Python 期末大作业：分析新冠疫情数据

目录

一. 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。

二. 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/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：

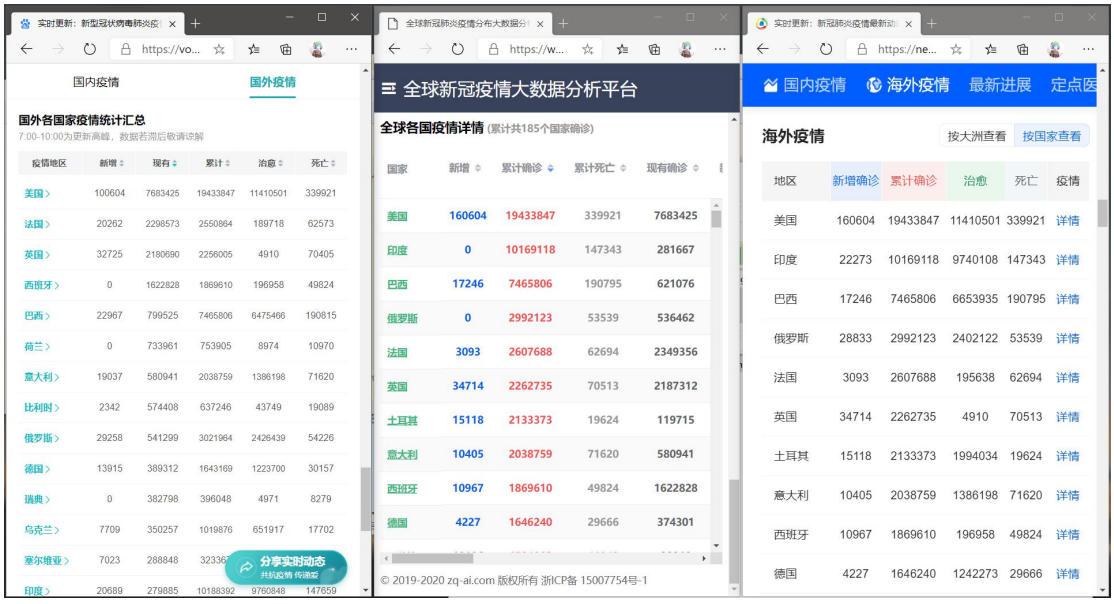

图 1 数据来源：左图是百度，中图是大数据分析平台，右图是腾讯

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。

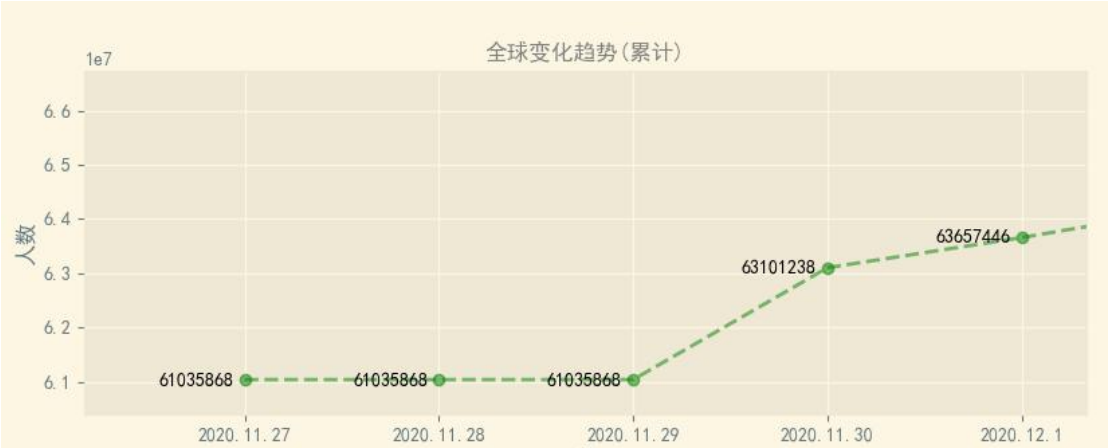

图 2 百度数据出现更新停滞的现象

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。

三. 数据分析和展示

1. 15 天中，全球新冠疫情的总体变化趋势

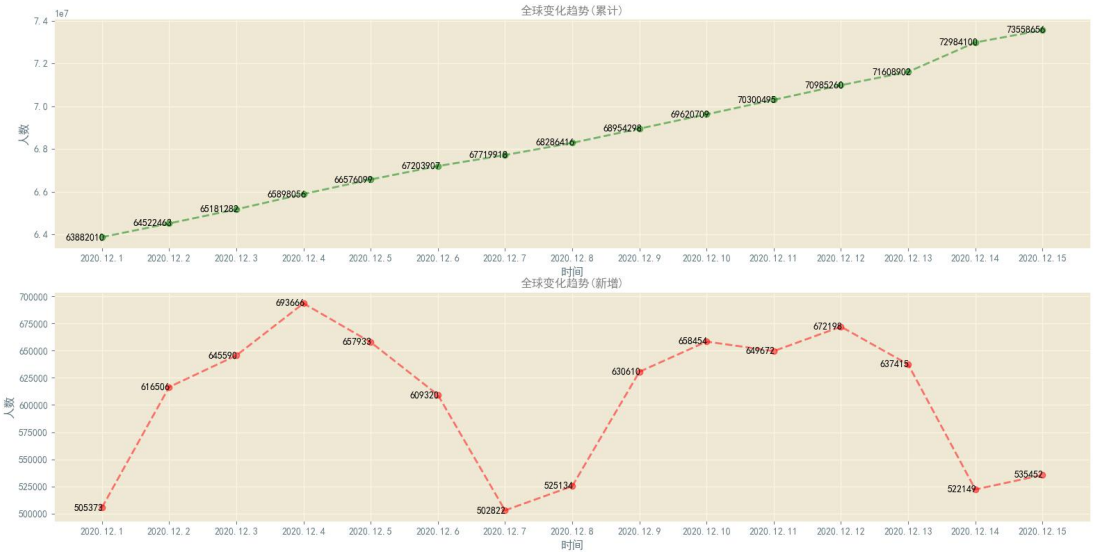

图 3 15 天中，全球新冠疫情的总体变化趋势

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。

2. 累计确诊数排名前 20 的国家名称及其数量

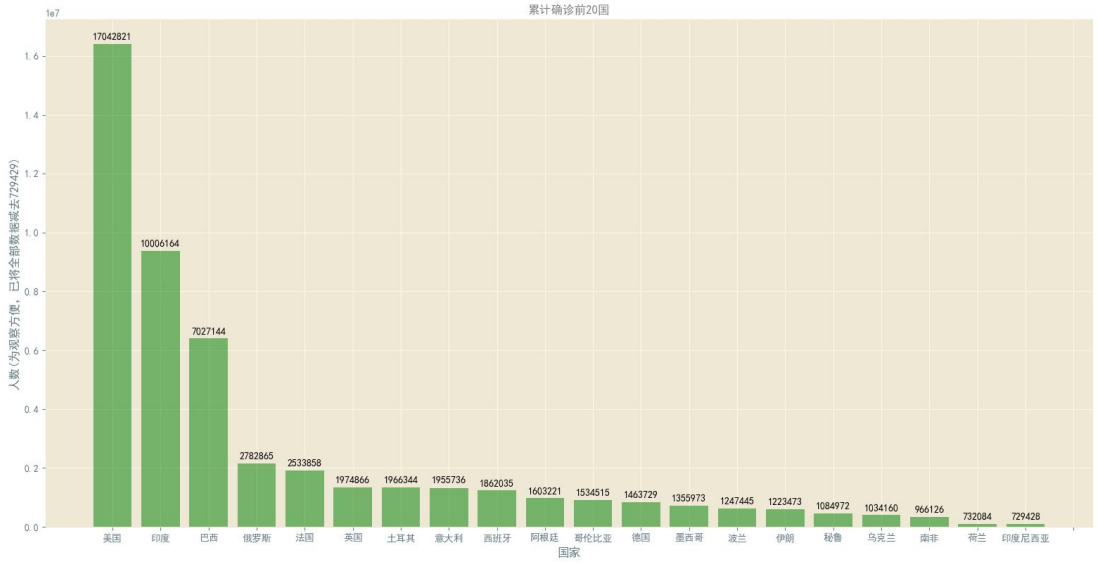

图 4 累计确诊数排名前 20 的国家名称及其数量

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。

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

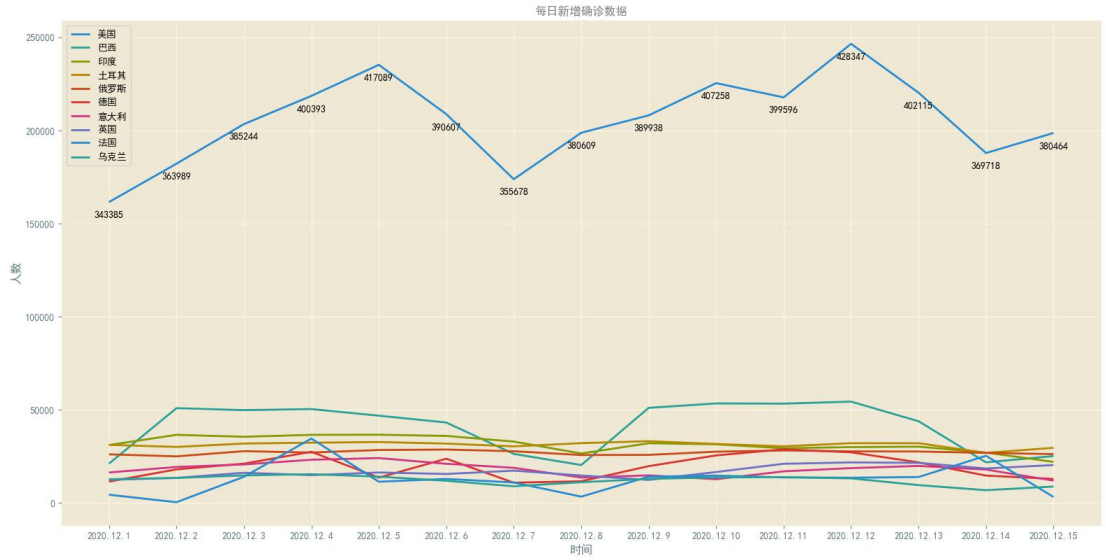

图 5 15 天中，每日新增确诊数累计排名前 10 个国家的每日新增确诊数据的曲线图

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。

4. 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：

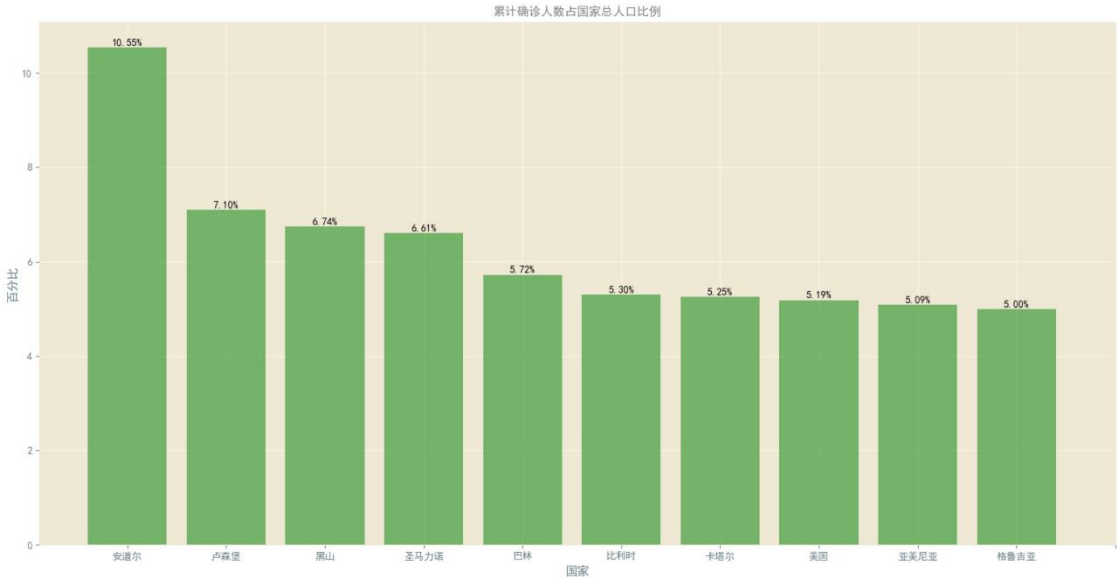

图 6 累计确诊人数占国家总人口比例最高的 10 个国家

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。

5. 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。

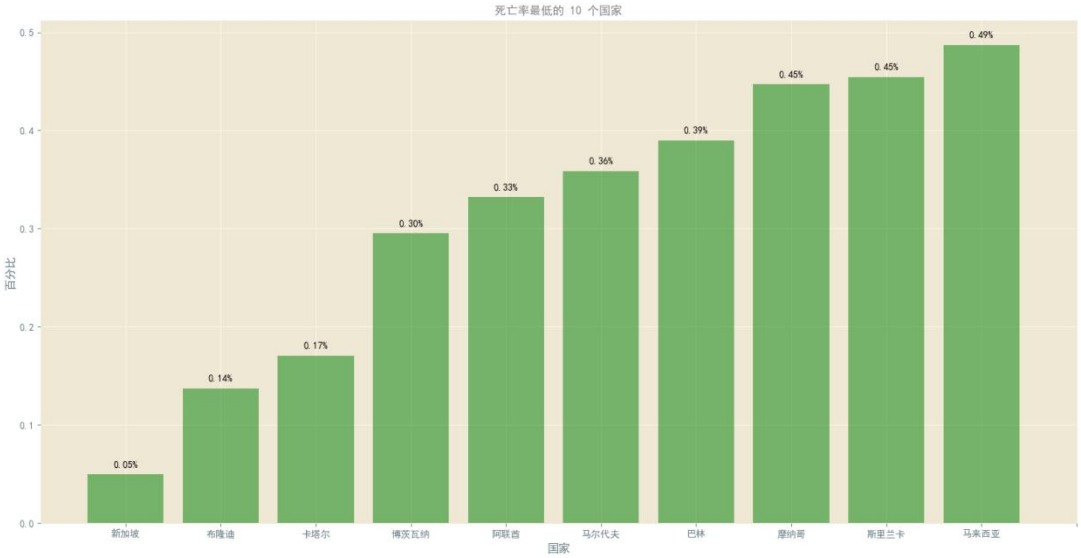

图 7 死亡率（累计死亡人数/累计确诊人数）最低的 10 个国家

6. 用饼图展示各个国家的累计确诊人数的比例

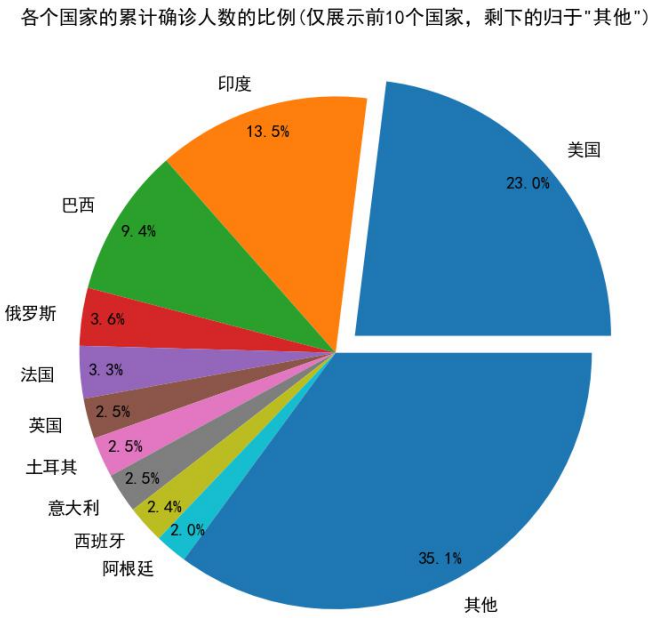

图 8 各个国家的累计确诊人数比例的饼图

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。

7. 展示全球各个国家累计确诊人数的箱型图，要有平均值

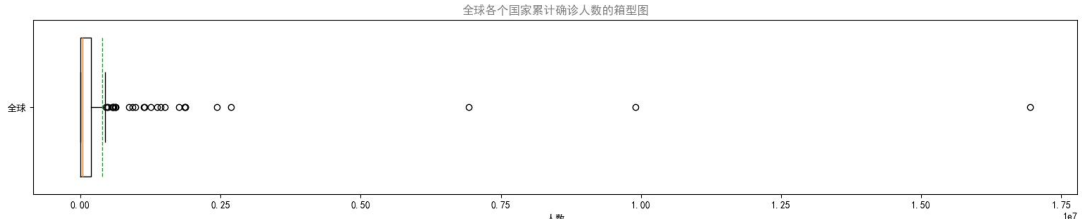

图 9 含异常值的箱型图

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:

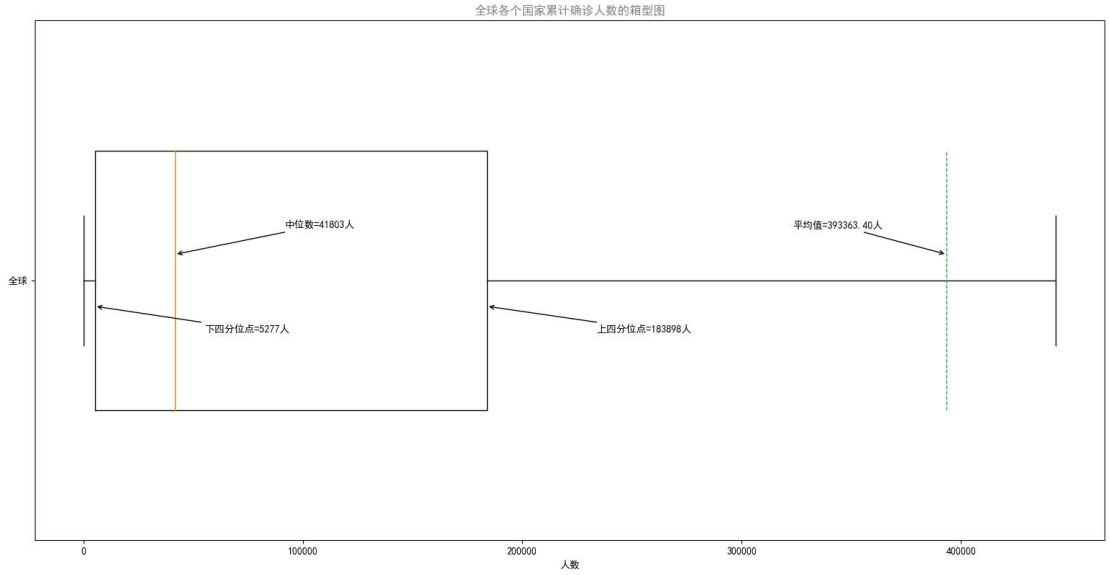

图 10 不含异常值的箱型图

8. 其他展示

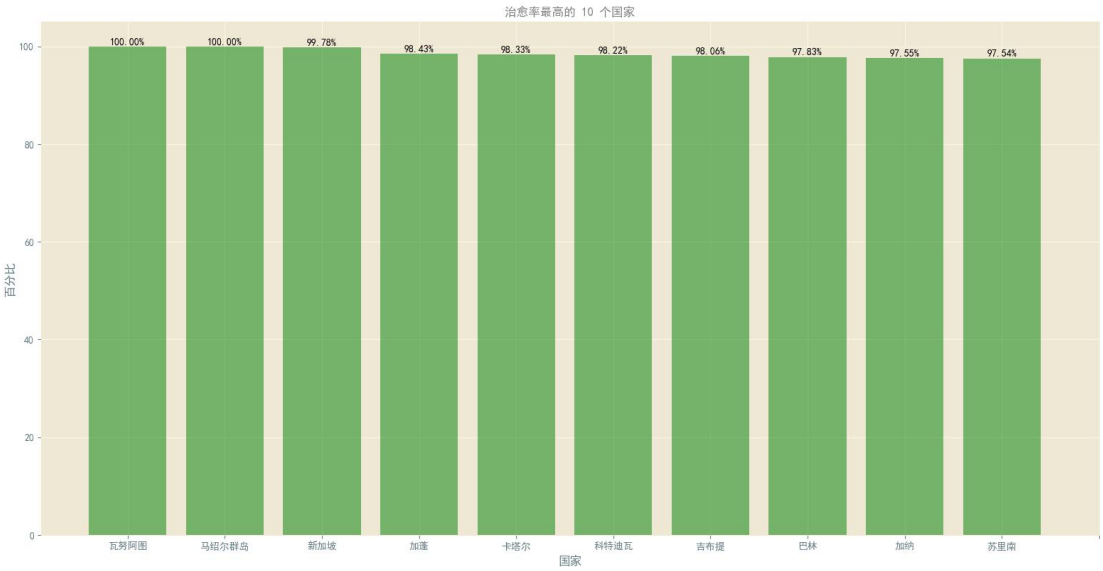

图 11 治愈率最高的 10 个国家

图 12 中国每日新增线图

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

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：

$$score = 治愈率 \times 15\% + (1 - 累计确诊比例) \times 30\% + (1 - 死亡率) \times 5\% + (1 - 新增确诊比例) \times 50\%$$

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 5000 的国家。

最终得出分数最高的国家如下：

country	cure_per	total_per	death_per	new_per	people	score
泰国	93.005181	0.006138	1.413095	0.000024	6918	98.364474
尼日利亚	90.378063	0.037461	1.631368	0.000203	19587	97.813489
越南	88.873039	0.001453	2.496434	0.000004	9649	97.300627
菲律宾	92.702710	0.424222	1.950252	0.001265	10651	96.718083
巴基斯坦	87.159952	0.220729	2.009042	0.001501	20081	96.619189
中国	93.225356	0.006805	4.999633	0.000009	140005	96.597701
日本	81.776861	0.145658	1.387869	0.001830	12718	96.470036
埃塞俄比亚	81.220894	0.109911	1.542962	0.000482	10667	96.435279
印度	95.118908	0.731595	1.450703	0.002336	135405	96.297834
肯尼亚	79.791429	0.180677	1.730487	0.001143	5095	95.928904

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。

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

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%：

GM(1,1)预测：

时间	预测	实际	误差
2020.12.11	70294657.27	70300495	-0.01%
2020.12.12	70954358.66	70985260	-0.04%
2020.12.13	71620251.23	71608902	0.02%
2020.12.14	72292393.06	72984100	-0.95%
2020.12.15	72970842.80	73558656	-0.80%

k阶差分预测：

时间	预测	实际	误差
2020.12.11	70288576.56	70300495	-0.02%
2020.12.12	70963576.82	70985260	-0.03%
2020.12.13	71645298.58	71608902	0.05%
2020.12.14	72333515.37	72984100	-0.89%
2020.12.15	73028332.67	73558656	-0.72%

一元线性回归预测：

时间	预测	实际	误差
2020.12.11	70260855.80	70300495	-0.06%
2020.12.12	70892917.62	70985260	-0.13%
2020.12.13	71524979.44	71608902	-0.12%
2020.12.14	72157041.25	72984100	-1.13%
2020.12.15	72789103.07	73558656	-1.05%

图 14 三种预测比较

同时作图观察：

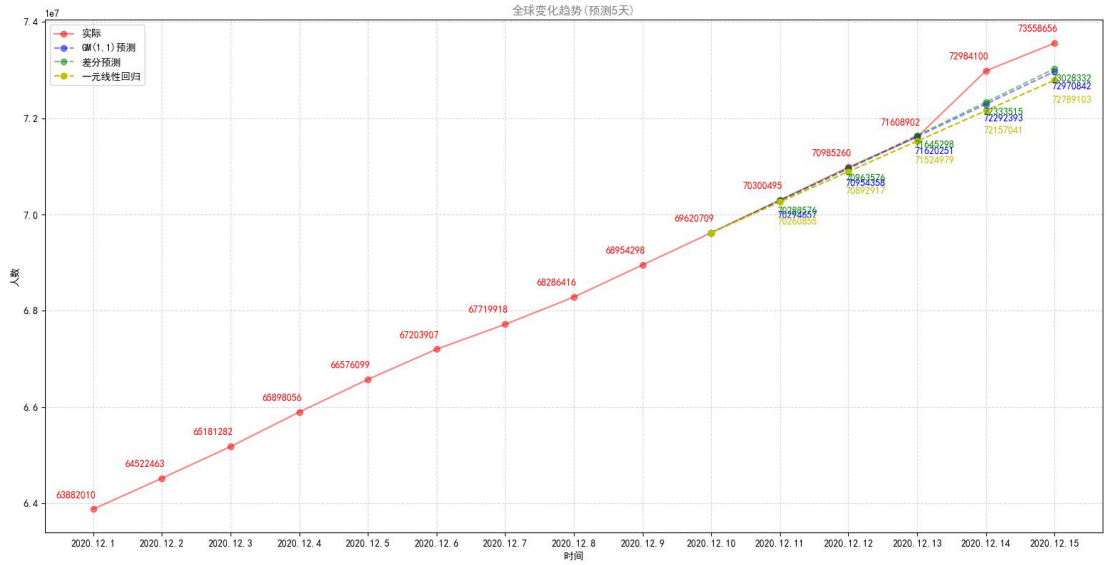

图 15 三种预测结果与实际的对比（全局趋势）

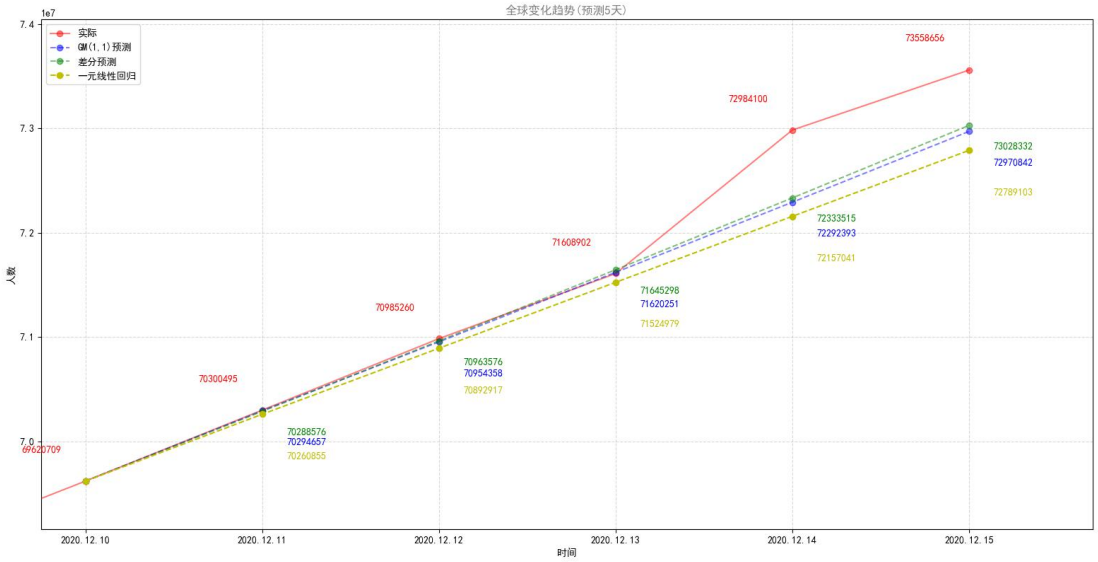

图 16 三种预测结果与实际的对比（放大预测部分）

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：

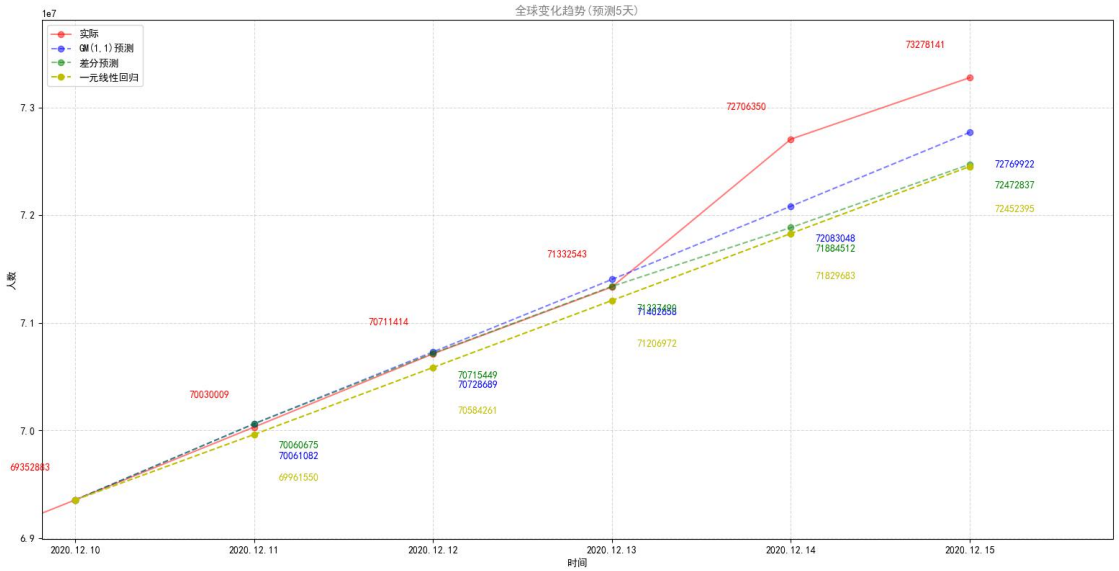

图 17 三种预测结果与实际的对比（腾讯数据）

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。

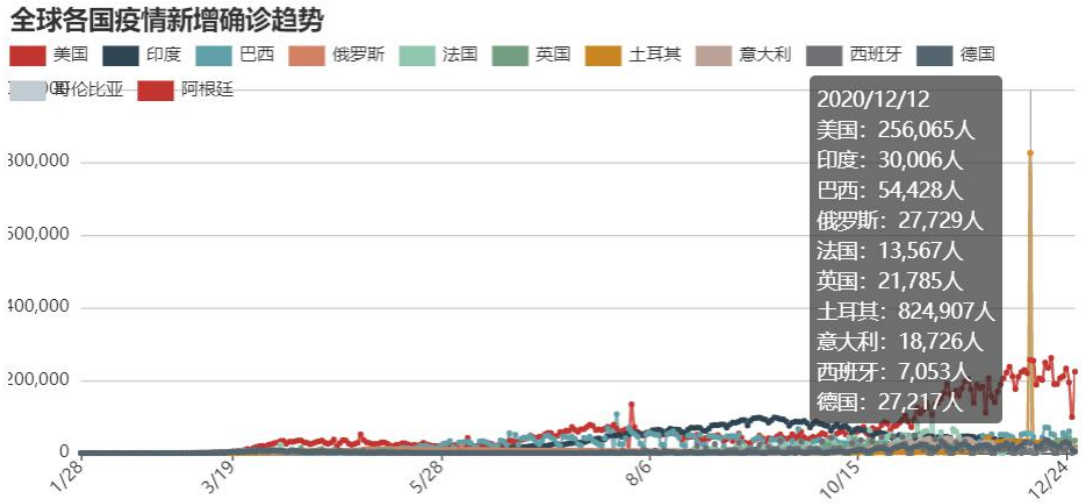

图 18 数据网站 https://www.zq-ai.com/#/fe/xgfybigdata 的曲线图

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。

六. 核心代码

1. 爬虫程序（使用 selenium）

spider：

```
import scrapy
from selenium import webdriver
from selenium.webdriver.chrome.options import Options
import datetime
from cov19.items import Cov19Item
from scrapy import Request

class CovspiderSpider(scrapy.Spider):
    name = 'covSpider'
    start_urls = ['https://voice.baidu.com/act/newpneumonia/newpneumonia']
    all_urls = ['https://voice.baidu.com/act/newpneumonia/newpneumonia',
                'https://news.qq.com/zt2020/page/feiyan.htm#/global?nojump=1',
                'https://www.zq-ai.com/#/fe/xgfybigdata']
    showall = False
    site = 0  # 指示当前正在爬取哪个网站

    def __init__(self):
        # 设定浏览器模式
        chrome_options = Options()
        chrome_options.add_argument('--headless')       # 使用无头谷歌浏览器模式
        chrome_options.add_argument('--disable-gpu')    # 禁用 GPU 加速
        chrome_options.add_argument('--no-sandbox')
        # 指定谷歌浏览器驱动路径
        self.driver = webdriver.Chrome(chrome_options=chrome_options,
                                       executable_path='C:/Users/87290/Desktop/PROGRAM/Python/chromedriver.exe')
        # 设定隐式等待时间 10s
        self.driver.implicitly_wait(10)
        # 输出提示信息
        print('Webdriver start init success!')

    def parse(self, response):
        item = Cov19Item()
        print('*' * 40)
        print("开爬 开爬 fo~↑ fo~↑")
        if self.site == 0:  # 如果是第一个网站(百度)
            print("# Baidu:")
            print("Get {} data".format(len(response.xpath('//*[@id="foreignTable"]/table/tbody/tr/td/table/tbody/tr[*]'))))
```

```
            for each in response.xpath('//*[@id="foreignTable"]/table/tbody/tr/td/table/tbody/tr[*]
'):
                # 获取原始数据
                country = each.xpath('td[1]/a/div[1]/text()').extract()
                if len(country) == 0:
                    country = each.xpath('td[1]/div/text()').extract()
                country = country[0]  # 获取国家名
                dailyNew = each.xpath('td[2]/text()').extract()[0]  # 获取每日新增
                now = each.xpath('td[3]/text()').extract()[0]  # 获取现有病例数
                total = each.xpath('td[4]/text()').extract()[0]  # 获取累计确诊数
                cure = each.xpath('td[5]/text()').extract()[0]  # 获取治愈数
                death = each.xpath('td[6]/text()').extract()[0]  # 获取死亡数

                # 装入 item
                item['year'] = datetime.datetime.now().year
                item['month'] = datetime.datetime.now().month
                item['day'] = datetime.datetime.now().day
                item['country'] = country.strip()
                item['dailyNew'] = dailyNew.strip()
                item['now'] = now.strip()
                item['total'] = total.strip()
                item['cure'] = cure.strip()
                item['death'] = death.strip()
                item['cure_rate'] = 'NA'
                item['death_rate'] = 'NA'
                item['site'] = 'Baidu'
                item['people'] = 'NA'
                yield item

            # 提交下一个网址
            self.site += 1
            yield Request(self.all_urls[self.site])

        elif self.site == 1:  # 如果是第二个网站(腾讯)
            print("# Tencent:")
              print("Get {} data".format(len(response.xpath('//*[@id="foreignWraper"]/table/tbody[*]
'))))
            for each in response.xpath('//*[@id="foreignWraper"]/table/tbody[*]'):
                # 获取原始数据
                country = each.xpath('tr/th/span/text()').extract()[0]  # 获取国家名
                dailyNew = each.xpath('tr/td[1]/text()').extract()[0]  # 获取每日新增
                total = each.xpath('tr/td[2]/text()').extract()[0]  # 获取累计确诊数
                cure = each.xpath('tr/td[3]/text()').extract()[0]  # 获取治愈数
                death = each.xpath('tr/td[4]/text()').extract()[0]  # 获取死亡数
```

```
80.
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
96.
97.             # 提交下一个网址
98.             self.site += 1
99.             yield Request(self.all_urls[self.site])
100.
101.         else:
102.             print("# AI Big Data:")
103.               print("Get {} data".format(len(response.xpath('//*[@id="app"]/div/section[2]/section/m
    ain/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(',', '')
115.
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()
```

```
                item['now'] = now.strip()
                item['total'] = total.strip()
                item['cure'] = cure.strip()
                item['death'] = death.strip()
                item['cure_rate'] = cure_rate.strip()
                item['death_rate'] = death_rate.strip()
                item['site'] = 'AI'
                item['people'] = people.strip()
                yield item
            pass

        print('*' * 40)
        pass
```

middlewares:

```
import scrapy
import time

class Cov19Middleware(object):
    def __init__(self):
        pass
    def process_request(self, request, spider):
        wd = spider.driver
        # 浏览器加载链接
        wd.get(request.url)

        # 数据完整性保证
        if spider.site == 0:  # 如果是百度，要展开全部数据
            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
                print("Try to show all details...")
                all = wd.find_element_by_xpath('//*[@id="foreignTable"]/div/span')
                all.click()
                spider.showall = True
        else:  # 否则等待两秒确保加载完毕
            print('continue')
            time.sleep(5)

        # 打包返回 response
        html = spider.driver.page_source
         return scrapy.http.HtmlResponse(url=spider.driver.current_url, body=html.encode('utf-8'),
encoding='utf-8',
                                        request=request)
```

piplines:

```
import csv

```

```
class Cov19Pipeline(object):
    all_path = []
    def open_spider(self, spider):
        self.all_path.append("cov19_Baidu.csv")
        self.all_path.append("cov19_Tencent.csv")
        self.all_path.append("cov19_AI.csv")
        # for path in self.all_path:
        #     # 写表头
        #     with open(path, 'w', encoding="utf-8", newline='') as f:
        #         csv_write = csv.writer(f)
         #          csv_head = ['year', 'month', 'day', 'country', 'dailyNew', 'now', 'total', 'cure', 'death',
        #                      'cure_rate', 'death_rate', 'site', 'people']
        #         csv_write.writerow(csv_head)

    def process_item(self, item, spider):
        # 确定往哪张表里添加数据
        path = self.all_path[spider.site]
        # 写数据
        with open(path, 'a', encoding="utf-8", newline='') as f:
            csv_write = csv.writer(f)
            csv_write.writerow(item.values())

        return item

    def close_spider(self, spider):
        spider.driver.quit()
        print('Chrome closed!')
        print('Spider closed!')
```

2. 各国应对措施打分代码

```
import pandas as pd

# 读取数据(直接前面的数据)
cure = pd.read_csv("cure_1.csv")
cure.columns = pd.Series(['country', 'cure_per'])  # 修改列名
total = pd.read_csv("d_1.csv")
total.columns = pd.Series(['country', 'total_per'])  # 修改列名
death = pd.read_csv("e_1.csv")
death.columns = pd.Series(['country', 'death_per'])  # 修改列名
new = pd.read_csv("c1_1.csv")
new.columns = pd.Series(['country', 'new_per'])  # 修改列名
people = pd.read_csv("people.csv")
new.columns = pd.Series(['country', 'people'])  # 修改列名

```

```
key = pd.concat([cure, total, death, new, people], axis=1, join='outer', ignore_index=True)  # 合并表
key.drop([2, 4, 6, 8], inplace=True, axis=1)  # 删除冗余列
key.columns = pd.Series(['country', 'cure_per', 'total_per', 'death_per', 'new_per', 'people'])  # 修改列名
key.drop(index=(key.loc[(key['country']=='钻石号邮轮')].index), inplace=True)  # 删除特殊数据
key.drop(index=(key.loc[(key['country']=='日本本土')].index), inplace=True)  # 删除特殊数据
key.drop(index=(key.loc[(key['death_per']==0)].index), inplace=True)  # 删除死亡人数为0的行
key.drop(index=(key.loc[(key['people']<5000)].index), inplace=True)  # 删除人口小于5000的国家

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
key.sort_values(by='score', ascending=False, inplace=True)  # 排序
print(key)
key.to_csv('score.csv')
```

3. 数据展示代码

a. 15天中，全球新冠疫情的总体变化趋势

```
import pandas as pd
import numpy as np
import matplotlib
import matplotlib.pyplot as plt

# 15天中，全球新冠疫情的总体变化趋势
matplotlib.rcParams['font.sans-serif'] = ['SimHei']  # 设置对中文的支持
plt.style.use('Solarize_Light2')

fig = plt.figure()
# fig, ax = plt.subplots()
ax1 = fig.add_subplot(211)
ax2 = fig.add_subplot(212)
csv_path = "cov19_Tencent.csv"
data = pd.read_csv(csv_path)
data[(data['dailyNew'] < 0)]['dailyNew'] = np.nan  # 将异常数据置为0

key = [data.groupby(['year', 'month', 'day'])['total'].sum()]  # 求得每日全球累计确诊数之和
key.append(data.groupby(['year', 'month', 'day'])['dailyNew'].sum())

keyy = pd.concat(key, axis=1)

print(keyy)

keyy.to_csv('a_1.csv')  # 储存每日全球累计确诊数之和
keyy = pd.read_csv('a_1.csv')
print(keyy)
```

```
28.
29.  x1 = np.arange(1, keyy.shape[0] + 1, 1)  # 图形 1x 数据
30.  y1 = (keyy['total']).tolist()  # 图形 1y 数据
31.  ax1.plot(x1, y1, marker='o', linestyle='--', alpha=0.5, color='g')  # 作图
32.
33.  x2 = np.arange(1, keyy.shape[0] + 1, 1)  # 图形 2x 数据
34.  y2 = (keyy['dailyNew']).tolist()  # 图形 2y 数据
35.  ax2.plot(x2, y2, marker='o', linestyle='--', alpha=0.5, color='r')  # 作图
36.
37.  xticks = []  # 构造 x 标签
38.  for i in range(keyy.shape[0]):
39.      tmplist = keyy.iloc[i].tolist()
40.      xticks.append(str(tmplist[0])+'.'+str(tmplist[1])+'.'+str(tmplist[2]))
41.      pass
42.  ax1.set_xticks(range(1, keyy.shape[0] + 1))  # 设置 x 轴刻度
43.  ax1.set_xticklabels(xticks)  # 设置 x 轴刻度标签
44.  ax1.set_xlabel("时间")  # 设置 x 轴上的标签
45.  ax1.set_ylabel("人数")  # 设置 y 轴上的标签
46.  ax1.set_title("全球变化趋势(累计)", fontsize=12, color='grey')  # 设置标题
47.
48.  ax2.set_xticks(range(1, keyy.shape[0] + 1))  # 设置 x 轴刻度
49.  ax2.set_xticklabels(xticks)  # 设置 x 轴刻度标签
50.  ax2.set_xlabel("时间")  # 设置 x 轴上的标签
51.  ax2.set_ylabel("人数")  # 设置 y 轴上的标签
52.  ax2.set_title("全球变化趋势(新增)", fontsize=12, color='grey')  # 设置标题
53.
54.  for a, b in zip(x1, y1):
55.      ax1.text(a - 0.25, b, '%d' % b, ha='center', va='center', fontsize=10)  # 数值点旁添加具体值
56.  for a, b in zip(x2, y2):
57.      ax2.text(a - 0.2, b, '%d' % b, ha='center', va='center', fontsize=10)  # 数值点旁添加具体值
58.
59.  plt.get_current_fig_manager().window.showMaximized()  # 窗口最大化
60.  plt.show()  # 显示
```

b. 累计确诊数排名前 20 的国家名称及其数量

```
1.  import pandas as pd
2.  import numpy as np
3.  import matplotlib
4.  import matplotlib.pyplot as plt
5.
6.  # 累计确诊数排名前 20 的国家名称及其数量
7.  matplotlib.rcParams['font.sans-serif'] = ['SimHei']  # 设置对中文的支持
8.  plt.style.use('Solarize_Light2')
9.
10.  fig, ax = plt.subplots()
```

```
csv_path = "cov19_AI.csv"
data = pd.read_csv(csv_path)

front = 20  # 取前 20 个国家

key = [data.groupby(['country'])['total'].max()]  # 每个国家累计确诊数
keyy = pd.concat(key, axis=1)
keyy.sort_values(by='total', ascending=False, inplace=True)
keyy.to_csv('b_1.csv')  # 储存每日全球累计确诊数之和
keyy = pd.read_csv('b_1.csv')
print(keyy)

x = np.arange(1, front + 1, 1)  # 图形 x 数据
y = [each for each in (keyy['total'] - keyy.iloc[front - 1][1] + 100000).tolist()[0:front]]  # 图形 y 数据
ax.bar(x, y, alpha=0.5, color='g')  # 作条形图

xticks = []  # 构造 x 标签
for i in range(front):
    tmplist = keyy.iloc[i].tolist()
    xticks.append(tmplist[0])
    pass
ax.set_xticks(range(1, front + 2))  # 设置 x 轴刻度
ax.set_xticklabels(xticks)  # 设置 x 轴刻度标签
ax.set_xlabel("国家")  # 设置 x 轴上的标签
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
ax.set_title("累计确诊前{}国".format(front), fontsize=12, color='grey')  # 设置标题

for a, b in zip(x, y):
    ax.text(a, b + keyy.iloc[0][1] // 70, '%d' % (b + keyy.iloc[front - 1][1] - 1),
            ha='center', va='center', fontsize=10)  # 数值点旁添加具体值

plt.get_current_fig_manager().window.showMaximized()  # 窗口最大化
plt.show()  # 显示
```

c. 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

```
import pandas as pd
import numpy as np
import matplotlib
import matplotlib.pyplot as plt

# 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
matplotlib.rcParams['font.sans-serif'] = ['SimHei']  # 设置对中文的支持
plt.style.use('Solarize_Light2')
```

```
9.
10.  fig, ax = plt.subplots()
11.  csv_path = "cov19_AI.csv"
12.  data = pd.read_csv(csv_path)
13.
14.  front = 10  # 取前 10 个国家
15.
16.  key = [data.groupby(['country'])['dailyNew'].sum()]  # 求得每个国家累计确诊数之和
17.  keyy = pd.concat(key, axis=1)
18.  keyy.sort_values(by='dailyNew', ascending=False, inplace=True)
19.  keyy.to_csv('c_1.csv')  # 储存
20.  keyy = pd.read_csv('c_1.csv')
21.  front_ten = []  # 提取前 10 国家名字
22.  for i in range(front):
23.      tmplist = keyy.iloc[i].tolist()
24.      front_ten.append(tmplist[0])
25.      pass
26.  print("15 天中，每日新增确诊数累计排名前 10 的国家")
27.  print(front_ten)
28.  print()
29.
30.  # 画 10 条曲线
31.  for i in range(len(front_ten)):
32.      tmp_data = data[(data['country'] == front_ten[i])]
33.      x = np.arange(1, tmp_data.shape[0] + 1, 1)  # 图形 x 数据
34.      y = (tmp_data['dailyNew']).tolist()  # 图形 y 数据
35.      plt.plot(x, y)
36.      if i == 0:
37.          for a, b in zip(x, y):
38.              ax.text(a, b - 7000, '%d' % (b + keyy.iloc[front - 1][1] - 1),
39.                      ha='center', va='center', fontsize=10)  # 数值点旁添加具体值
40.  xticks = []  # 构造 x 标签
41.  for i in range(tmp_data.shape[0]):
42.      tmplist = tmp_data.iloc[i].tolist()
43.      xticks.append(str(tmplist[0]) + '.' + str(tmplist[1]) + '.' + str(tmplist[2]))
44.      pass
45.  ax.set_xticks(range(1, tmp_data.shape[0] + 1))  # 设置 x 轴刻度
46.  print(tmp_data)
47.
48.  ax.legend(front_ten, loc='upper left')  # 设置图例
49.  ax.set_xticklabels(xticks)  # 设置 x 轴刻度标签
50.  ax.set_xlabel("时间")  # 设置 x 轴上的标签
51.  ax.set_ylabel("人数")  # 设置 y 轴上的标签
52.  ax.set_title("每日新增确诊数据", fontsize=12, color='grey')  # 设置标题
```

```
53.
54.  plt.get_current_fig_manager().window.showMaximized()  # 窗口最大化
55.  plt.show()  # 显示
```

d. 累计确诊人数占国家总人口比例最高的 10 个国家

```
1.   import pandas as pd
2.   import numpy as np
3.   import matplotlib
4.   import matplotlib.pyplot as plt
5.
6.   # 累计确诊人数占国家总人口比例最高的 10 个国家
7.   matplotlib.rcParams['font.sans-serif'] = ['SimHei']  # 设置对中文的支持
8.   plt.style.use('Solarize_Light2')
9.
10.  fig, ax = plt.subplots()
11.  csv_path = "cov19_AI.csv"
12.
13.  front = 10  # 取前 10 个国家
14.
15.  data = pd.read_csv(csv_path)
16.  data['people'] = data['people'].replace('-', 'NaN')  # 数据清洗：用 NaN 代替-
17.  data['people'] = data['people'].astype('float')  # 数据转化为 float
18.
19.  key = [data.groupby(['country'])['total'].max() / (data.groupby(['country'])['people'].max() * 100)
     ] # 每个国家百分比
20.  keyy = pd.concat(key, axis=1)  # 转化为 df
21.  keyy.to_csv('d_1.csv')  # 储存
22.  keyy = pd.read_csv('d_1.csv')
23.  keyy.columns = pd.Series(['country', 'per'])  # 修改列名
24.  keyy.drop(index=(keyy.loc[(keyy['country']=='钻石号邮轮')].index), inplace=True)  # 删除特殊数据
25.  keyy.sort_values(by='per', ascending=False, inplace=True)  # 排序
26.  print(keyy)
27.
28.  x = np.arange(1, front + 1, 1)  # 图形 x 数据
29.  y = [each for each in keyy['per'].tolist()[0:front]]  # 图形 y 数据
30.  ax.bar(x, y, alpha=0.5, color='g')  # 作条形图
31.
32.  xticks = []  # 构造 x 标签
33.  for i in range(front):
34.      tmplist = keyy.iloc[i].tolist()
35.      xticks.append(tmplist[0])
36.      pass
37.  ax.set_xticks(range(1, front + 2))  # 设置 x 轴刻度
38.  ax.set_xticklabels(xticks)  # 设置 x 轴刻度标签
39.  ax.set_xlabel("国家")  # 设置 x 轴上的标签
```

```
ax.set_ylabel("百分比")  # 设置 y 轴上的标签
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
print(xticks)

for a, b in zip(x, y):
    ax.text(a, b + 0.1, '{:.2f}%'.format(b),
            ha='center', va='center', fontsize=10)  # 数值点旁添加具体值

plt.get_current_fig_manager().window.showMaximized()  # 窗口最大化
plt.show()  # 显示
```

e. 死亡率（累计死亡人数/累计确诊人数）最低的 10 个国家

```
import pandas as pd
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
# 死亡率

matplotlib.rcParams['font.sans-serif'] = ['SimHei']  # 设置对中文的支持
plt.style.use('Solarize_Light2')

fig, ax = plt.subplots()
csv_path = "cov19_AI.csv"

front = 10  # 取前 10 个国家

data = pd.read_csv(csv_path)
data['death'] = data['death'].astype('float')  # 数据转化为 float
data['total'] = data['total'].astype('float')  # 数据转化为 float

key = [data.groupby(['country'])['death'].max() / data.groupby(['country'])['total'].max() * 100]
# 每个国家百分比
keyy = pd.concat(key, axis=1)  # 转化为 df
keyy.to_csv('e_1.csv')  # 储存
keyy = pd.read_csv('e_1.csv')
keyy.columns = pd.Series(['country', 'death_per'])  # 修改列名
keyy.drop(index=(keyy.loc[(keyy['country']=='钻石号邮轮')].index), inplace=True)  # 删除特殊数据
keyy.drop(index=(keyy.loc[(keyy['death_per']==0)].index), inplace=True)  # 删除死亡人数为 0 的行
keyy.sort_values(by='death_per', ascending=True, inplace=True)  # 排序
print(keyy)

x = np.arange(1, front + 1, 1)  # 图形 x 数据
y = [each for each in keyy['death_per'].tolist()[0:front]]  # 图形 y 数据
ax.bar(x, y, alpha=0.5, color='g')  # 作条形图

```

```
xticks = []  # 构造 x 标签
for i in range(front):
    tmplist = keyy.iloc[i].tolist()
    xticks.append(tmplist[0])
    pass
ax.set_xticks(range(1, front + 2))  # 设置 x 轴刻度
ax.set_xticklabels(xticks)  # 设置 x 轴刻度标签
ax.set_xlabel("国家")  # 设置 x 轴上的标签
ax.set_ylabel("百分比")  # 设置 y 轴上的标签
ax.set_title("死亡率最低的 10 个国家", fontsize=12, color='grey')  # 设置标题
print(xticks)

for a, b in zip(x, y):
    ax.text(a, b + 0.01, '{:.2f}%'.format(b),
            ha='center', va='center', fontsize=10)  # 数值点旁添加具体值

plt.get_current_fig_manager().window.showMaximized()  # 窗口最大化
plt.show()  # 显示
```

f. 用饼图展示各个国家的累计确诊人数的比例

```
import pandas as pd
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
from matplotlib import font_manager as fm

# 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）
matplotlib.rcParams['font.sans-serif'] = ['SimHei']  # 设置对中文的支持

fig, ax = plt.subplots()
csv_path = "b_1.csv"
keyy = pd.read_csv(csv_path)  # 直接利用第 b 问的数据

border = 600000  # 合并小于 border 国家 为 "其他"
borderBySort = keyy.iloc[9][1]  # 或者合并 10 名以后的国家 为 "其他"

keyy.loc[keyy.shape[0]] = ["其他", 0]  # 添加 "其他" 列，初始化为 0

drop_lines = []  # 记录要删除的行

for i in range(keyy['country'].count() - 1):
    if keyy.iloc[i][1] < borderBySort:
        keyy.iloc[keyy.loc[(keyy['country'] == '其他')].index, 1] += keyy.iloc[i][1]  # 修改 "其他" 行
        drop_lines.append(i)  # 记录要删除的行
```

```
            pass
        pass
    keyy.drop(index=drop_lines, inplace=True)  # 删除数据
    keyy = keyy.reset_index(drop=True)  # 重设索引

    print(keyy)

    # 作图
    size = []  # 各国家的人数
    label_list = []  # 各部分标签
    for i in range(keyy.shape[0]):
        tmplist = keyy.iloc[i].tolist()
        size.append(tmplist[1])
        label_list.append(tmplist[0])
        pass

    explode = np.zeros(keyy.shape[0])  # 各部分的突出显示比例(法 1)
    explode[0] = 0.1
    # explode = np.linspace(0.2, 0.05, keyy.shape[0])  # 各部分的突出显示比例(法 2)
    patches, texts, autotexts = plt.pie(size, explode=explode, labels=label_list,
                                        labeldistance=1.1, autopct="%1.1f%%", shadow=False, startangle=
    0, pctdistance=0.9)
    # 调整字体
    proptease = fm.FontProperties()
    proptease.set_size('x-large')
    plt.setp(texts, fontproperties=proptease)
    plt.setp(autotexts, fontproperties=proptease)
    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
    plt.get_current_fig_manager().window.showMaximized()  # 窗口最大化
    plt.show()
```

g. 展示全球各个国家累计确诊人数的箱型图，要有平均值

```
import pandas as pd
import matplotlib
import matplotlib.pyplot as plt
from util import util

# 展示全球各个国家累计确诊人数的箱型图，要有平均值
matplotlib.rcParams['font.sans-serif'] = ['SimHei']  # 设置对中文的支持

fig, ax = plt.subplots()
csv_path = "b_1.csv"
keyy = pd.read_csv(csv_path)  # 直接利用第 b 问的数据
avee = keyy['total'].mean()
```

```
maxx = keyy['total'].max()
medd = keyy['total'].median()
upperQuartile = keyy['total'].quantile(0.75)
lowerQuartile = keyy['total'].quantile(0.25)

x = keyy['total'].tolist()  # 图形数据
f = plt.boxplot(x, showmeans=True, vert=False, meanline=True, widths=0.5,  showfliers=False)

ax.set_xlabel("人数")  # 设置 x 轴上的标签
ax.set_yticklabels(['全球'])  # 设置 y 轴刻度标签
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

# 设置关键点注释
ax.annotate("平均值={:.2f}人".format(avee), xy=(avee, 1.05),
            xytext=(avee - 70000, 1.1), arrowprops=dict(arrowstyle="->"))
ax.annotate("最大值={:.0f}人".format(maxx), xy=(maxx, 1),
            xytext=(maxx - 50000, 1.1), arrowprops=dict(arrowstyle="->"))
ax.annotate("中位数={:.0f}人".format(medd), xy=(medd, 1.05),
            xytext=(medd + 50000, 1.1), arrowprops=dict(arrowstyle="->"))
ax.annotate("上四分位点={:.0f}人".format(upperQuartile), xy=(upperQuartile, 0.95),
            xytext=(upperQuartile + 50000, 0.9), arrowprops=dict(arrowstyle="->"))
ax.annotate("下四分位点={:.0f}人".format(lowerQuartile), xy=(lowerQuartile, 0.95),
            xytext=(lowerQuartile + 50000, 0.9), arrowprops=dict(arrowstyle="->"))

plt.get_current_fig_manager().window.showMaximized()  # 窗口最大化
plt.show()
```

h. 其他展示

```
import pandas as pd
import numpy as np
import matplotlib
import matplotlib.pyplot as plt

# 中国
matplotlib.rcParams['font.sans-serif'] = ['SimHei']  # 设置对中文的支持
plt.style.use('Solarize_Light2')

fig, ax = plt.subplots()
csv_path = "cov19_AI.csv"
data = pd.read_csv(csv_path)

keyyy = data[(data['country'] == '中国')][['year', 'month', 'day', 'dailyNew']]

x = np.arange(1, keyyy.shape[0] + 1, 1)  # 图形 2x 数据
y = (keyyy['dailyNew']).tolist()  # 图形 2y 数据
```

```
    ax.plot(x, y, marker='o', linestyle='--', alpha=0.5, color='r')  # 作图

    xticks = []  # 构造 x 标签
    for i in range(keyyy.shape[0]):
        tmplist = keyyy.iloc[i].tolist()
        xticks.append(str(tmplist[0])+'.'+str(tmplist[1])+'.'+str(tmplist[2]))
        pass

    ax.set_xticks(range(1, keyyy.shape[0] + 1))  # 设置 x 轴刻度
    ax.set_xticklabels(xticks)  # 设置 x 轴刻度标签
    ax.set_xlabel("时间")  # 设置 x 轴上的标签
    ax.set_ylabel("人数")  # 设置 y 轴上的标签
    ax.set_title("中国每日新增", fontsize=12, color='grey')  # 设置标题
    plt.show()
```

```
    import pandas as pd
    import numpy as np
    import matplotlib
    import matplotlib.pyplot as plt

    # 治愈
    matplotlib.rcParams['font.sans-serif'] = ['SimHei']  # 设置对中文的支持
    plt.style.use('Solarize_Light2')

    fig, ax = plt.subplots()
    csv_path = "cov19_AI.csv"

    front = 10  # 取前 10 个国家

    data = pd.read_csv(csv_path)
    data['death'] = data['cure'].astype('float')  # 数据转化为 float
    data['total'] = data['total'].astype('float')  # 数据转化为 float

    key = [data.groupby(['country'])['cure'].max() / data.groupby(['country'])['total'].max() * 100]
    # 每个国家百分比
    keyy = pd.concat(key, axis=1)  # 转化为 df
    keyy.to_csv('cure_1.csv')  # 储存
    keyy = pd.read_csv('cure_1.csv')
    keyy.columns = pd.Series(['country', 'cure_per'])  # 修改列名
    keyy.drop(index=(keyy.loc[(keyy['country']=='钻石号邮轮')].index), inplace=True)  # 删除特殊数据
    keyy.drop(index=(keyy.loc[(keyy['cure_per']==0)].index), inplace=True)  # 删除死亡人数为 0 的行
    keyy.sort_values(by='cure_per', ascending=False, inplace=True)  # 排序
    print(keyy)

```

```
x = np.arange(1, front + 1, 1)  # 图形 x 数据
y = [each for each in keyy['cure_per'].tolist()[0:front]]  # 图形 y 数据
ax.bar(x, y, alpha=0.5, color='g')  # 作条形图

xticks = []  # 构造 x 标签
for i in range(front):
    tmplist = keyy.iloc[i].tolist()
    xticks.append(tmplist[0])
    pass
ax.set_xticks(range(1, front + 2))  # 设置 x 轴刻度
ax.set_xticklabels(xticks)  # 设置 x 轴刻度标签
ax.set_xlabel("国家")  # 设置 x 轴上的标签
ax.set_ylabel("百分比")  # 设置 y 轴上的标签
ax.set_title("治愈率最高的 10 个国家", fontsize=12, color='grey')  # 设置标题

for a, b in zip(x, y):
    ax.text(a, b + 0.9, '{:.2f}%'.format(b),
            ha='center', va='center', fontsize=10)  # 数值点旁添加具体值

plt.get_current_fig_manager().window.showMaximized()  # 窗口最大化
plt.show()  # 显示
```

4. 数据预测代码

```
import pandas as pd
import matplotlib
import matplotlib.pyplot as plt
import numpy as np
from scipy.optimize import nnls
import math
from sklearn.linear_model import LinearRegression


# 图形界面预设以及生成画布与坐标轴
matplotlib.rcParams['font.sans-serif'] = ['SimHei']  # 设置对中文的支持
fig, ax = plt.subplots()

# 读取数据
csv_path = "a_1.csv"
keyy = pd.read_csv(csv_path)  # 直接利用第 a 问的数据

# 进一步设定初始数据
preDays = 5  # 预测的天数
real = keyy['total'].values.tolist()  # 取得真实数据
history = keyy['total'].values.tolist()  # 取得历史数据
```

```
for i in range(preDays):
    history.pop()  # 删除后几天的数据

# ------------------------------------------------------------------------ #
# GM(1,1)预测

pre1 = []  # 存放预测的数据
n = len(history)  # 用于训练的数据量
pre1.append(history[-1])  # 便于画图，添加一项真实数据

X0 = np.array(history)
X1 = np.array([sum(history[0:i+1]) for i in range(n)])  # 对数据作一次累加

# 构造数据矩阵 B 及数据向量 Y
B = np.zeros([n-1, 2])
Y = np.zeros([n-1, 1])
for i in range(0, n-1):
    B[i, 0] = -0.5*(X1[i] + X1[i+1])
    B[i, 1] = 1
    Y[i, 0] = X0[i+1]

# 计算 GM(1,1)微分方程的参数 a 和 b
U = np.linalg.inv(B.T.dot(B)).dot(B.T).dot(Y)
a = U[0][0]
b = U[1][0]

# 预测
f = np.zeros(preDays)
print('GM(1,1)预测:')
print('时间          预测          实际          误差')
for i in range(0, preDays):
    tmplist = keyy.iloc[keyy.shape[0] - preDays + i].tolist()
    date = str(tmplist[0]) + '.' + str(tmplist[1]) + '.' + str(tmplist[2]) + '\t'
    f[i] = (X0[0] - b / a) * (1 - math.exp(a)) * math.exp(-a * (i + n))  # 此次预测的值
    pre1.append(int(f[i]))
    print(date, '{:.2f}'.format(f[i]), '\t', real[keyy.shape[0] - preDays + i], '\t',
          '{:.2f}%'.format((f[i] - real[keyy.shape[0] - preDays + i]) / real[keyy.shape[0] - preDa
ys + i] * 100))

# ------------------------------------------------------------------------ #
# k 阶差分预测
pre2 = []
pre2.append(history[-1])
```

```
k = 8  # k 阶差分预测

n = len(history)
x = [history[k - i: n-i] for i in range(1, k+1)]
x.append([1 for j in range(n-k)])
x = np.array(x).T
y = np.array(history[k: n]).T

paras = nnls(x, y)[0]

sq = history.copy()
j = n
print('k 阶差分预测:')
print('时间          预测              实际          误差')
for i in range(preDays):
    tmplist = keyy.iloc[keyy.shape[0] - preDays + i].tolist()
    date = str(tmplist[0]) + '.' + str(tmplist[1]) + '.' + str(tmplist[2]) + '\t'
    temp = 0
    for j in range(k):
        temp += paras[j] * sq[len(sq)-j-1]
    temp += paras[k]
    pre2.append(int(temp))
    sq.append(temp)
    # print(temp)
    print(date, '{:.2f}'.format(temp), '\t', real[keyy.shape[0] - preDays + i], '\t',
          '{:.2f}%'.format((temp - real[keyy.shape[0] - preDays + i]) / real[keyy.shape[0] - preDa
ys + i] * 100))


# ------------------------------------------------------------------------ #
# 一元线性回归
pre3 = []
pre3.append(history[-1])

Model = LinearRegression()
Model.fit(np.arange(1, keyy.shape[0] + 1 - preDays, 1).reshape(-1, 1), history)

print('一元线性回归预测:')
print('时间          预测              实际          误差')
for i in range(0, preDays):
    tmplist = keyy.iloc[keyy.shape[0] - preDays + i].tolist()
    date = str(tmplist[0]) + '.' + str(tmplist[1]) + '.' + str(tmplist[2]) + '\t'
    tmppre = Model.predict([[keyy.shape[0] - preDays + i + 1]]).tolist()[0]
    pre3.append(int(tmppre))
```

```
    print(date, '{:.2f}'.format(tmppre), '\t', real[keyy.shape[0] - preDays + i], '\t',
          '{:.2f}%'.format((tmppre - real[keyy.shape[0] - preDays + i]) / real[keyy.shape[0] - preDays + i] * 100))


# ---------------------------------------------------------------------- #
# 作图
x1 = np.arange(1, keyy.shape[0] + 1, 1)
y1 = real  # 实际数据
x2 = np.arange(keyy.shape[0] - preDays, keyy.shape[0] + 1, 1)
y2 = pre1  # 预测数据
x3 = np.arange(keyy.shape[0] - preDays, keyy.shape[0] + 1, 1)
y3 = pre2  # 预测数据
x4 = np.arange(keyy.shape[0] - preDays, keyy.shape[0] + 1, 1)
y4 = pre3  # 预测数据

ax.plot(x1, y1, marker='o', linestyle='-', alpha=0.5, color='r')  # 作图
ax.plot(x2, y2, marker='o', linestyle='--', alpha=0.5, color='b')  # 作图
ax.plot(x3, y3, marker='o', linestyle='--', alpha=0.5, color='g')  # 作图
ax.plot(x4, y4, marker='o', linestyle='--', color='y')  # 作图

xticks = []  # 构造 x 标签
for i in range(keyy.shape[0]):
    tmplist = keyy.iloc[i].tolist()
    xticks.append(str(tmplist[0])+'.'+str(tmplist[1])+'.'+str(tmplist[2]))
    pass
ax.set_xticks(range(1, keyy.shape[0] + 1))  # 设置 x 轴刻度
ax.set_xticklabels(xticks)  # 设置 x 轴刻度标签
ax.set_xlabel("时间")  # 设置 x 轴上的标签
ax.set_ylabel("人数")  # 设置 y 轴上的标签
ax.set_title("全球变化趋势(预测{}天)".format(preDays), fontsize=12, color='grey')  # 设置标题
first = True
for a, b in zip(x1, y1):
    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
for a, b in zip(x2, y2):
    if first:
        first = False
    else:
        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
first = True
for a, b in zip(x3, y3):
    if first:
```

```
            first = False
        else:
            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
    first = True
    for a, b in zip(x4, y4):
        if first:
            first = False
        else:
            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
    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
    plt.grid(True, linestyle='--', alpha=0.3, color='grey')  # 设置网格线
    plt.get_current_fig_manager().window.showMaximized()  # 窗口最大化
    plt.show()
```