Python期末大作业：分析新冠疫情数据

目录

[一. 题目要求 2](#_Toc31588)

[二. 数据来源 2](#_Toc22366)

[三. 数据分析和展示 3](#_Toc29571)

[1. 15天中，全球新冠疫情的总体变化趋势 3](#_Toc23621)

[2. 累计确诊数排名前 20 的国家名称及其数量 3](#_Toc16921)

[3. 15 天中，每日新增确诊数累计排名前 10 个国家的每日新增确诊数据的曲线图 4](#_Toc26945)

[4. 累计确诊人数占国家总人口比例最高的 10 个国家 4](#_Toc32527)

[5. 死亡率（累计死亡人数/累计确诊人数）最低的 10 个国家 5](#_Toc7238)

[6. 用饼图展示各个国家的累计确诊人数的比例 5](#_Toc26480)

[7. 展示全球各个国家累计确诊人数的箱型图，要有平均值 6](#_Toc8563)

[8. 其他展示 6](#_Toc21787)

[四. 根据以上数据，列出全世界应对新冠疫情最好的 10 个国家，并说明你的理由 7](#_Toc14459)

[五. 针对全球累计确诊数，利用前 10 天采集到的数据做后 5 天的预测 8](#_Toc26878)

[六. 核心代码 10](#_Toc1660)

[1. 爬虫程序（使用selenium） 10](#_Toc9246)

[2. 各国应对措施打分代码 14](#_Toc8024)

[3. 数据展示代码 15](#_Toc19963)

[4. 数据预测代码 25](#_Toc19212)

#### 题目要求

分析新冠疫情数据：找一个有全球新冠病毒数据的网站，爬取其中的数据（禁止使用数据接口直接获取数据）。要求爬取从 2020 年 12 月 1 日开始的连续 15 天的数据，国家数不少于 100 个。

#### 数据来源

初步观察发现，各网站的疫情数据各不相同，为保证最终数据分析不受网站因素影响，爬取数据使用的数据来源有三个。并在爬取到一定量的数据后确定使用哪个数据源。

一是百度疫情实时大数据报告([https://voice.baidu.com/act/newpneumonia/newpneumon](https://voice.baidu.com/act/newpneumonia/newpneumonia)

[ia](https://voice.baidu.com/act/newpneumonia/newpneumonia)),一个是全球新冠疫情大数据分析平台(<https://www.zq-ai.com/#/fe/xgfybigdata>)，最后一个是腾讯(<https://news.qq.com/zt2020/page/feiyan.htm#/global?nojump=1>)。首页截图如图1：

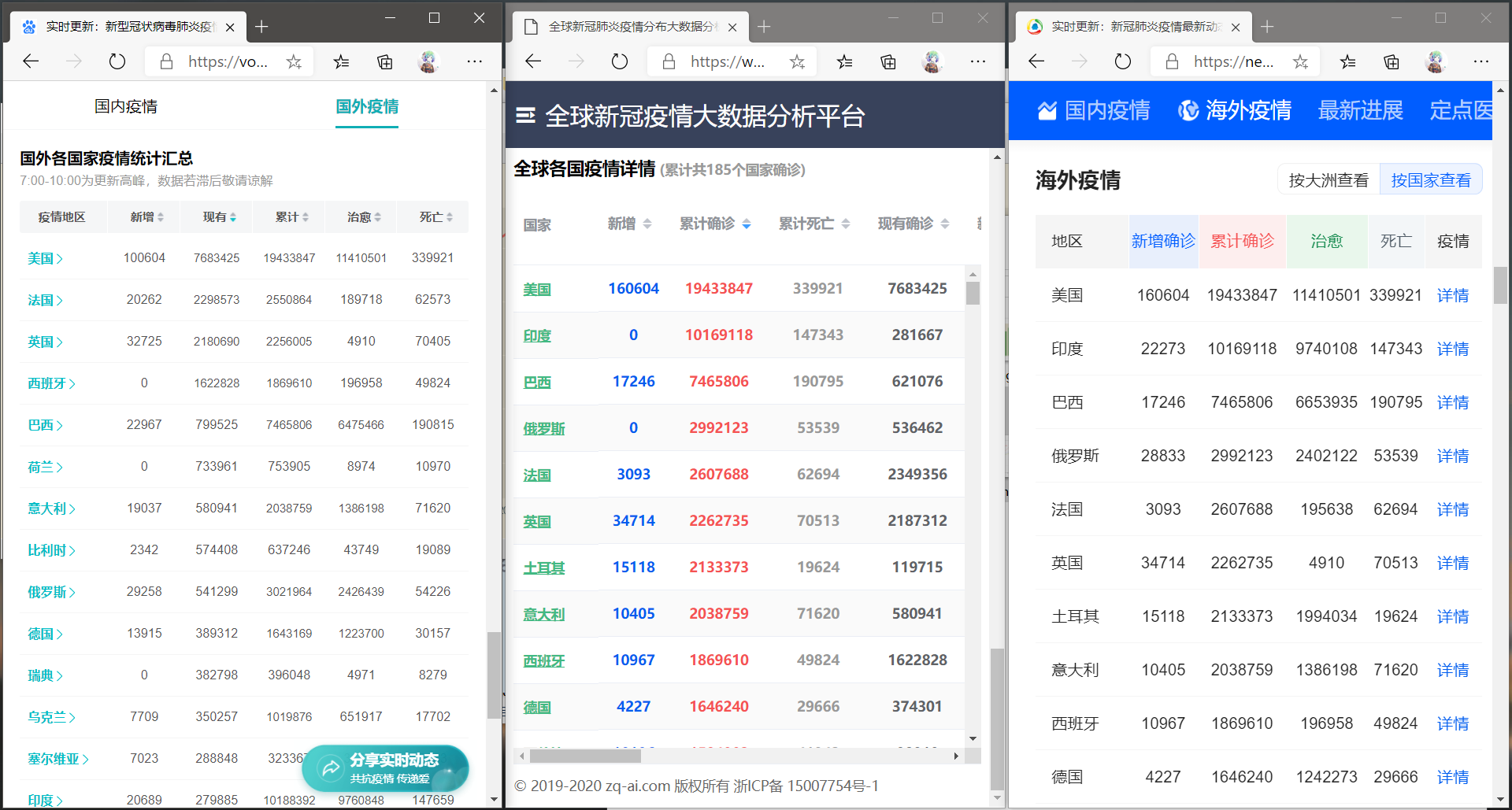


图1 数据来源：左图是百度，中图是大数据分析平台，右图是腾讯

进行数据分析时，首先根据数据质量判断要使用哪个数据源。通过作图可以看到，百度的数据更新存在明显的滞后以及更新不及时的情况，就如下图图2中前三天数据无任何变化。与此同时也看到另一个数据源无此类问题。因此决定使用第二个网站的数据。

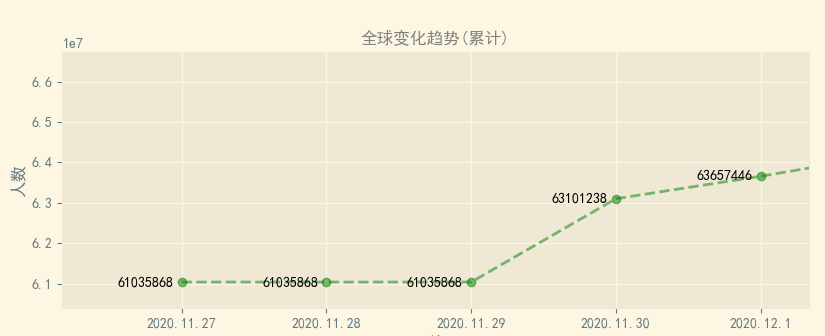


图2 百度数据出现更新停滞的现象

此外，值得说明的是，为保证数据完全更新，每天爬取数据的时间为晚上11点45分。

#### 数据分析和展示

##### 15天中，全球新冠疫情的总体变化趋势



图3 15天中，全球新冠疫情的总体变化趋势

通过以上两幅线图可以直观地认识到，全球累计确诊数的总体变化趋势基本符合线性增长；每日新增在50000与70000之间波动。

##### 累计确诊数排名前 20 的国家名称及其数量

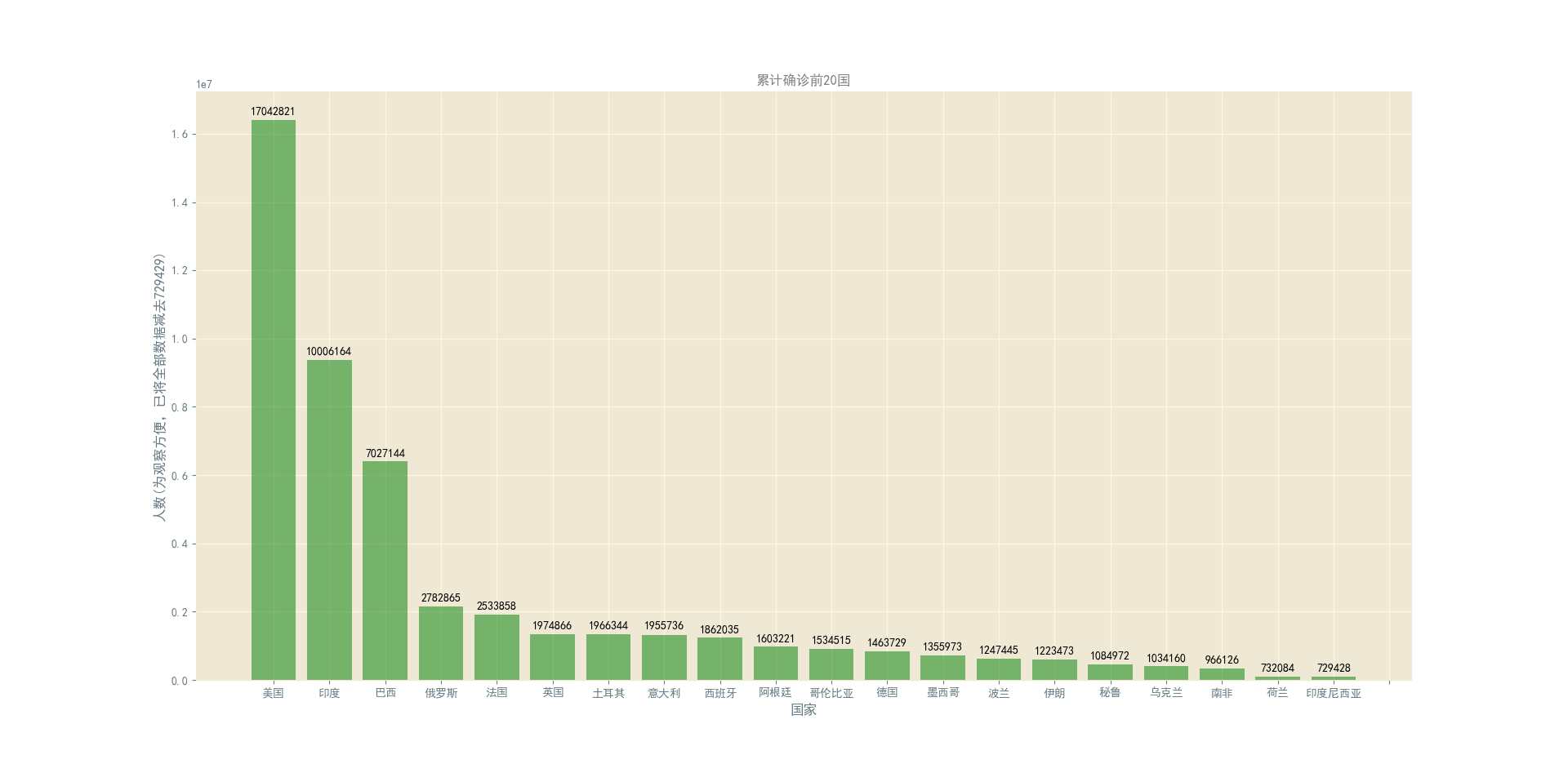


图4 累计确诊数排名前 20 的国家名称及其数量

由上图的柱状图看到，累计确诊最多的是美国，其次是印度、巴西，这三个国家占据了累计确诊数的大部分；接着是一些欧洲国家如俄罗斯、法国、英国。

##### 15 天中，每日新增确诊数累计排名前 10 个国家的每日新增确诊数据的曲线图

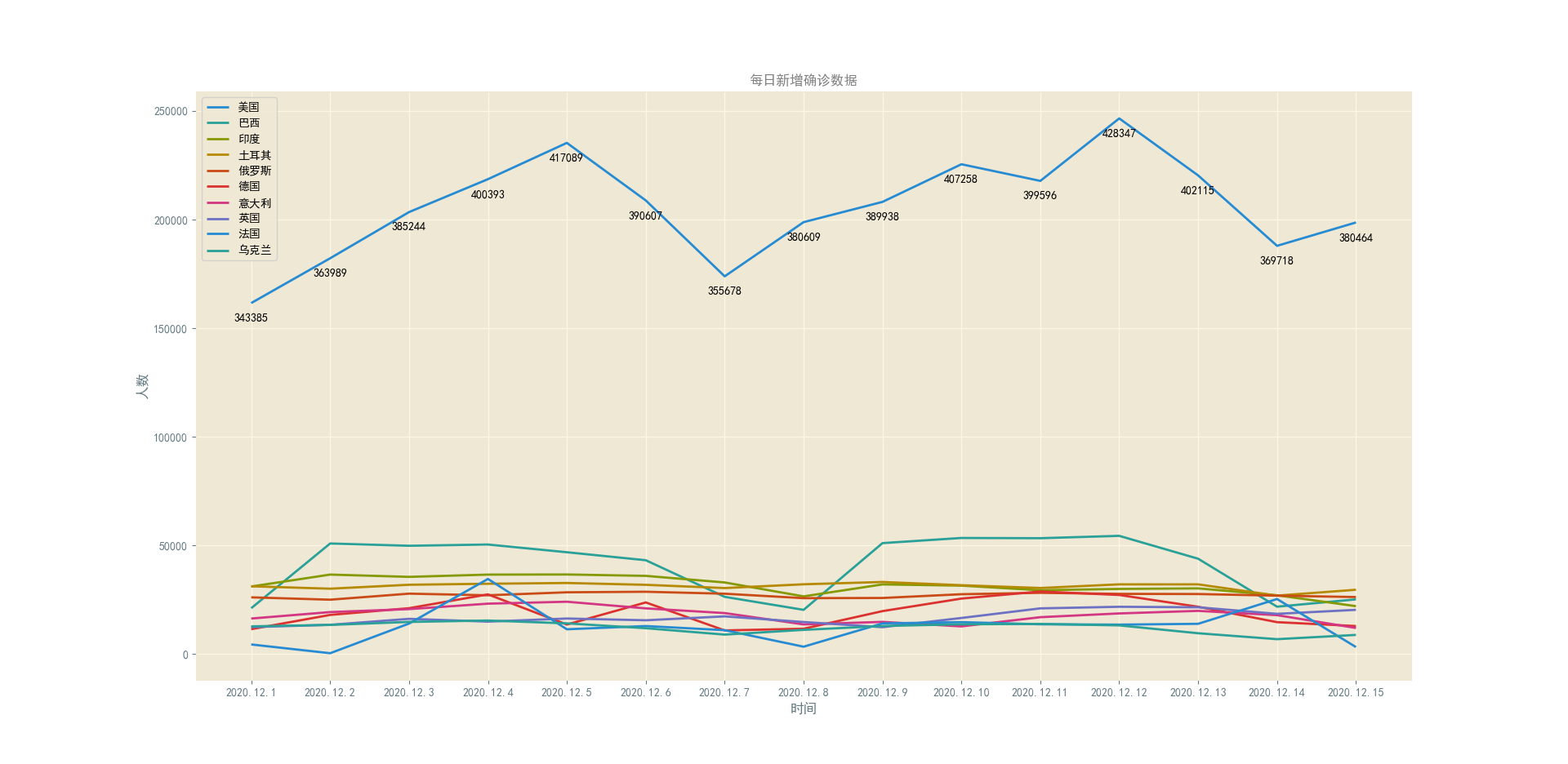


图5 15 天中，每日新增确诊数累计排名前 10 个国家的每日新增确诊数据的曲线图

由于国家过于密集，故只在距离其他国家较远的”美国”曲线上标注出具体数值。

在曲线图中也可以看出，美国的每日新增远大于其他国家，对照图3的全球每日新增数据发现每天的新增确诊人数有约60%来自美国。

此外，观察图3与图5的曲线，其大致走向是相同的。

##### 累计确诊人数占国家总人口比例最高的 10 个国家

通过数据处理，得到占比最高的国家是：安道尔, 卢森堡, 黑山, 圣马力诺, 巴林, 比利时, 卡塔尔, 美国, 亚美尼亚, 格鲁吉亚。

并对这10个国家做出柱状图：

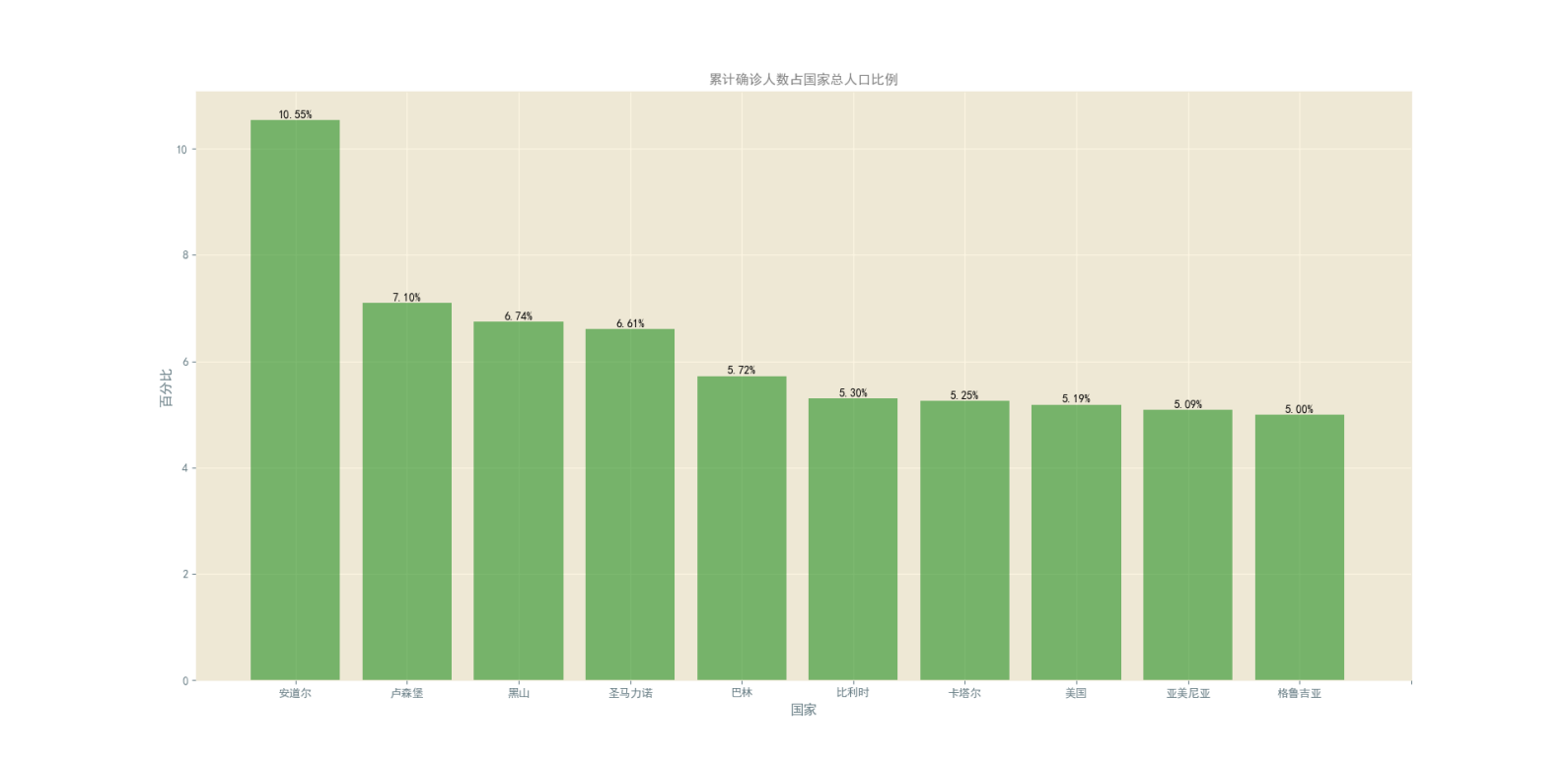


图6 累计确诊人数占国家总人口比例最高的 10 个国家

以第一个国家安道尔为例，猜测其确诊人口占比高的重要因素是国土面积小以及人口数少，导致一旦发生病例就会迅速波及到其大量人口。

##### 死亡率（累计死亡人数/累计确诊人数）最低的 10 个国家

通过数据处理，得到占比最高的国家是：新加坡, 布隆迪, 卡塔尔, 博茨瓦纳, 阿联酋, 马尔代夫, 巴林, 摩纳哥, 斯里兰卡, 马来西亚。

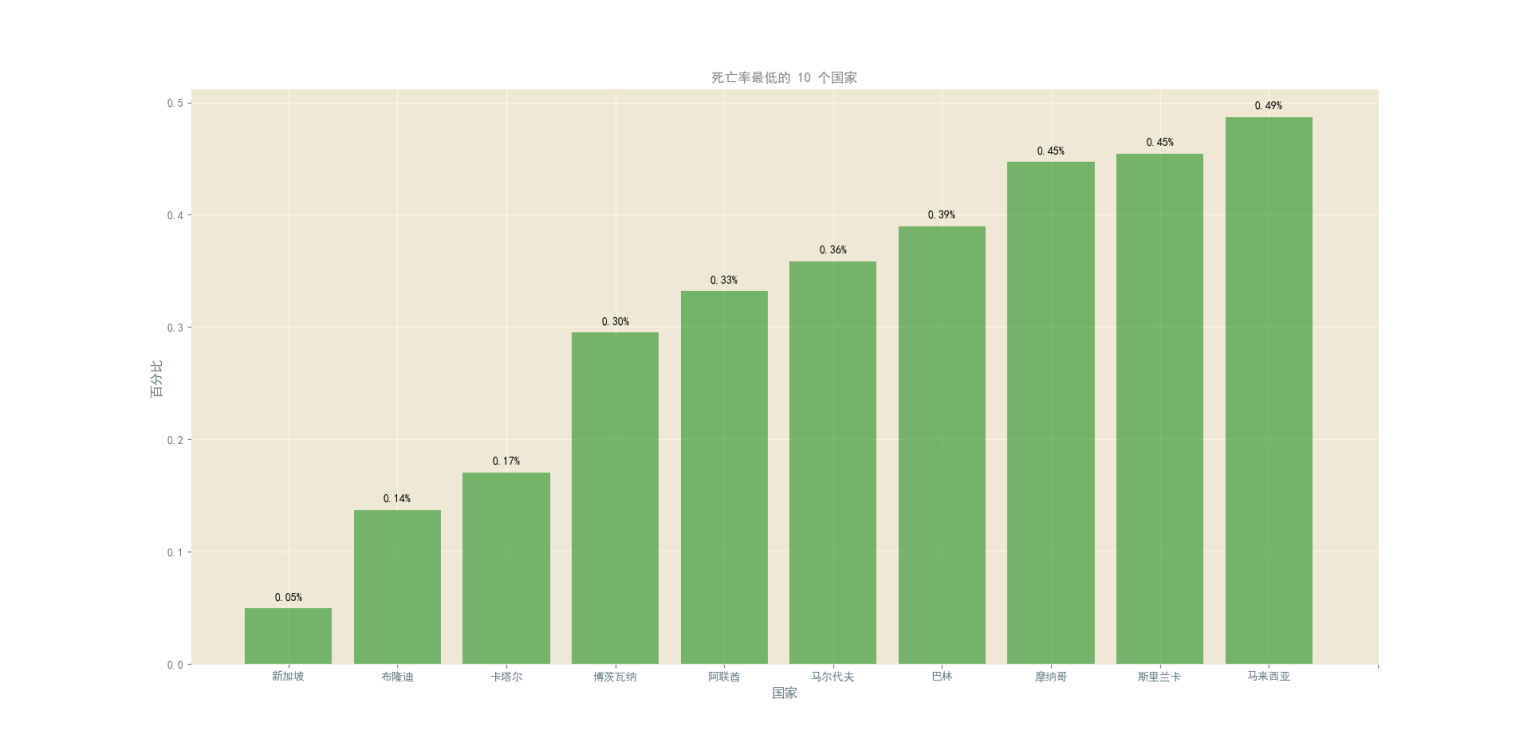


图7 死亡率（累计死亡人数/累计确诊人数）最低的 10 个国家

##### 用饼图展示各个国家的累计确诊人数的比例

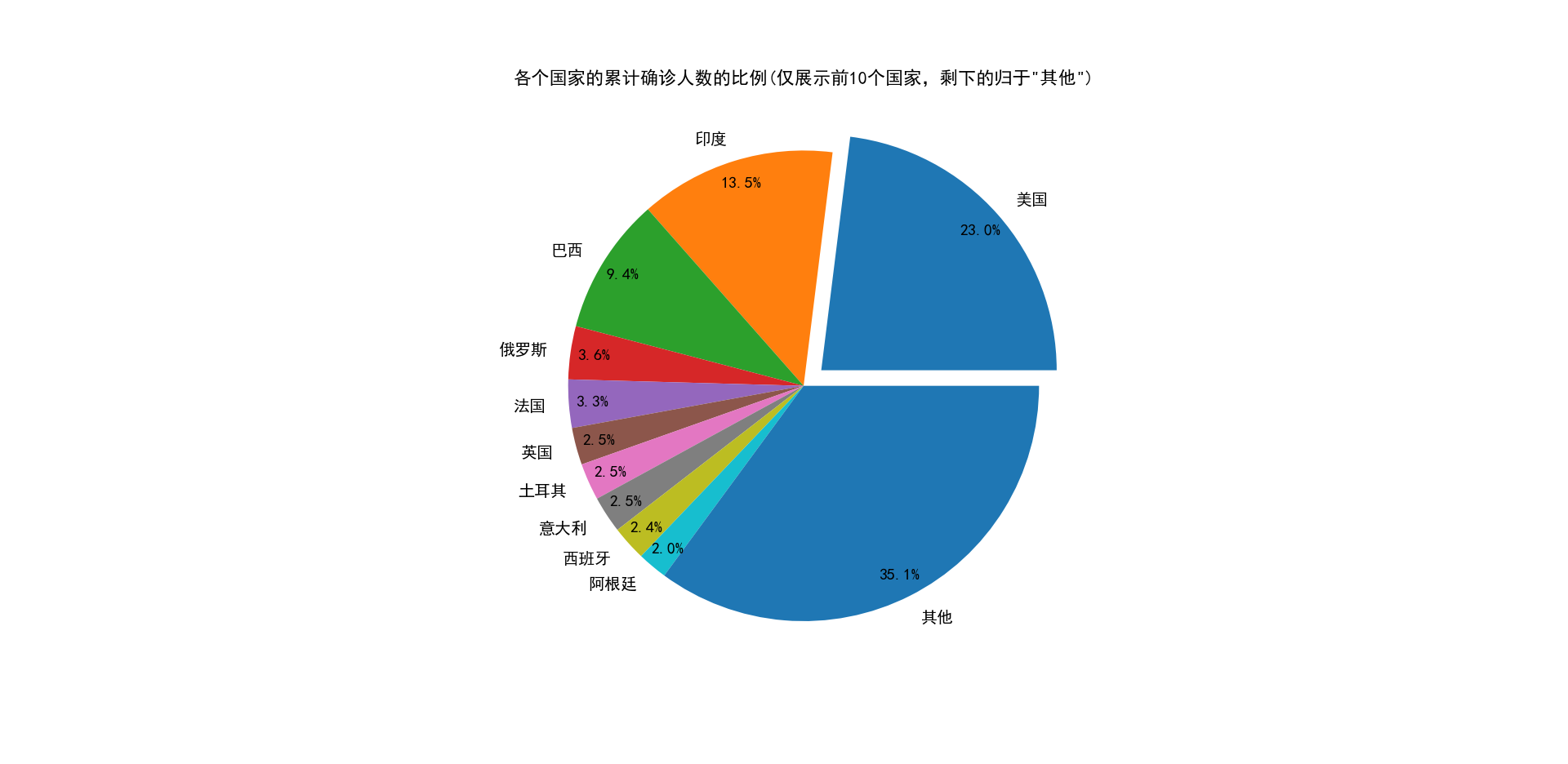


图8 各个国家的累计确诊人数比例的饼图

由饼图可以更加直观地看出，美国、印度、巴西、俄罗斯四国的累计确诊合计达到全球的一半。

##### 展示全球各个国家累计确诊人数的箱型图，要有平均值

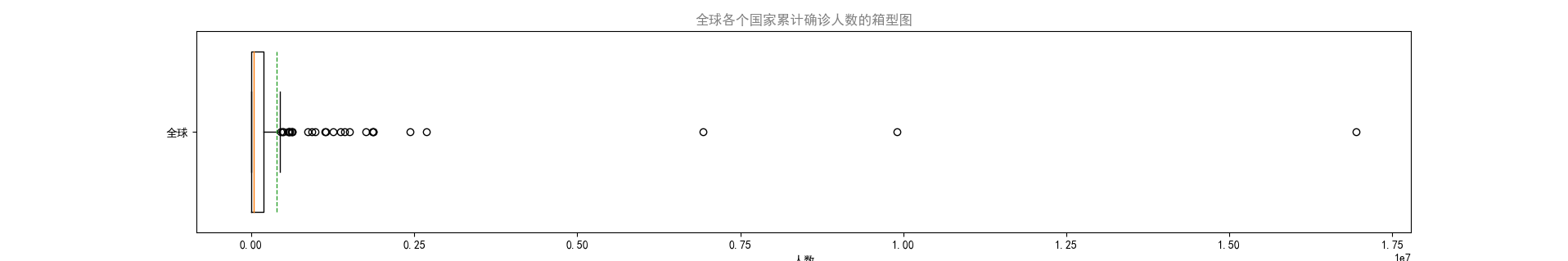


图9 含异常值的箱型图

由于各国累计确诊差距较大，故异常值较多，影响观感，将上图去掉异常值并添加必要的注释后，绘图如下:

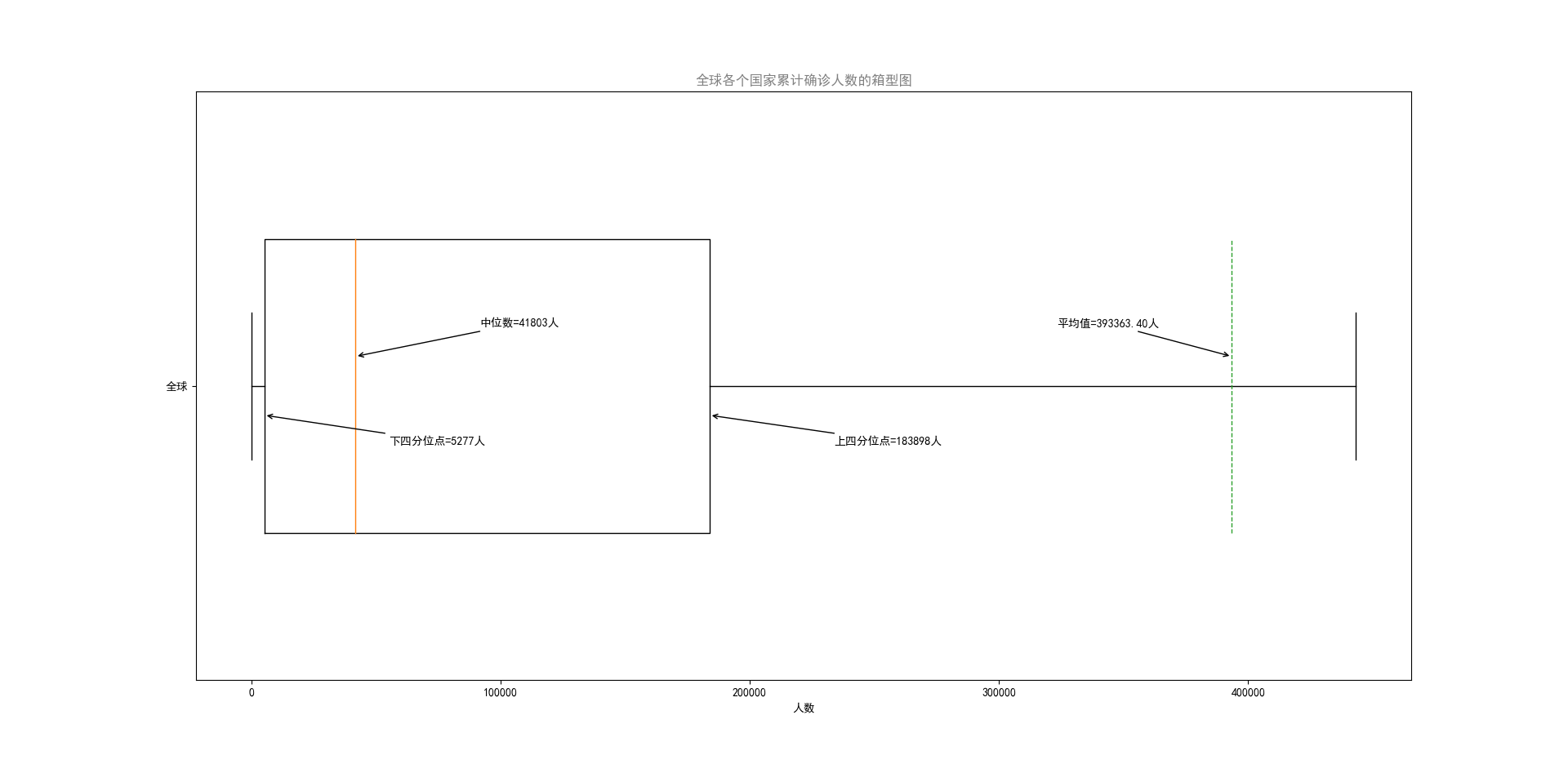


图10 不含异常值的箱型图

##### 其他展示

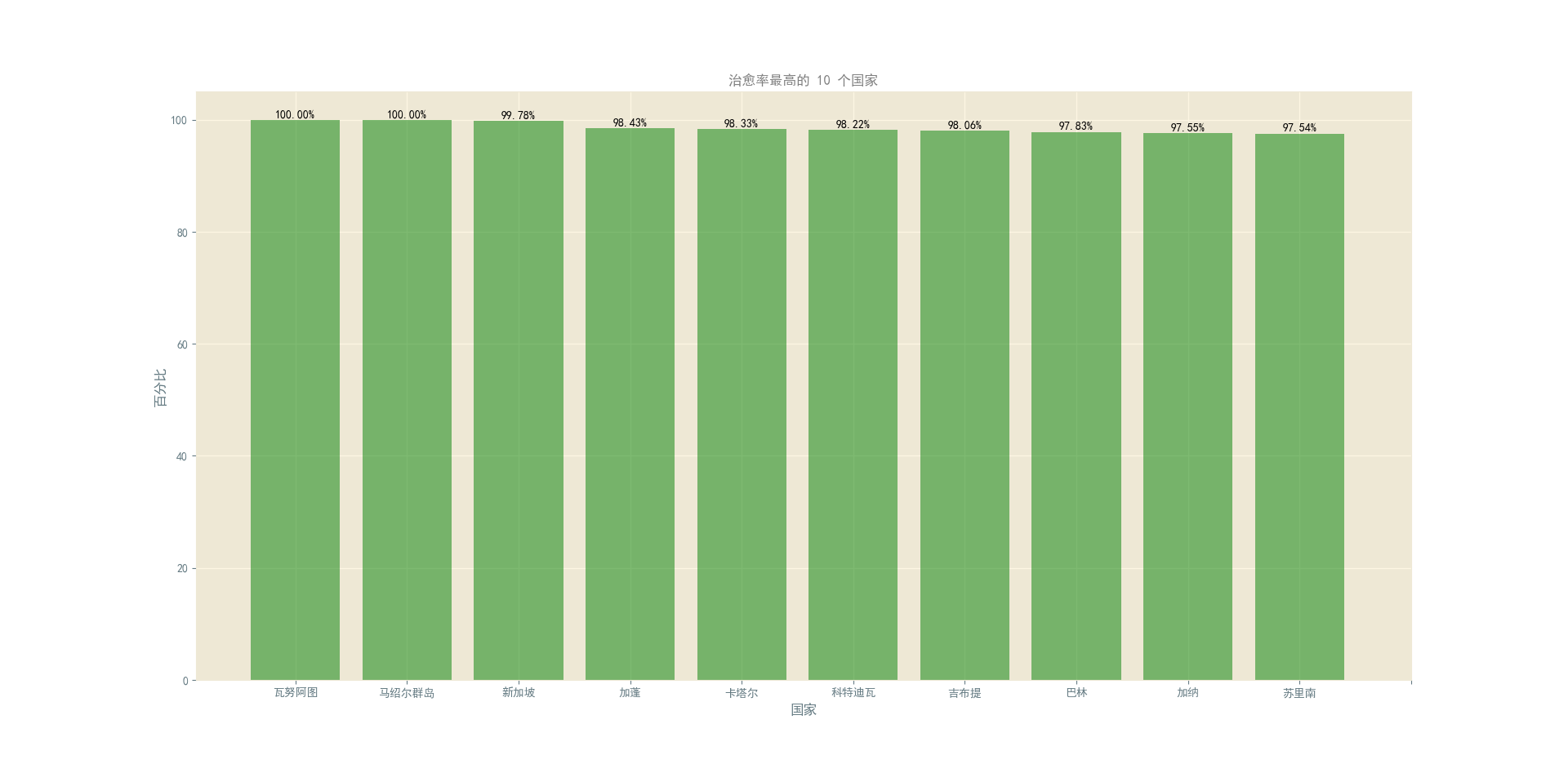


图11 治愈率最高的10个国家

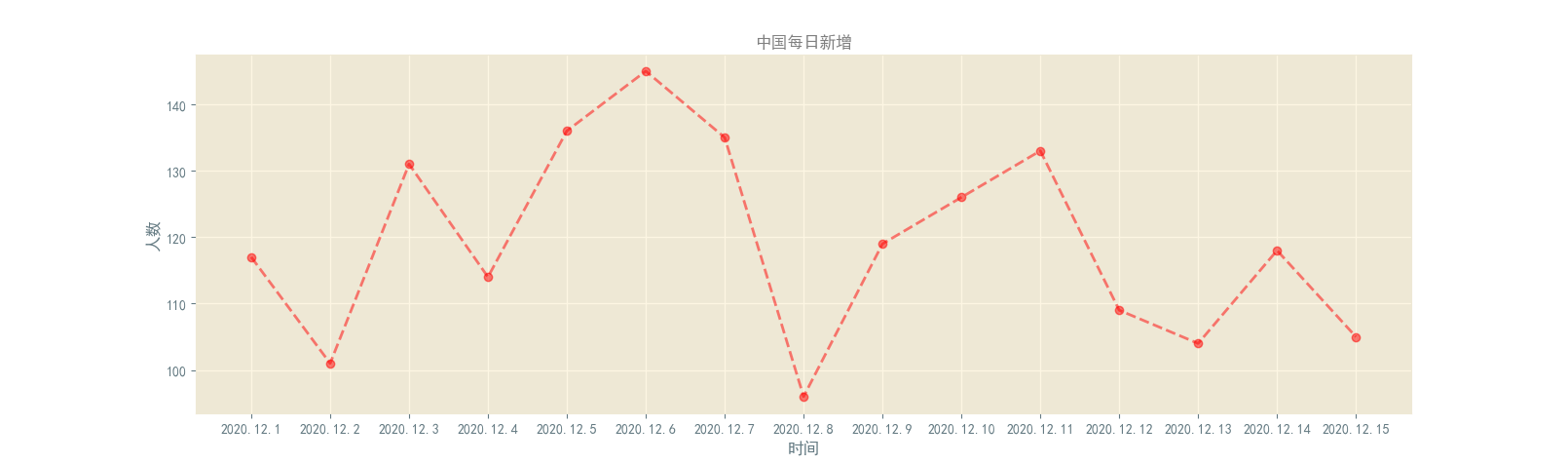


图12 中国每日新增线图

#### 根据以上数据，列出全世界应对新冠疫情最好的 10 个国家，并说明你的理由

对国家应对措施的评分采用加权计分，满分100，具体计算公式如下：

 对上式的解读如下：对疫情的应对应该体现在“对突发情况的处理以及进一步的对策”，体现在数据上就是累计确诊比例与新增确诊比例。故上式中这两项占比较大。

实际计算时，由于累计确诊比例，死亡率与新增确诊比例均较小，故还做了放大处理（即乘以合适的倍数）。

考虑到国与国之间人口数量差距较大，故采用百分比计算；同时由于部分国家人口过少，对数据影响较大，故不考虑总人口小于5000w的国家。

最终得出分数最高的国家如下：

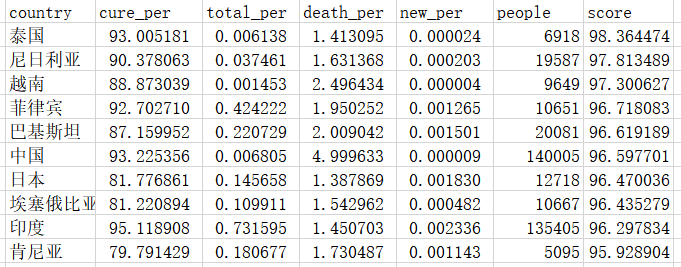


图13 应对较好的10个国家

观察上表，其中泰国优点在于累计确诊较少，治愈率也较高；越南这是累计确诊极少；中国优点在于治愈率较高，同时对新增确诊的控制较好。

总之，应对最好的10个国家是：泰国，尼日利亚，越南，菲律宾，巴基斯坦，中国，日本，埃塞俄比亚，印度，肯尼亚。

#### 针对全球累计确诊数，利用前 10 天采集到的数据做后 5 天的预测

考虑到数据量较小，预测效果可能不好，故除去上课介绍的线性回归预测方法外，还采用了两种对少量数据有较好预测效果的GM(1,1)灰度预测与k阶差分预测。

三种预测在前3天都有较好的预测效果，误差基本低于0.1%，其中GM(1,1)灰度预测与k阶差分预测效果优于线性回归；而之后两天则出现误差的明显增加，误差达到约1%：

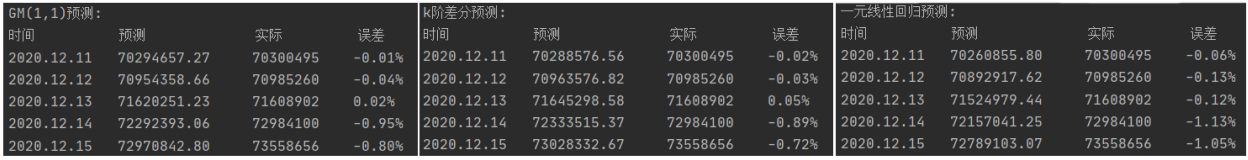


图14 三种预测比较

同时作图观察：

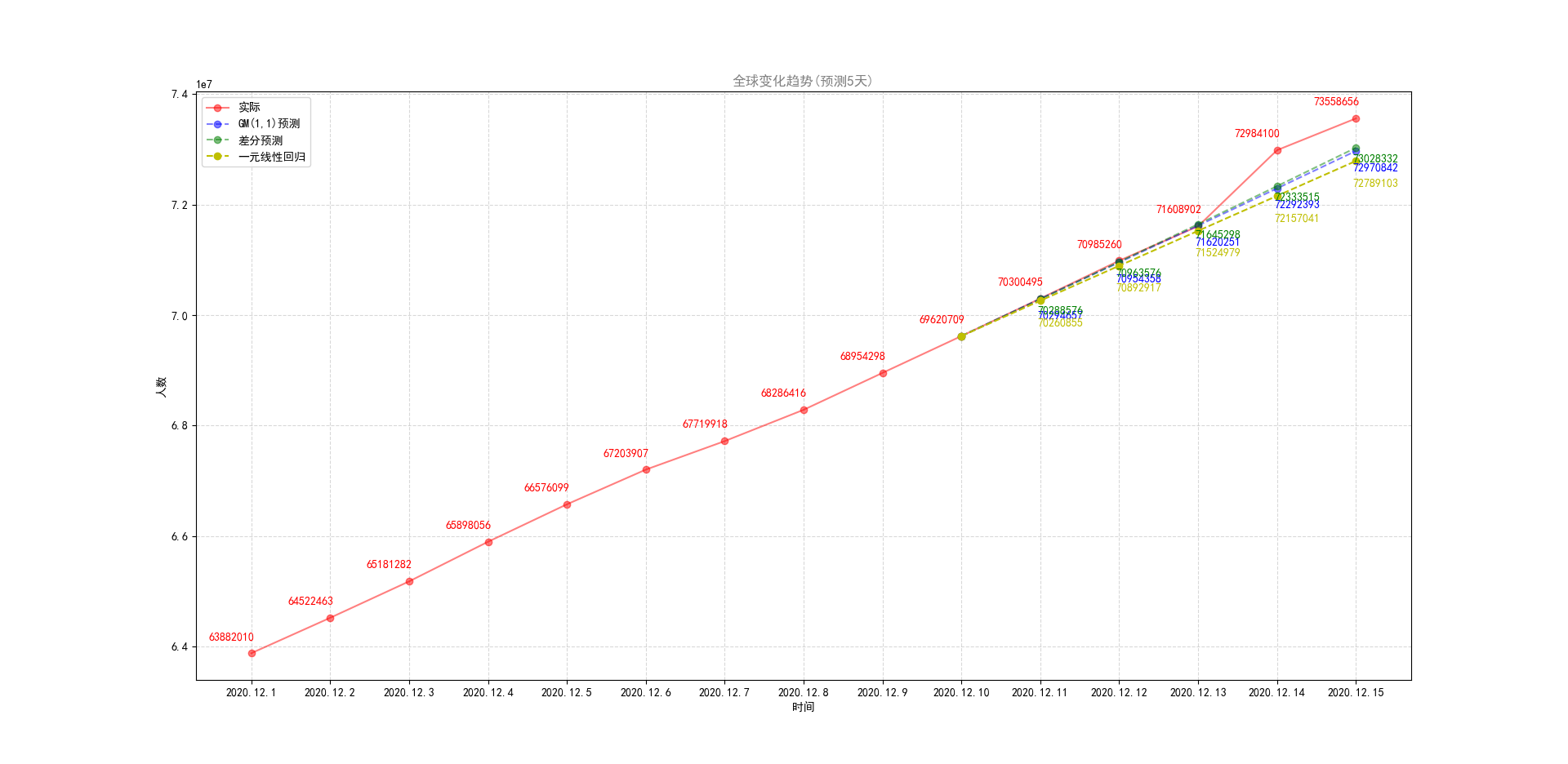


图15 三种预测结果与实际的对比（全局趋势）

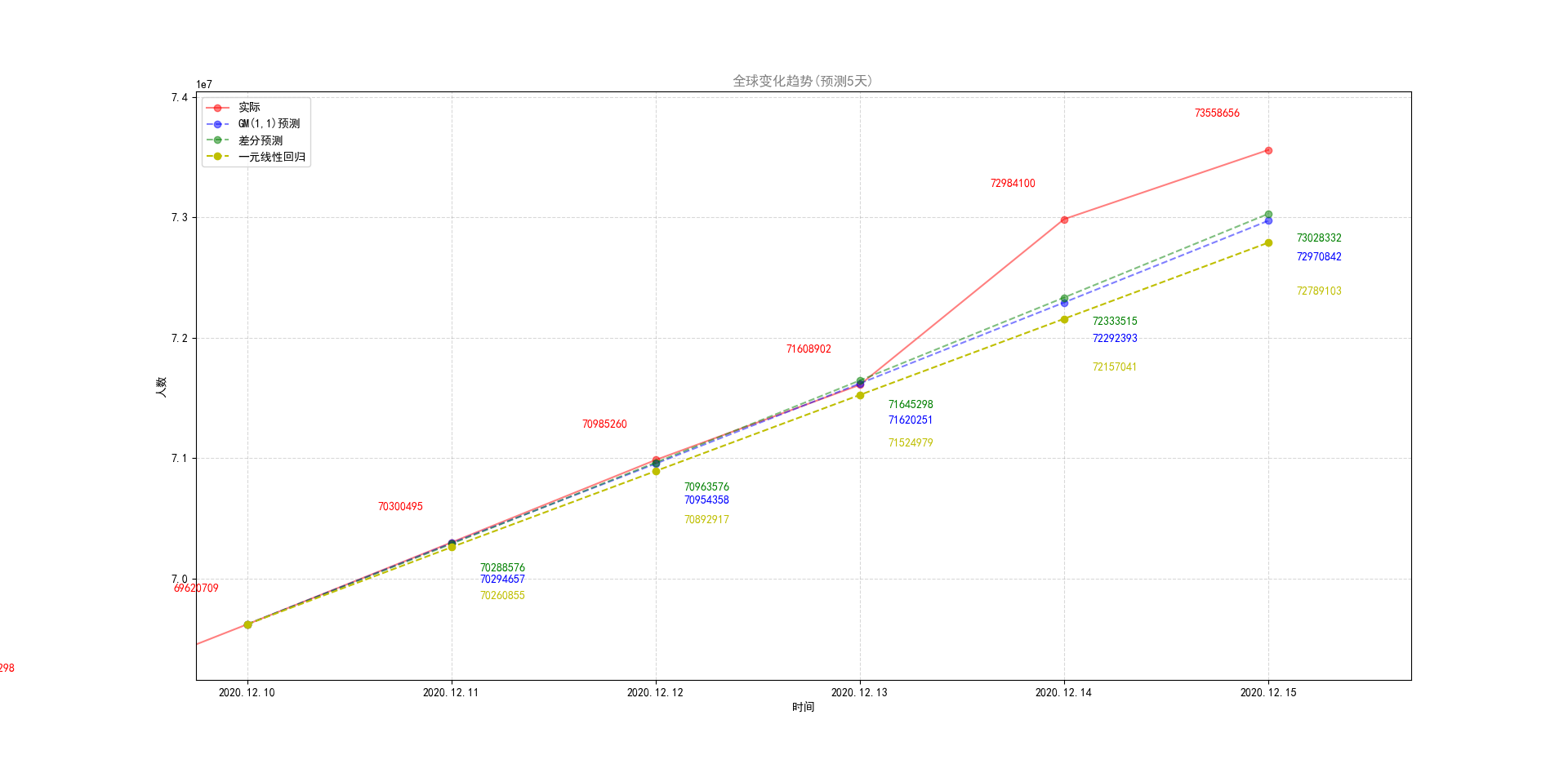


图16 三种预测结果与实际的对比（放大预测部分）

从图中可以看到，12月14日累计确诊人数突然发生一次较大的上升，破坏了前13天的规律，导致预测不准。

考察12月14日的新增数据（图3，5），并未发现新增人数明显增加；查阅新闻也未发现当天有集中疫情爆发的国家。同时注意到，网站数据并不满足“后一天累计确诊=前一天累计确诊+后一条新增”，也不满足“各国每日新增之和等于全球每日新增数据”。于是有理由猜测，网站数据中“每日新增”与“累计确诊”来源于不同组织/机构发布；而且，与我国积极检测并上报的政策不同，其他国家的数据真实性与准确性无法保证。

考察腾讯的数据如下图，走向基本相同：

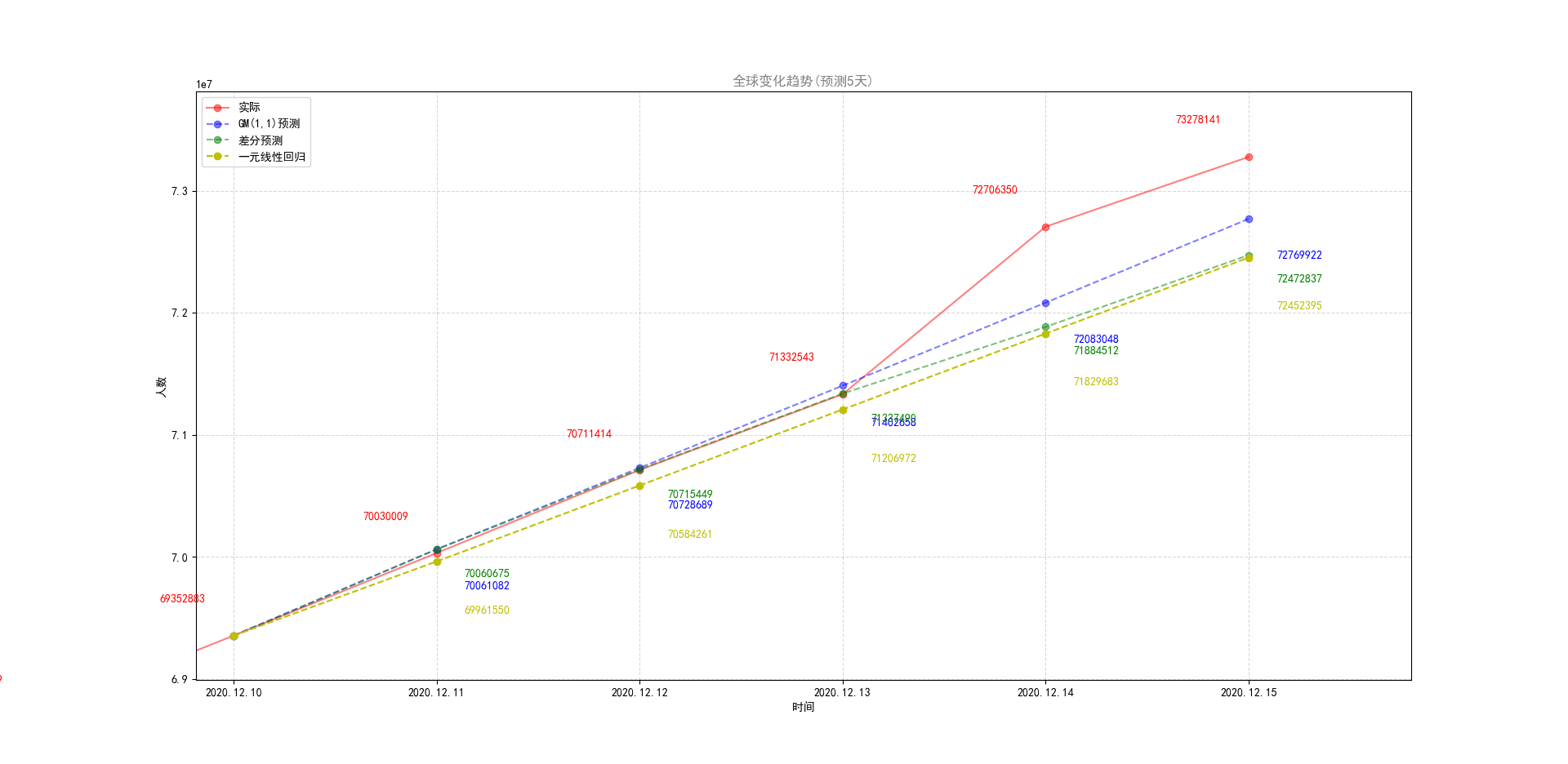


图17 三种预测结果与实际的对比（腾讯数据）

最后，查看数据网站<https://www.zq-ai.com/#/fe/xgfybigdata>的曲线图（图18），观察到12约12日土耳其新增确诊异常提升，提升幅度与我绘制的线图基本一致，我爬取的同一网站数据在12日也有较高的每日新增数。至于为何累计确诊在14日才表现出异常，我认为是数据源本身导致的不一致。

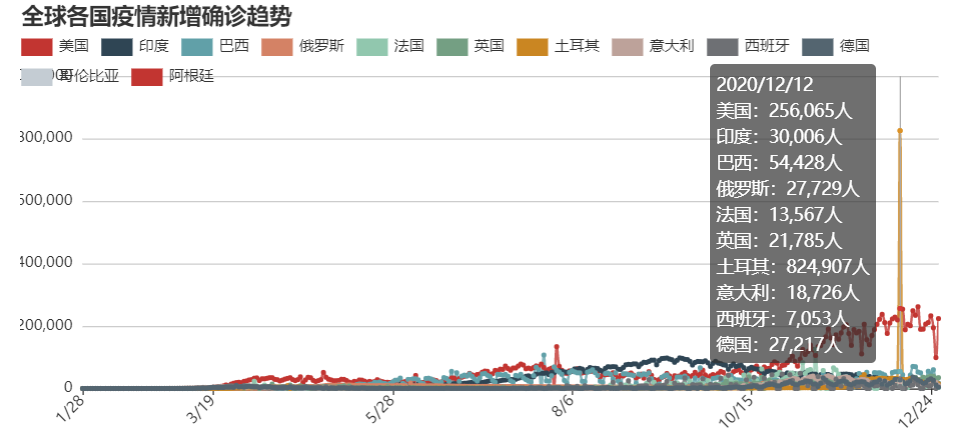


图18 数据网站https://www.zq-ai.com/#/fe/xgfybigdata的曲线图

综合来看，三种数据预测方式都很好地达到了预期效果，而12月14日的异常值属于现实世界中无法预测的突发事件。

#### 核心代码

##### 爬虫程序（使用selenium）

spider：

1. **import** scrapy
2. **from** selenium **import** webdriver
3. **from** selenium.webdriver.chrome.options **import** Options
4. **import** datetime
5. **from** cov19.items **import** Cov19Item
6. **from** scrapy **import** Request
8. **class** CovspiderSpider(scrapy.Spider):
9. name = 'covSpider'
10. start\_urls = ['https://voice.baidu.com/act/newpneumonia/newpneumonia']
11. all\_urls = ['https://voice.baidu.com/act/newpneumonia/newpneumonia',
12. 'https://news.qq.com/zt2020/page/feiyan.htm#/global?nojump=1',
13. 'https://www.zq-ai.com/#/fe/xgfybigdata']
14. showall = False
15. site = 0  # 指示当前正在爬取哪个网站
17. **def** \_\_init\_\_(self):
18. # 设定浏览器模式
19. chrome\_options = Options()
20. chrome\_options.add\_argument('--headless')       # 使用无头谷歌浏览器模式
21. chrome\_options.add\_argument('--disable-gpu')    # 禁用GPU加速
22. chrome\_options.add\_argument('--no-sandbox')
23. # 指定谷歌浏览器驱动路径
24. self.driver = webdriver.Chrome(chrome\_options=chrome\_options,
25. executable\_path='C:/Users/87290/Desktop/PROGRAM/Python/chromedriver.exe')
26. # 设定隐式等待时间10s
27. self.driver.implicitly\_wait(10)
28. # 输出提示信息
29. **print**('Webdriver start init success!')
31. **def** parse(self, response):
32. item = Cov19Item()
33. **print**('\*' \* 40)
34. **print**("开爬 开爬 fo~↑ fo~↑")
35. **if** self.site == 0:  # 如果是第一个网站(百度)
36. **print**("# Baidu:")
37. **print**("Get {} data".format(len(response.xpath('//\*[@id="foreignTable"]/table/tbody/tr/td/table/tbody/tr[\*]'))))
38. **for** each **in** response.xpath('//\*[@id="foreignTable"]/table/tbody/tr/td/table/tbody/tr[\*]'):
39. # 获取原始数据
40. country = each.xpath('td[1]/a/div[1]/text()').extract()
41. **if** len(country) == 0:
42. country = each.xpath('td[1]/div/text()').extract()
43. country = country[0]  # 获取国家名
44. dailyNew = each.xpath('td[2]/text()').extract()[0]  # 获取每日新增
45. now = each.xpath('td[3]/text()').extract()[0]  # 获取现有病例数
46. total = each.xpath('td[4]/text()').extract()[0]  # 获取累计确诊数
47. cure = each.xpath('td[5]/text()').extract()[0]  # 获取治愈数
48. death = each.xpath('td[6]/text()').extract()[0]  # 获取死亡数
50. # 装入item
51. item['year'] = datetime.datetime.now().year
52. item['month'] = datetime.datetime.now().month
53. item['day'] = datetime.datetime.now().day
54. item['country'] = country.strip()
55. item['dailyNew'] = dailyNew.strip()
56. item['now'] = now.strip()
57. item['total'] = total.strip()
58. item['cure'] = cure.strip()
59. item['death'] = death.strip()
60. item['cure\_rate'] = 'NA'
61. item['death\_rate'] = 'NA'
62. item['site'] = 'Baidu'
63. item['people'] = 'NA'
64. **yield** item
66. # 提交下一个网址
67. self.site += 1
68. **yield** Request(self.all\_urls[self.site])
70. **elif** self.site == 1:  # 如果是第二个网站(腾讯)
71. **print**("# Tencent:")
72. **print**("Get {} data".format(len(response.xpath('//\*[@id="foreignWraper"]/table/tbody[\*]'))))
73. **for** each **in** response.xpath('//\*[@id="foreignWraper"]/table/tbody[\*]'):
74. # 获取原始数据
75. country = each.xpath('tr/th/span/text()').extract()[0]  # 获取国家名
76. dailyNew = each.xpath('tr/td[1]/text()').extract()[0]  # 获取每日新增
77. total = each.xpath('tr/td[2]/text()').extract()[0]  # 获取累计确诊数
78. cure = each.xpath('tr/td[3]/text()').extract()[0]  # 获取治愈数
79. death = each.xpath('tr/td[4]/text()').extract()[0]  # 获取死亡数
81. # 装入item
82. item['year'] = datetime.datetime.now().year
83. item['month'] = datetime.datetime.now().month
84. item['day'] = datetime.datetime.now().day
85. item['country'] = country.strip()
86. item['dailyNew'] = dailyNew.strip()
87. item['now'] = 'NA'
88. item['total'] = total.strip()
89. item['cure'] = cure.strip()
90. item['death'] = death.strip()
91. item['cure\_rate'] = 'NA'
92. item['death\_rate'] = 'NA'
93. item['site'] = 'Tencent'
94. item['people'] = 'NA'
95. **yield** item
97. # 提交下一个网址
98. self.site += 1
99. **yield** Request(self.all\_urls[self.site])
101. **else**:
102. **print**("# AI Big Data:")
103. **print**("Get {} data".format(len(response.xpath('//\*[@id="app"]/div/section[2]/section/main/div/div[6]/div[3]/table/tbody/tr[\*]'))))
104. **for** each **in** response.xpath('//\*[@id="app"]/div/section[2]/section/main/div/div[6]/div[3]/table/tbody/tr[\*]'):
105. # 获取原始数据
106. country = each.xpath('td[1]/div/a/span/text()').extract()[0]
107. dailyNew = each.xpath('td[2]/div/span/text()').extract()[0]
108. now = each.xpath('td[4]/div/span/text()').extract()[0]
109. total = each.xpath('td[3]/div/span/text()').extract()[0]
110. cure = each.xpath('td[6]/div/span/text()').extract()[0]
111. death = each.xpath('td[9]/div/span/text()').extract()[0]
112. cure\_rate = each.xpath('td[7]/div/span/text()').extract()[0]
113. death\_rate = each.xpath('td[10]/div/span/text()').extract()[0]
114. people = each.xpath('td[12]/div/span/text()').extract()[0].replace(',', '')
116. # 装入item
117. item['year'] = datetime.datetime.now().year
118. item['month'] = datetime.datetime.now().month
119. item['day'] = datetime.datetime.now().day
120. item['country'] = country.strip()
121. item['dailyNew'] = dailyNew.strip()
122. item['now'] = now.strip()
123. item['total'] = total.strip()
124. item['cure'] = cure.strip()
125. item['death'] = death.strip()
126. item['cure\_rate'] = cure\_rate.strip()
127. item['death\_rate'] = death\_rate.strip()
128. item['site'] = 'AI'
129. item['people'] = people.strip()
130. **yield** item
131. **pass**
133. **print**('\*' \* 40)
134. **pass**

middlewares:

1. **import** scrapy
2. **import** time
4. **class** Cov19Middleware(object):
5. **def** \_\_init\_\_(self):
6. **pass**
7. **def** process\_request(self, request, spider):
8. wd = spider.driver
9. # 浏览器加载链接
10. wd.get(request.url)
12. # 数据完整性保证
13. **if** spider.site == 0:  # 如果是百度，要展开全部数据
14. **if** spider.showall == False:  # 如果还没展开全部数据，就通过click点击按钮使其展开
15. **print**("Try to show all details...")
16. all = wd.find\_element\_by\_xpath('//\*[@id="foreignTable"]/div/span')
17. all.click()
18. spider.showall = True
19. **else**:  # 否则等待两秒确保加载完毕
20. **print**('continue')
21. time.sleep(5)
23. # 打包返回response
24. html = spider.driver.page\_source
25. **return** scrapy.http.HtmlResponse(url=spider.driver.current\_url, body=html.encode('utf-8'), encoding='utf-8',
26. request=request)

piplines:

1. **import** csv
3. **class** Cov19Pipeline(object):
4. all\_path = []
5. **def** open\_spider(self, spider):
6. self.all\_path.append("cov19\_Baidu.csv")
7. self.all\_path.append("cov19\_Tencent.csv")
8. self.all\_path.append("cov19\_AI.csv")
9. # for path in self.all\_path:
10. #     # 写表头
11. #     with open(path, 'w', encoding="utf-8", newline='') as f:
12. #         csv\_write = csv.writer(f)
13. #         csv\_head = ['year', 'month', 'day', 'country', 'dailyNew', 'now', 'total', 'cure', 'death',
14. #                     'cure\_rate', 'death\_rate', 'site', 'people']
15. #         csv\_write.writerow(csv\_head)
17. **def** process\_item(self, item, spider):
18. # 确定往哪张表里添加数据
19. path = self.all\_path[spider.site]
20. # 写数据
21. with open(path, 'a', encoding="utf-8", newline='') as f:
22. csv\_write = csv.writer(f)
23. csv\_write.writerow(item.values())
25. **return** item
27. **def** close\_spider(self, spider):
28. spider.driver.quit()
29. **print**('Chrome closed!')
30. **print**('Spider closed!')

##### 各国应对措施打分代码

1. **import** pandas as pd
3. # 读取数据(直接前面的数据)
4. cure = pd.read\_csv("cure\_1.csv")
5. cure.columns = pd.Series(['country', 'cure\_per'])  # 修改列名
6. total = pd.read\_csv("d\_1.csv")
7. total.columns = pd.Series(['country', 'total\_per'])  # 修改列名
8. death = pd.read\_csv("e\_1.csv")
9. death.columns = pd.Series(['country', 'death\_per'])  # 修改列名
10. new = pd.read\_csv("c1\_1.csv")
11. new.columns = pd.Series(['country', 'new\_per'])  # 修改列名
12. people = pd.read\_csv("people.csv")
13. new.columns = pd.Series(['country', 'people'])  # 修改列名
15. key = pd.concat([cure, total, death, new, people], axis=1, join='outer', ignore\_index=True)  # 合并表
16. key.drop([2, 4, 6, 8], inplace=True, axis=1)  # 删除冗余列
17. key.columns = pd.Series(['country', 'cure\_per', 'total\_per', 'death\_per', 'new\_per', 'people'])  # 修改列名
18. key.drop(index=(key.loc[(key['country']=='钻石号邮轮')].index), inplace=True)  # 删除特殊数据
19. key.drop(index=(key.loc[(key['country']=='日本本土')].index), inplace=True)  # 删除特殊数据
20. key.drop(index=(key.loc[(key['death\_per']==0)].index), inplace=True)  # 删除死亡人数为0的行
21. key.drop(index=(key.loc[(key['people']<5000)].index), inplace=True)  # 删除人口小于5000的国家
23. key['score'] = (key['cure\_per'] \* 13 + (100 - key['total\_per'] \* 10) \* 32 + (100 - key['death\_per'] \* 10) \* 5 + (100 - key['new\_per'] \* 1) \* 50) / 100
24. key.sort\_values(by='score', ascending=False, inplace=True)  # 排序
25. **print**(key)
26. key.to\_csv('score.csv')

##### 数据展示代码

1. 15天中，全球新冠疫情的总体变化趋势
2. **import** pandas as pd
3. **import** numpy as np
4. **import** matplotlib
5. **import** matplotlib.pyplot as plt
7. # 15天中，全球新冠疫情的总体变化趋势
8. matplotlib.rcParams['font.sans-serif'] = ['SimHei']  # 设置对中文的支持
9. plt.style.use('Solarize\_Light2')
11. fig = plt.figure()
12. # fig, ax = plt.subplots()
13. ax1 = fig.add\_subplot(211)
14. ax2 = fig.add\_subplot(212)
15. csv\_path = "cov19\_Tencent.csv"
16. data = pd.read\_csv(csv\_path)
17. data[(data['dailyNew'] < 0)]['dailyNew'] = np.nan  # 将异常数据置为0
19. key = [data.groupby(['year', 'month', 'day'])['total'].sum()]  # 求得每日全球累计确诊数之和
20. key.append(data.groupby(['year', 'month', 'day'])['dailyNew'].sum())
22. keyy = pd.concat(key, axis=1)
24. **print**(keyy)
26. keyy.to\_csv('a\_1.csv')  # 储存每日全球累计确诊数之和
27. keyy = pd.read\_csv('a\_1.csv')
28. **print**(keyy)
30. x1 = np.arange(1, keyy.shape[0] + 1, 1)  # 图形1x数据
31. y1 = (keyy['total']).tolist()  # 图形1y数据
32. ax1.plot(x1, y1, marker='o', linestyle='--', alpha=0.5, color='g')  # 作图
34. x2 = np.arange(1, keyy.shape[0] + 1, 1)  # 图形2x数据
35. y2 = (keyy['dailyNew']).tolist()  # 图形2y数据
36. ax2.plot(x2, y2, marker='o', linestyle='--', alpha=0.5, color='r')  # 作图
38. xticks = []  # 构造x标签
39. **for** i **in** range(keyy.shape[0]):
40. tmplist = keyy.iloc[i].tolist()
41. xticks.append(str(tmplist[0])+'.'+str(tmplist[1])+'.'+str(tmplist[2]))
42. **pass**
43. ax1.set\_xticks(range(1, keyy.shape[0] + 1))  # 设置x轴刻度
44. ax1.set\_xticklabels(xticks)  # 设置x轴刻度标签
45. ax1.set\_xlabel("时间")  # 设置x轴上的标签
46. ax1.set\_ylabel("人数")  # 设置y轴上的标签
47. ax1.set\_title("全球变化趋势(累计)", fontsize=12, color='grey')  # 设置标题
49. ax2.set\_xticks(range(1, keyy.shape[0] + 1))  # 设置x轴刻度
50. ax2.set\_xticklabels(xticks)  # 设置x轴刻度标签
51. ax2.set\_xlabel("时间")  # 设置x轴上的标签
52. ax2.set\_ylabel("人数")  # 设置y轴上的标签
53. ax2.set\_title("全球变化趋势(新增)", fontsize=12, color='grey')  # 设置标题
55. **for** a, b **in** zip(x1, y1):
56. ax1.text(a - 0.25, b, '%d' % b, ha='center', va='center', fontsize=10)  # 数值点旁添加具体值
57. **for** a, b **in** zip(x2, y2):
58. ax2.text(a - 0.2, b, '%d' % b, ha='center', va='center', fontsize=10)  # 数值点旁添加具体值
60. plt.get\_current\_fig\_manager().window.showMaximized()  # 窗口最大化
61. plt.show()  # 显示
62. 累计确诊数排名前 20 的国家名称及其数量
63. **import** pandas as pd
64. **import** numpy as np
65. **import** matplotlib
66. **import** matplotlib.pyplot as plt
68. # 累计确诊数排名前 20 的国家名称及其数量
69. matplotlib.rcParams['font.sans-serif'] = ['SimHei']  # 设置对中文的支持
70. plt.style.use('Solarize\_Light2')
72. fig, ax = plt.subplots()
73. csv\_path = "cov19\_AI.csv"
74. data = pd.read\_csv(csv\_path)
76. front = 20  # 取前20个国家
78. key = [data.groupby(['country'])['total'].max()]  # 每个国家累计确诊数
79. keyy = pd.concat(key, axis=1)
80. keyy.sort\_values(by='total', ascending=False, inplace=True)
81. keyy.to\_csv('b\_1.csv')  # 储存每日全球累计确诊数之和
82. keyy = pd.read\_csv('b\_1.csv')
83. **print**(keyy)
85. x = np.arange(1, front + 1, 1)  # 图形x数据
86. y = [each **for** each **in** (keyy['total'] - keyy.iloc[front - 1][1] + 100000).tolist()[0:front]]  # 图形y数据
87. ax.bar(x, y, alpha=0.5, color='g')  # 作条形图
89. xticks = []  # 构造x标签
90. **for** i **in** range(front):
91. tmplist = keyy.iloc[i].tolist()
92. xticks.append(tmplist[0])
93. **pass**
94. ax.set\_xticks(range(1, front + 2))  # 设置x轴刻度
95. ax.set\_xticklabels(xticks)  # 设置x轴刻度标签
96. ax.set\_xlabel("国家")  # 设置x轴上的标签
97. ax.set\_ylabel("人数(为观察方便，已将全部数据减去{})".format(keyy.iloc[front - 1][1] + 100000))  # 设置y轴上的标签
98. ax.set\_title("累计确诊前{}国".format(front), fontsize=12, color='grey')  # 设置标题
100. **for** a, b **in** zip(x, y):
101. ax.text(a, b + keyy.iloc[0][1] // 70, '%d' % (b + keyy.iloc[front - 1][1] - 1),
102. ha='center', va='center', fontsize=10)  # 数值点旁添加具体值
104. plt.get\_current\_fig\_manager().window.showMaximized()  # 窗口最大化
105. plt.show()  # 显示
106. 15 天中，每日新增确诊数累计排名前 10 个国家的每日新增确诊数据的曲线图
107. **import** pandas as pd
108. **import** numpy as np
109. **import** matplotlib
110. **import** matplotlib.pyplot as plt
112. # 15 天中，每日新增确诊数累计排名前 10 个国家的每日新增确诊数据的曲线图
113. matplotlib.rcParams['font.sans-serif'] = ['SimHei']  # 设置对中文的支持
114. plt.style.use('Solarize\_Light2')
116. fig, ax = plt.subplots()
117. csv\_path = "cov19\_AI.csv"
118. data = pd.read\_csv(csv\_path)
120. front = 10  # 取前10个国家
122. key = [data.groupby(['country'])['dailyNew'].sum()]  # 求得每个国家累计确诊数之和
123. keyy = pd.concat(key, axis=1)
124. keyy.sort\_values(by='dailyNew', ascending=False, inplace=True)
125. keyy.to\_csv('c\_1.csv')  # 储存
126. keyy = pd.read\_csv('c\_1.csv')
127. front\_ten = []  # 提取前10国家名字
128. **for** i **in** range(front):
129. tmplist = keyy.iloc[i].tolist()
130. front\_ten.append(tmplist[0])
131. **pass**
132. **print**("15 天中，每日新增确诊数累计排名前 10 的国家")
133. **print**(front\_ten)
134. **print**()
136. # 画10条曲线
137. **for** i **in** range(len(front\_ten)):
138. tmp\_data = data[(data['country'] == front\_ten[i])]
139. x = np.arange(1, tmp\_data.shape[0] + 1, 1)  # 图形x数据
140. y = (tmp\_data['dailyNew']).tolist()  # 图形y数据
141. plt.plot(x, y)
142. **if** i == 0:
143. **for** a, b **in** zip(x, y):
144. ax.text(a, b - 7000, '%d' % (b + keyy.iloc[front - 1][1] - 1),
145. ha='center', va='center', fontsize=10)  # 数值点旁添加具体值
146. xticks = []  # 构造x标签
147. **for** i **in** range(tmp\_data.shape[0]):
148. tmplist = tmp\_data.iloc[i].tolist()
149. xticks.append(str(tmplist[0]) + '.' + str(tmplist[1]) + '.' + str(tmplist[2]))
150. **pass**
151. ax.set\_xticks(range(1, tmp\_data.shape[0] + 1))  # 设置x轴刻度
152. **print**(tmp\_data)
154. ax.legend(front\_ten, loc='upper left')  # 设置图例
155. ax.set\_xticklabels(xticks)  # 设置x轴刻度标签
156. ax.set\_xlabel("时间")  # 设置x轴上的标签
157. ax.set\_ylabel("人数")  # 设置y轴上的标签
158. ax.set\_title("每日新增确诊数据", fontsize=12, color='grey')  # 设置标题
160. plt.get\_current\_fig\_manager().window.showMaximized()  # 窗口最大化
161. plt.show()  # 显示
162. 累计确诊人数占国家总人口比例最高的 10 个国家
163. **import** pandas as pd
164. **import** numpy as np
165. **import** matplotlib
166. **import** matplotlib.pyplot as plt
168. # 累计确诊人数占国家总人口比例最高的 10 个国家
169. matplotlib.rcParams['font.sans-serif'] = ['SimHei']  # 设置对中文的支持
170. plt.style.use('Solarize\_Light2')
172. fig, ax = plt.subplots()
173. csv\_path = "cov19\_AI.csv"
175. front = 10  # 取前10个国家
177. data = pd.read\_csv(csv\_path)
178. data['people'] = data['people'].replace('-', 'NaN')  # 数据清洗：用NaN代替-
179. data['people'] = data['people'].astype('float')  # 数据转化为float
181. key = [data.groupby(['country'])['total'].max() / (data.groupby(['country'])['people'].max() \* 100)]  # 每个国家百分比
182. keyy = pd.concat(key, axis=1)  # 转化为df
183. keyy.to\_csv('d\_1.csv')  # 储存
184. keyy = pd.read\_csv('d\_1.csv')
185. keyy.columns = pd.Series(['country', 'per'])  # 修改列名
186. keyy.drop(index=(keyy.loc[(keyy['country']=='钻石号邮轮')].index), inplace=True)  # 删除特殊数据
187. keyy.sort\_values(by='per', ascending=False, inplace=True)  # 排序
188. **print**(keyy)
190. x = np.arange(1, front + 1, 1)  # 图形x数据
191. y = [each **for** each **in** keyy['per'].tolist()[0:front]]  # 图形y数据
192. ax.bar(x, y, alpha=0.5, color='g')  # 作条形图
194. xticks = []  # 构造x标签
195. **for** i **in** range(front):
196. tmplist = keyy.iloc[i].tolist()
197. xticks.append(tmplist[0])
198. **pass**
199. ax.set\_xticks(range(1, front + 2))  # 设置x轴刻度
200. ax.set\_xticklabels(xticks)  # 设置x轴刻度标签
201. ax.set\_xlabel("国家")  # 设置x轴上的标签
202. ax.set\_ylabel("百分比")  # 设置y轴上的标签
203. ax.set\_title("累计确诊人数占国家总人口比例", fontsize=12, color='grey')  # 设置标题
204. **print**(xticks)
206. **for** a, b **in** zip(x, y):
207. ax.text(a, b + 0.1, '{:.2f}%'.format(b),
208. ha='center', va='center', fontsize=10)  # 数值点旁添加具体值
210. plt.get\_current\_fig\_manager().window.showMaximized()  # 窗口最大化
211. plt.show()  # 显示
212. 死亡率（累计死亡人数/累计确诊人数）最低的 10 个国家
213. **import** pandas as pd
214. **import** numpy as np
215. **import** matplotlib
216. **import** matplotlib.pyplot as plt
217. # 死亡率
219. matplotlib.rcParams['font.sans-serif'] = ['SimHei']  # 设置对中文的支持
220. plt.style.use('Solarize\_Light2')
222. fig, ax = plt.subplots()
223. csv\_path = "cov19\_AI.csv"
225. front = 10  # 取前10个国家
227. data = pd.read\_csv(csv\_path)
228. data['death'] = data['death'].astype('float')  # 数据转化为float
229. data['total'] = data['total'].astype('float')  # 数据转化为float
231. key = [data.groupby(['country'])['death'].max() / data.groupby(['country'])['total'].max() \* 100]  # 每个国家百分比
232. keyy = pd.concat(key, axis=1)  # 转化为df
233. keyy.to\_csv('e\_1.csv')  # 储存
234. keyy = pd.read\_csv('e\_1.csv')
235. keyy.columns = pd.Series(['country', 'death\_per'])  # 修改列名
236. keyy.drop(index=(keyy.loc[(keyy['country']=='钻石号邮轮')].index), inplace=True)  # 删除特殊数据
237. keyy.drop(index=(keyy.loc[(keyy['death\_per']==0)].index), inplace=True)  # 删除死亡人数为0的行
238. keyy.sort\_values(by='death\_per', ascending=True, inplace=True)  # 排序
239. **print**(keyy)
241. x = np.arange(1, front + 1, 1)  # 图形x数据
242. y = [each **for** each **in** keyy['death\_per'].tolist()[0:front]]  # 图形y数据
243. ax.bar(x, y, alpha=0.5, color='g')  # 作条形图
245. xticks = []  # 构造x标签
246. **for** i **in** range(front):
247. tmplist = keyy.iloc[i].tolist()
248. xticks.append(tmplist[0])
249. **pass**
250. ax.set\_xticks(range(1, front + 2))  # 设置x轴刻度
251. ax.set\_xticklabels(xticks)  # 设置x轴刻度标签
252. ax.set\_xlabel("国家")  # 设置x轴上的标签
253. ax.set\_ylabel("百分比")  # 设置y轴上的标签
254. ax.set\_title("死亡率最低的 10 个国家", fontsize=12, color='grey')  # 设置标题
255. **print**(xticks)
257. **for** a, b **in** zip(x, y):
258. ax.text(a, b + 0.01, '{:.2f}%'.format(b),
259. ha='center', va='center', fontsize=10)  # 数值点旁添加具体值
261. plt.get\_current\_fig\_manager().window.showMaximized()  # 窗口最大化
262. plt.show()  # 显示
263. 用饼图展示各个国家的累计确诊人数的比例
264. **import** pandas as pd
265. **import** numpy as np
266. **import** matplotlib
267. **import** matplotlib.pyplot as plt
268. **from** matplotlib **import** font\_manager as fm
270. # 用饼图展示各个国家的累计确诊人数的比例（你爬取的所有国家，数据较小的国家可以合并处理）
271. matplotlib.rcParams['font.sans-serif'] = ['SimHei']  # 设置对中文的支持
273. fig, ax = plt.subplots()
274. csv\_path = "b\_1.csv"
275. keyy = pd.read\_csv(csv\_path)  # 直接利用第b问的数据
277. border = 600000  # 合并小于border国家 为 "其他"
278. borderBySort = keyy.iloc[9][1]  # 或者合并10名以后的国家 为 "其他"
280. keyy.loc[keyy.shape[0]] = ["其他", 0]  # 添加 "其他" 列，初始化为0
282. drop\_lines = []  # 记录要删除的行
284. **for** i **in** range(keyy['country'].count() - 1):
285. **if** keyy.iloc[i][1] < borderBySort:
286. keyy.iloc[keyy.loc[(keyy['country'] == '其他')].index, 1] += keyy.iloc[i][1]  # 修改 "其他" 行
287. drop\_lines.append(i)  # 记录要删除的行
288. **pass**
289. **pass**
290. keyy.drop(index=drop\_lines, inplace=True)  # 删除数据
291. keyy = keyy.reset\_index(drop=True)  # 重设索引
293. **print**(keyy)
295. # 作图
296. size = []  # 各国家的人数
297. label\_list = []  # 各部分标签
298. **for** i **in** range(keyy.shape[0]):
299. tmplist = keyy.iloc[i].tolist()
300. size.append(tmplist[1])
301. label\_list.append(tmplist[0])
302. **pass**
304. explode = np.zeros(keyy.shape[0])  # 各部分的突出显示比例(法1)
305. explode[0] = 0.1
306. # explode = np.linspace(0.2, 0.05, keyy.shape[0])  # 各部分的突出显示比例(法2)
307. patches, texts, autotexts = plt.pie(size, explode=explode, labels=label\_list,
308. labeldistance=1.1, autopct="%1.1f%%", shadow=False, startangle=0, pctdistance=0.9)
309. # 调整字体
310. proptease = fm.FontProperties()
311. proptease.set\_size('x-large')
312. plt.setp(texts, fontproperties=proptease)
313. plt.setp(autotexts, fontproperties=proptease)
314. ax.set\_title('各个国家的累计确诊人数的比例(仅展示前10个国家，剩下的归于"其他")', fontsize=16)  # 设置标题
315. plt.get\_current\_fig\_manager().window.showMaximized()  # 窗口最大化
316. plt.show()
317. 展示全球各个国家累计确诊人数的箱型图，要有平均值
318. **import** pandas as pd
319. **import** matplotlib
320. **import** matplotlib.pyplot as plt
321. **from** util **import** util
323. # 展示全球各个国家累计确诊人数的箱型图，要有平均值
324. matplotlib.rcParams['font.sans-serif'] = ['SimHei']  # 设置对中文的支持
326. fig, ax = plt.subplots()
327. csv\_path = "b\_1.csv"
328. keyy = pd.read\_csv(csv\_path)  # 直接利用第b问的数据
329. avee = keyy['total'].mean()
330. maxx = keyy['total'].max()
331. medd = keyy['total'].median()
332. upperQuartile = keyy['total'].quantile(0.75)
333. lowerQuartile = keyy['total'].quantile(0.25)
335. x = keyy['total'].tolist()  # 图形数据
336. f = plt.boxplot(x, showmeans=True, vert=False, meanline=True, widths=0.5,  showfliers=False)
338. ax.set\_xlabel("人数")  # 设置x轴上的标签
339. ax.set\_yticklabels(['全球'])  # 设置y轴刻度标签
340. ax.set\_title("全球各个国家累计确诊人数的箱型图", fontsize=12, color='grey')  # 设置标题
342. # 设置关键点注释
343. ax.annotate("平均值={:.2f}人".format(avee), xy=(avee, 1.05),
344. xytext=(avee - 70000, 1.1), arrowprops=dict(arrowstyle="->"))
345. ax.annotate("最大值={:.0f}人".format(maxx), xy=(maxx, 1),
346. xytext=(maxx - 50000, 1.1), arrowprops=dict(arrowstyle="->"))
347. ax.annotate("中位数={:.0f}人".format(medd), xy=(medd, 1.05),
348. xytext=(medd + 50000, 1.1), arrowprops=dict(arrowstyle="->"))
349. ax.annotate("上四分位点={:.0f}人".format(upperQuartile), xy=(upperQuartile, 0.95),
350. xytext=(upperQuartile + 50000, 0.9), arrowprops=dict(arrowstyle="->"))
351. ax.annotate("下四分位点={:.0f}人".format(lowerQuartile), xy=(lowerQuartile, 0.95),
352. xytext=(lowerQuartile + 50000, 0.9), arrowprops=dict(arrowstyle="->"))
354. plt.get\_current\_fig\_manager().window.showMaximized()  # 窗口最大化
355. plt.show()
356. 其他展示
357. **import** pandas as pd
358. **import** numpy as np
359. **import** matplotlib
360. **import** matplotlib.pyplot as plt
362. # 中国
363. matplotlib.rcParams['font.sans-serif'] = ['SimHei']  # 设置对中文的支持
364. plt.style.use('Solarize\_Light2')
366. fig, ax = plt.subplots()
367. csv\_path = "cov19\_AI.csv"
368. data = pd.read\_csv(csv\_path)
370. keyyy = data[(data['country'] == '中国')][['year', 'month', 'day', 'dailyNew']]
372. x = np.arange(1, keyyy.shape[0] + 1, 1)  # 图形2x数据
373. y = (keyyy['dailyNew']).tolist()  # 图形2y数据
374. ax.plot(x, y, marker='o', linestyle='--', alpha=0.5, color='r')  # 作图
376. xticks = []  # 构造x标签
377. **for** i **in** range(keyyy.shape[0]):
378. tmplist = keyyy.iloc[i].tolist()
379. xticks.append(str(tmplist[0])+'.'+str(tmplist[1])+'.'+str(tmplist[2]))
380. **pass**
382. ax.set\_xticks(range(1, keyyy.shape[0] + 1))  # 设置x轴刻度
383. ax.set\_xticklabels(xticks)  # 设置x轴刻度标签
384. ax.set\_xlabel("时间")  # 设置x轴上的标签
385. ax.set\_ylabel("人数")  # 设置y轴上的标签
386. ax.set\_title("中国每日新增", fontsize=12, color='grey')  # 设置标题
387. plt.show()
388. **import** pandas as pd
389. **import** numpy as np
390. **import** matplotlib
391. **import** matplotlib.pyplot as plt
393. # 治愈
394. matplotlib.rcParams['font.sans-serif'] = ['SimHei']  # 设置对中文的支持
395. plt.style.use('Solarize\_Light2')
397. fig, ax = plt.subplots()
398. csv\_path = "cov19\_AI.csv"
400. front = 10  # 取前10个国家
402. data = pd.read\_csv(csv\_path)
403. data['death'] = data['cure'].astype('float')  # 数据转化为float
404. data['total'] = data['total'].astype('float')  # 数据转化为float
406. key = [data.groupby(['country'])['cure'].max() / data.groupby(['country'])['total'].max() \* 100]  # 每个国家百分比
407. keyy = pd.concat(key, axis=1)  # 转化为df
408. keyy.to\_csv('cure\_1.csv')  # 储存
409. keyy = pd.read\_csv('cure\_1.csv')
410. keyy.columns = pd.Series(['country', 'cure\_per'])  # 修改列名
411. keyy.drop(index=(keyy.loc[(keyy['country']=='钻石号邮轮')].index), inplace=True)  # 删除特殊数据
412. keyy.drop(index=(keyy.loc[(keyy['cure\_per']==0)].index), inplace=True)  # 删除死亡人数为0的行
413. keyy.sort\_values(by='cure\_per', ascending=False, inplace=True)  # 排序
414. **print**(keyy)
416. x = np.arange(1, front + 1, 1)  # 图形x数据
417. y = [each **for** each **in** keyy['cure\_per'].tolist()[0:front]]  # 图形y数据
418. ax.bar(x, y, alpha=0.5, color='g')  # 作条形图
420. xticks = []  # 构造x标签
421. **for** i **in** range(front):
422. tmplist = keyy.iloc[i].tolist()
423. xticks.append(tmplist[0])
424. **pass**
425. ax.set\_xticks(range(1, front + 2))  # 设置x轴刻度
426. ax.set\_xticklabels(xticks)  # 设置x轴刻度标签
427. ax.set\_xlabel("国家")  # 设置x轴上的标签
428. ax.set\_ylabel("百分比")  # 设置y轴上的标签
429. ax.set\_title("治愈率最高的 10 个国家", fontsize=12, color='grey')  # 设置标题
431. **for** a, b **in** zip(x, y):
432. ax.text(a, b + 0.9, '{:.2f}%'.format(b),
433. ha='center', va='center', fontsize=10)  # 数值点旁添加具体值
435. plt.get\_current\_fig\_manager().window.showMaximized()  # 窗口最大化
436. plt.show()  # 显示

##### 数据预测代码

1. **import** pandas as pd
2. **import** matplotlib
3. **import** matplotlib.pyplot as plt
4. **import** numpy as np
5. **from** scipy.optimize **import** nnls
6. **import** math
7. **from** sklearn.linear\_model **import** LinearRegression

10. # 图形界面预设以及生成画布与坐标轴
11. matplotlib.rcParams['font.sans-serif'] = ['SimHei']  # 设置对中文的支持
12. fig, ax = plt.subplots()
14. # 读取数据
15. csv\_path = "a\_1.csv"
16. keyy = pd.read\_csv(csv\_path)  # 直接利用第a问的数据
18. # 进一步设定初始数据
19. preDays = 5  # 预测的天数
20. real = keyy['total'].values.tolist()  # 取得真实数据
21. history = keyy['total'].values.tolist()  # 取得历史数据
22. **for** i **in** range(preDays):
23. history.pop()  # 删除后几天的数据
25. # ---------------------------------------------------------------------------- #
26. # GM(1,1)预测
28. pre1 = []  # 存放预测的数据
29. n = len(history)  # 用于训练的数据量
30. pre1.append(history[-1])  # 便于画图，添加一项真实数据
32. X0 = np.array(history)
33. X1 = np.array([sum(history[0:i+1]) **for** i **in** range(n)])  # 对数据作一次累加
35. # 构造数据矩阵B及数据向量Y
36. B = np.zeros([n-1, 2])
37. Y = np.zeros([n-1, 1])
38. **for** i **in** range(0, n-1):
39. B[i, 0] = -0.5\*(X1[i] + X1[i+1])
40. B[i, 1] = 1
41. Y[i, 0] = X0[i+1]
43. # 计算GM(1,1)微分方程的参数a和b
44. U = np.linalg.inv(B.T.dot(B)).dot(B.T).dot(Y)
45. a = U[0][0]
46. b = U[1][0]
48. # 预测
49. f = np.zeros(preDays)
50. **print**('GM(1,1)预测:')
51. **print**('时间          预测             实际         误差')
52. **for** i **in** range(0, preDays):
53. tmplist = keyy.iloc[keyy.shape[0] - preDays + i].tolist()
54. date = str(tmplist[0]) + '.' + str(tmplist[1]) + '.' + str(tmplist[2]) + '\t'
55. f[i] = (X0[0] - b / a) \* (1 - math.exp(a)) \* math.exp(-a \* (i + n))  # 此次预测的值
56. pre1.append(int(f[i]))
57. **print**(date, '{:.2f}'.format(f[i]), '\t', real[keyy.shape[0] - preDays + i], '\t',
58. '{:.2f}%'.format((f[i] - real[keyy.shape[0] - preDays + i]) / real[keyy.shape[0] - preDays + i] \* 100))
60. # ---------------------------------------------------------------------------- #
61. # k阶差分预测
62. pre2 = []
63. pre2.append(history[-1])
65. k = 8  # k阶差分预测
67. n = len(history)
68. x = [history[k - i: n-i] **for** i **in** range(1, k+1)]
69. x.append([1 **for** j **in** range(n-k)])
70. x = np.array(x).T
71. y = np.array(history[k: n]).T
73. paras = nnls(x, y)[0]
75. sq = history.copy()
76. j = n
77. **print**('k阶差分预测:')
78. **print**('时间          预测             实际         误差')
79. **for** i **in** range(preDays):
80. tmplist = keyy.iloc[keyy.shape[0] - preDays + i].tolist()
81. date = str(tmplist[0]) + '.' + str(tmplist[1]) + '.' + str(tmplist[2]) + '\t'
82. temp = 0
83. **for** j **in** range(k):
84. temp += paras[j] \* sq[len(sq)-j-1]
85. temp += paras[k]
86. pre2.append(int(temp))
87. sq.append(temp)
88. # print(temp)
89. **print**(date, '{:.2f}'.format(temp), '\t', real[keyy.shape[0] - preDays + i], '\t',
90. '{:.2f}%'.format((temp - real[keyy.shape[0] - preDays + i]) / real[keyy.shape[0] - preDays + i] \* 100))

93. # ---------------------------------------------------------------------------- #
94. # 一元线性回归
95. pre3 = []
96. pre3.append(history[-1])
98. Model = LinearRegression()
99. Model.fit(np.arange(1, keyy.shape[0] + 1 - preDays, 1).reshape(-1, 1), history)
101. **print**('一元线性回归预测:')
102. **print**('时间          预测             实际         误差')
103. **for** i **in** range(0, preDays):
104. tmplist = keyy.iloc[keyy.shape[0] - preDays + i].tolist()
105. date = str(tmplist[0]) + '.' + str(tmplist[1]) + '.' + str(tmplist[2]) + '\t'
106. tmppre = Model.predict([[keyy.shape[0] - preDays + i + 1]]).tolist()[0]
107. pre3.append(int(tmppre))
108. **print**(date, '{:.2f}'.format(tmppre), '\t', real[keyy.shape[0] - preDays + i], '\t',
109. '{:.2f}%'.format((tmppre - real[keyy.shape[0] - preDays + i]) / real[keyy.shape[0] - preDays + i] \* 100))

112. # ---------------------------------------------------------------------------- #
113. # 作图
114. x1 = np.arange(1, keyy.shape[0] + 1, 1)
115. y1 = real  # 实际数据
116. x2 = np.arange(keyy.shape[0] - preDays, keyy.shape[0] + 1, 1)
117. y2 = pre1  # 预测数据
118. x3 = np.arange(keyy.shape[0] - preDays, keyy.shape[0] + 1, 1)
119. y3 = pre2  # 预测数据
120. x4 = np.arange(keyy.shape[0] - preDays, keyy.shape[0] + 1, 1)
121. y4 = pre3  # 预测数据
123. ax.plot(x1, y1, marker='o', linestyle='-', alpha=0.5, color='r')  # 作图
124. ax.plot(x2, y2, marker='o', linestyle='--', alpha=0.5, color='b')  # 作图
125. ax.plot(x3, y3, marker='o', linestyle='--', alpha=0.5, color='g')  # 作图
126. ax.plot(x4, y4, marker='o', linestyle='--', color='y')  # 作图
128. xticks = []  # 构造x标签
129. **for** i **in** range(keyy.shape[0]):
130. tmplist = keyy.iloc[i].tolist()
131. xticks.append(str(tmplist[0])+'.'+str(tmplist[1])+'.'+str(tmplist[2]))
132. **pass**
133. ax.set\_xticks(range(1, keyy.shape[0] + 1))  # 设置x轴刻度
134. ax.set\_xticklabels(xticks)  # 设置x轴刻度标签
135. ax.set\_xlabel("时间")  # 设置x轴上的标签
136. ax.set\_ylabel("人数")  # 设置y轴上的标签
137. ax.set\_title("全球变化趋势(预测{}天)".format(preDays), fontsize=12, color='grey')  # 设置标题
138. first = True
139. **for** a, b **in** zip(x1, y1):
140. ax.text(a - 0.25, b + 300000, '%d' % b, ha='center', va='center', fontsize=10, color='r')  # 数值点旁添加具体值
141. **for** a, b **in** zip(x2, y2):
142. **if** first:
143. first = False
144. **else**:
145. ax.text(a + 0.25, b - 300000, '%d' % b, ha='center', va='center', fontsize=10, color='b')  # 数值点旁添加具体值
146. first = True
147. **for** a, b **in** zip(x3, y3):
148. **if** first:
149. first = False
150. **else**:
151. ax.text(a + 0.25, b - 200000, '%d' % b, ha='center', va='center', fontsize=10, color='g')  # 数值点旁添加具体值
152. first = True
153. **for** a, b **in** zip(x4, y4):
154. **if** first:
155. first = False
156. **else**:
157. ax.text(a + 0.25, b - 400000, '%d' % b, ha='center', va='center', fontsize=10, color='y')  # 数值点旁添加具体值
158. ax.legend(['实际', 'GM(1,1)预测', '差分预测', '一元线性回归'], loc='upper left')  # 设置图例
159. plt.grid(True, linestyle='--', alpha=0.3, color='grey')  # 设置网格线
160. plt.get\_current\_fig\_manager().window.showMaximized()  # 窗口最大化
161. plt.show()