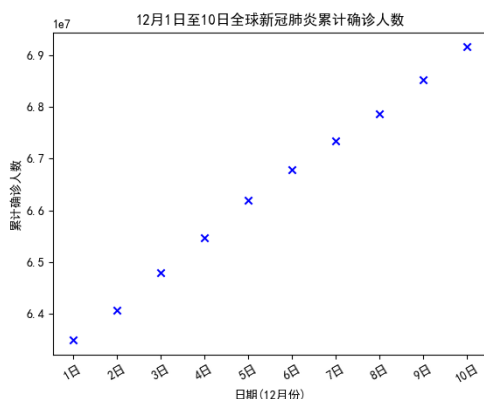


图 14

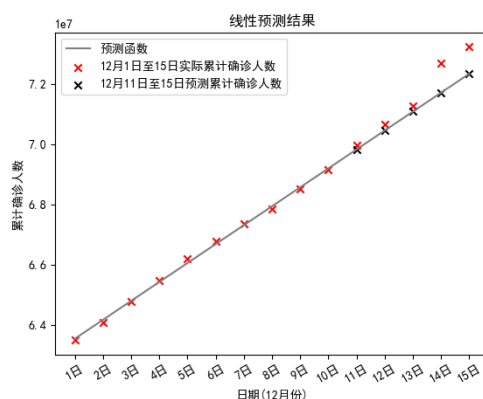


图 15

从图中可以看出，12月11日至13日的预测效果良好，而14日和15日的效果较差，我认为原因可能有如下两点：

- 由于14号爬取数据的时间是晚上10点，而其他日期的爬取时间是在下午5点左右，可能由于爬取时间的不同，正好赶上了网站刷新数据，所以14日的数据有一个飞跃的趋势，同时可以看出14日至15日的确诊人数增长趋势与1日至13日的趋势相似，所以很有可能是爬取14日数据的时间与其他数据的爬取时间不同，才会导致13日到14日的累计确诊人数有一个飞跃的增加，而14日至15日的增加趋势又和之前1日至13日的增加趋势相似。
- 疫情新增确诊人数的增加本来就有不确定因素，可能14日这天某个国家或地区的疫情再度爆发，导致新增的病例相较于之前增加很多，出现了13日到14日的累计确诊人数有一个飞跃的情况发生，也间接的导致了15日预测结果不准确。

预测数据代码如下

```
1 import pandas as pd
2 import matplotlib.pyplot as plt
3 import numpy as np
4
5 plt.rcParams['font.sans-serif'] = ['SimHei'] # 用来正常显示中文标签
6 fileNameStr = 'total.csv' # 打开文件
7 df_result = pd.read_csv(fileNameStr, encoding='utf-8')
8 df_world = df_result[df_result['Country'] == '世界'] # 选出世界数据
9 num = df_world['Total'].tolist() # 将累计确诊人数变成列表形式
10 Time = df_world['Time'].tolist() # 取出对于的时间
11 for i in range(0, len(Time)):
12     temp = str(Time[i]).split()[0]
13     Time[i] = int(temp.replace("2020-12-", ""))
14     # 从时间列中将对于的日期转换成整数
```

```

15 fig = plt.figure() # 创建画图窗口
16 ax1 = fig.add_subplot(1, 1, 1)
17 ax1.set_title("线性预测结果") # 直方图标题
18 ax1.set_ylabel('累计确诊人数')
19 ax1.set_xlabel('日期(12月份)')
20 date = [] # 设置x轴的标签
21 for i in range(len(Time)):
22     date.append(str(Time[i]) + '日')
23 plt.xticks(Time, date, rotation=30)
24 ax1.scatter(Time, num, color='red', marker='x', label='12月1日至15日
    实际累计确诊人数')
25 # 利用numpy的线性拟合
26 coeff = np.polyfit(Time[0:10], num[0:10], 1)
27 print(coeff)
28 x_1 = []
29 for i in range(11, 16, 1):
30     x_1.append(int(i))
31 x_1 = np.array(x_1)
32 # 画出预测人数的散点图
33 ax1.scatter(x_1, coeff[0] * x_1 + coeff[1], color='black', marker='x',
    , label='12月11日至15日预测累计确诊人数')
34 x_2 = []
35 for i in range(1, 16, 1):
36     x_2.append(int(i))
37 x_2 = np.array(x_2)
38 # 画出预测函数
39 ax1.plot(x_2, coeff[0] * x_2 + coeff[1], color='grey', label='预测函
    数')
40 ax1.legend()
41 plt.show()

```