
北京邮电大学

Beijing University of Posts and Telecommunications

实 验 报 告

题目：期末大作业-Covid-19 疫情数据分析

课 程 名 称：Python 程序设计

姓 名（学号）：凌国瀚 (2018213344)

学 院：计算机学院

专 业：网络工程

指 导 教 师：杨亚老师

二〇二〇年 十二月

目录

实 验 报 告	1
1. 题目要求.....	3
2. 数据来源与爬虫.....	3
3. 数据分析与展示.....	5
3.1 全球新冠疫情的总体变化趋势.....	5
3.2 累计确诊数排名前 20 的国家名称及其数量.....	6
3.3 日新增确诊数累计排名前 10 的国家.....	6
3.4 累计确诊人数占国家总人口比例最高的 10 个国家.....	7
3.5 死亡率最低的 10 个国家.....	8
3.6 展示各个国家的累计确诊人数的比例.....	8
3.7 展示全球各个国家累计确诊人数的箱型图.....	9
3.8 康复率最高的 10 个国家.....	11
4. 分析全世界应对新冠疫情最好的 10 个国家.....	12
5. 疫情预测.....	13
6. 源代码.....	15
6.1 爬虫.....	15
6.2 数据分析.....	16
6.2.1 Part-A	16
6.2.2 Part-B	18
6.2.3 Part-C	21
6.2.4 Part-D	22
6.2.5 Part-E	23
6.2.6 Part-F.....	24
6.2.7 Part-G	25
6.2.8 Extra	27
6.3 最佳的 10 个国家.....	27
6.4 预测分析.....	28

1. 题目要求

1. 标明你的数据来源：包括网址和首页截图
2. 数据分析和展示应包括：
 - a) 15 天中，全球新冠疫情的总体变化趋势；
 - b) 累计确诊数排名前 20 的国家名称及其数量；
 - 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；
 - d) 累计确诊人数占国家总人口比例最高的 10 个国家；
 - e) 死亡率（累计死亡人数/累计确诊人数）最低的 10 个国家；
 - f) 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）；
 - g) 展示全球各个国家累计确诊人数的箱型图，要有平均值；
 - h) 其它你希望分析和展示的数据。

以上图形应包括完整的坐标、刻度、标签、图例等，如有必要请配上说明文字，对图中的内容进行解释。
3. 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。
4. 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，并与实际数据进行对比。说明你预测的方法，并分析与实际数据的差距和原因。
5. 请贴上爬虫程序的核心代码、数据处理及数据展示的核心代码（应有足够的注释）；
6. 以 pdf 格式提交到爱课堂平台上，文件名为学号，总页数不超过 30 页。
7. 截止时间 12 月 27 日 24 点。

2. 数据来源与爬虫

疫情数据来源于 <https://ncov2019.live/>，首页截图如下。

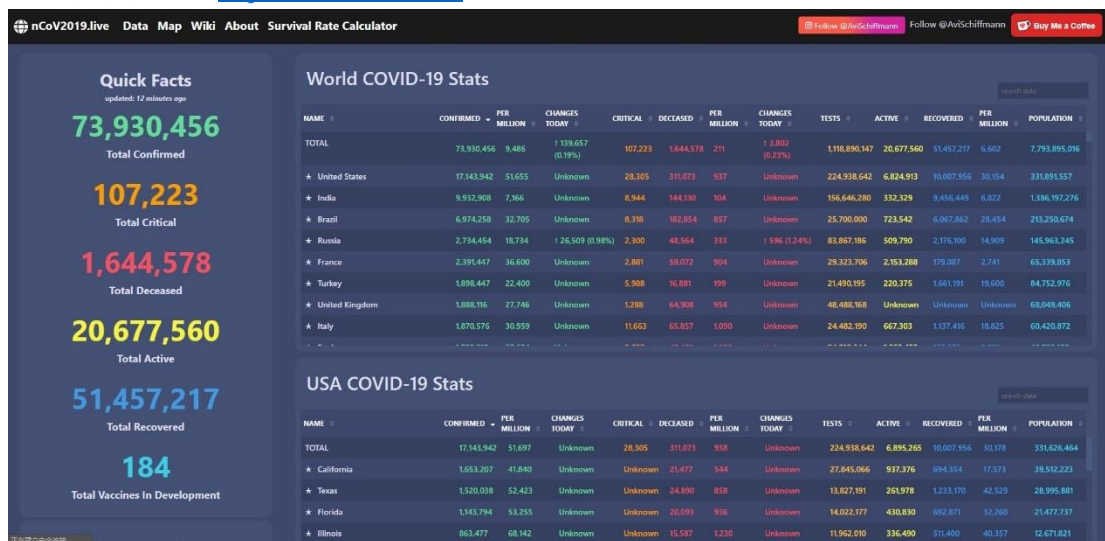
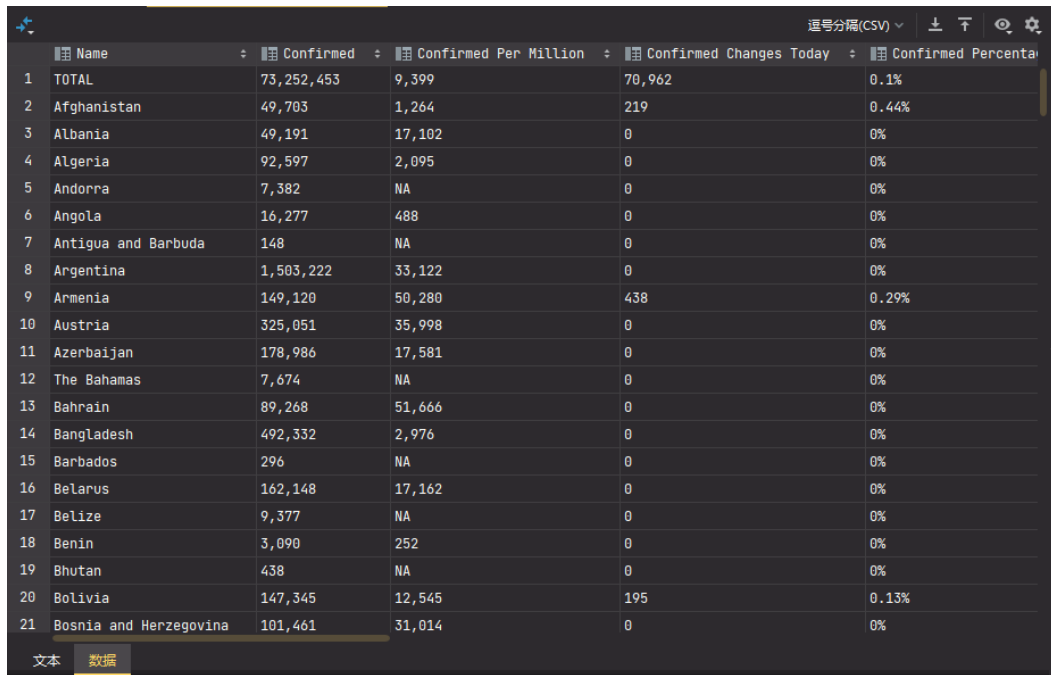


图 1 数据源网站首页截图

利用 `requests` 库和 `bs4` 库在 12 月 1 日至 12 月 15 日每日北京时间下午 5 点爬取数据，取得的数据用 `csv` 格式存储，数据示例如下：



	Name	Confirmed	Confirmed Per Million	Confirmed Changes Today	Confirmed Percentage
1	TOTAL	73,252,453	9,399	70,962	0.1%
2	Afghanistan	49,703	1,264	219	0.44%
3	Albania	49,191	17,102	0	0%
4	Algeria	92,597	2,095	0	0%
5	Andorra	7,382	NA	0	0%
6	Angola	16,277	488	0	0%
7	Antigua and Barbuda	148	NA	0	0%
8	Argentina	1,503,222	33,122	0	0%
9	Armenia	149,120	50,280	438	0.29%
10	Austria	325,051	35,998	0	0%
11	Azerbaijan	178,986	17,581	0	0%
12	The Bahamas	7,674	NA	0	0%
13	Bahrain	89,268	51,666	0	0%
14	Bangladesh	492,332	2,976	0	0%
15	Barbados	296	NA	0	0%
16	Belarus	162,148	17,162	0	0%
17	Belize	9,377	NA	0	0%
18	Benin	3,090	252	0	0%
19	Bhutan	438	NA	0	0%
20	Bolivia	147,345	12,545	195	0.13%
21	Bosnia and Herzegovina	101,461	31,014	0	0%

图 2 爬取的 csv 文件

数据包含如下字段:

```
data = {'Name': '', 'Confirmed': '', 'Confirmed Per Million': '', 'Confirmed Changes Today': '', 'Confirmed Percentage Day Change': '', 'Critical': '', 'Deceased': '', 'Deceased Per Million': '', 'Deceased Changes Today': '', 'Death Percentage Day Change': '', 'Tests': '', 'Active': '', 'Recovered': '', 'Recovered Per Million': '', 'Population': ''}
```

3. 数据分析与展示

3.1 全球新冠疫情的总体变化趋势

通过分析 15 日全球新冠疫情数据，得到如下数据分析图：

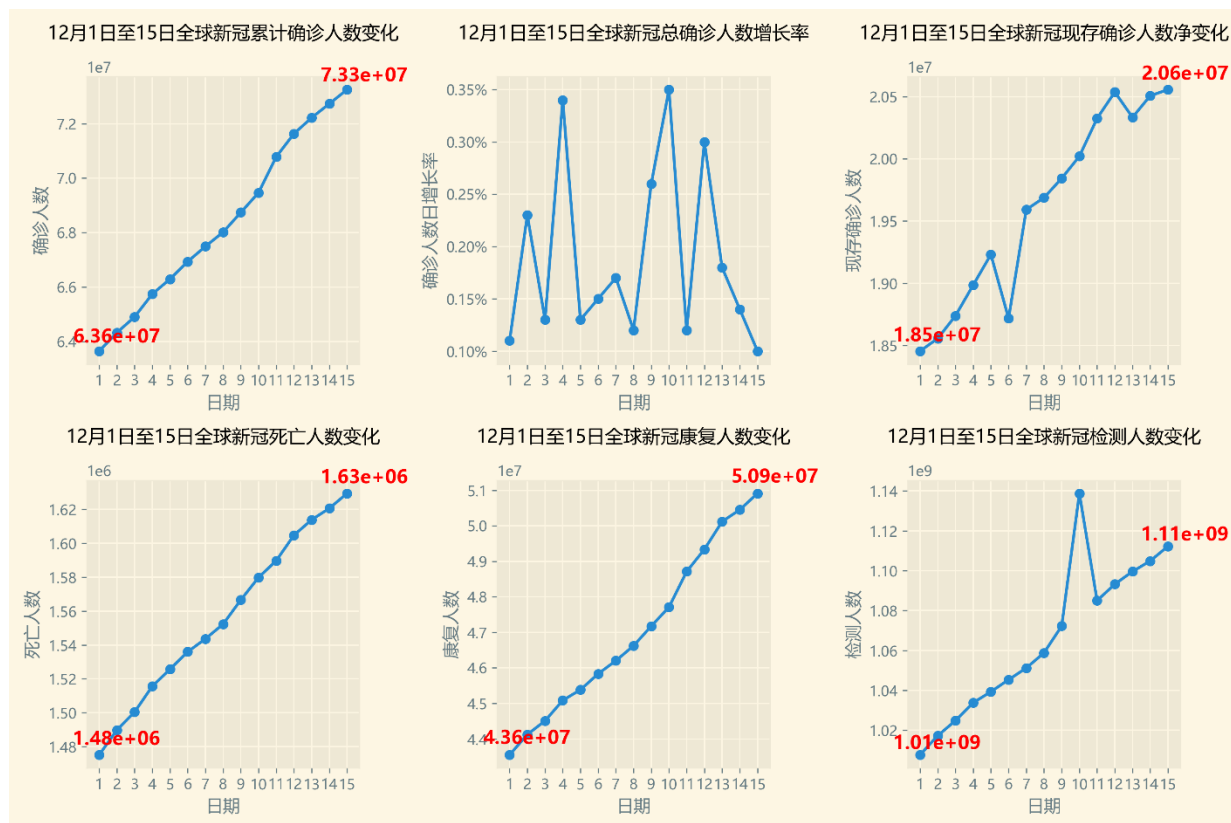


图 3 全球疫情总体变化趋势

可以看出在这 15 日内，全球新冠确诊人数仍然在持续上涨，累计确诊人数从 1 日的 6.36 千万人增长至 7.33 千万人，净确诊人数（总增长人数减去死亡与治愈人数）从 1.85 千万人增长至 2.06 千万人，总确诊人数日增长率维持在 0.10% 到 0.35% 之间。

另外，全球死亡人数、康复人数与检测人数则呈现类似线性的增长趋势上升。注意到全球检测人数的数据在 12 月 10 日出现异常，经过分析后判断是数据源提供的数据出现问题，在之后的分析中将特别注意忽略 12 月 10 日的检测人数数据。

3.2 累计确诊数排名前 20 的国家名称及其数量

利用 12 月 15 日的数据，分析累计确诊数排名前 20 的国家名称及其数量，得到如下柱状图：

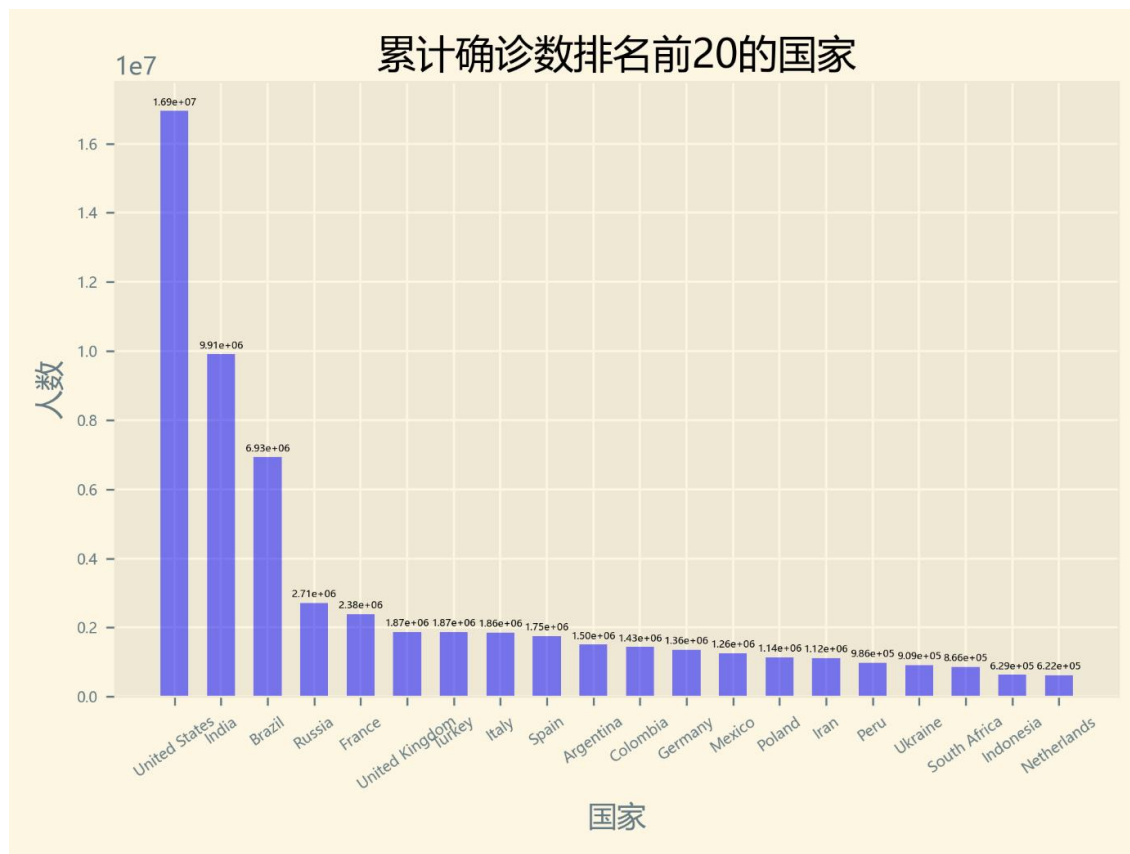


图 4 累计确诊数排名前 20 的国家

其中，美国、印度与巴西的累计确诊人数数据较为突出，分别为 1.69 千万人、9.91 百万人和 6.93 百万人。而其它所有国家的累计确诊数据均不超过 3 百万人。

3.3 日新增确诊数累计排名前 10 的国家

分析 15 天中，每日新增确诊数累计排名前 10 的国家，并作从 2 日开始的每日新增确诊数据的曲线图如下所示：

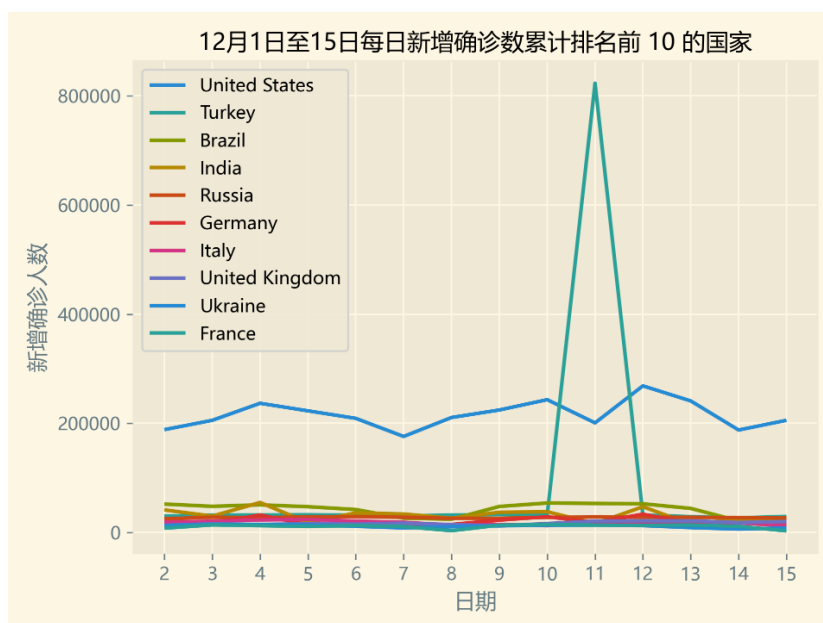


图 5 日新增确诊数累计排名前 10 的国家

图例由上至下为该排名的先后顺序。美国为每日新增确诊人数最多的国家，其每日新增人数远远超过其它任何国家。然而，在 12 月 11 日，土耳其新增确诊人数突然增加，这是由于土耳其疫情反弹，检测人数大量增加导致的（相关新闻链接：<https://baijiahao.baidu.com/s?id=1685787350858665817&wfr=spider&for=pc>）。

3.4 累计确诊人数占国家总人口比例最高的 10 个国家

利用 12 月 15 日的数据，分析累计确诊人数占国家总人口比例最高的 10 个国家，得到如下柱状图：

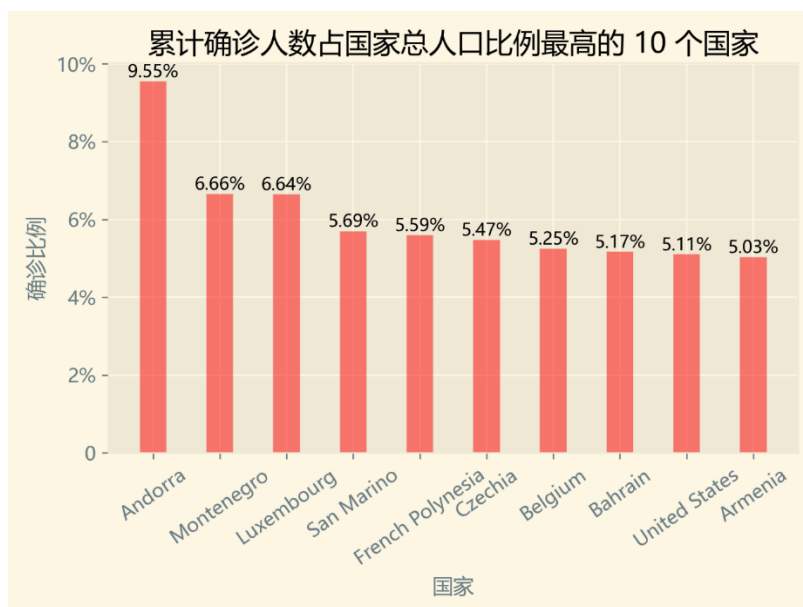


图 6 累计确诊人数占国家总人口比例最高的 10 个国家

安道尔公国是累计确诊人数占国家总人口比例最高的国家，这是由于其位于欧洲南部，紧邻欧洲疫情中心，且其本身土地面积小（468 平方公里）、人口总数少（77321 人）。美国这是这 10 个国家中人口总数最高的国家。

3.5 死亡率最低的 10 个国家

利用 12 月 15 日的数据，分析死亡率最低的 10 个国家，得到如下柱状图：

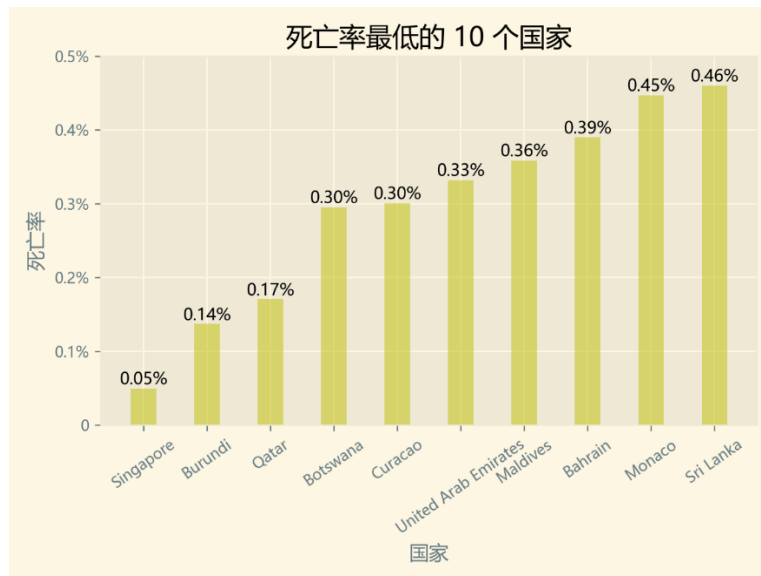


图 7 死亡率最低的 10 个国家

新加坡是全球新冠疫情死亡率最低的国家，这也是在后续分析应对疫情最好的国家中，该国取得较好成绩的主要原因。

3.6 展示各个国家的累计确诊人数的比例

在 15 日数据的基础上，利用饼图绘制展示各个国家的累计确诊人数的比例，饼图如下所示：

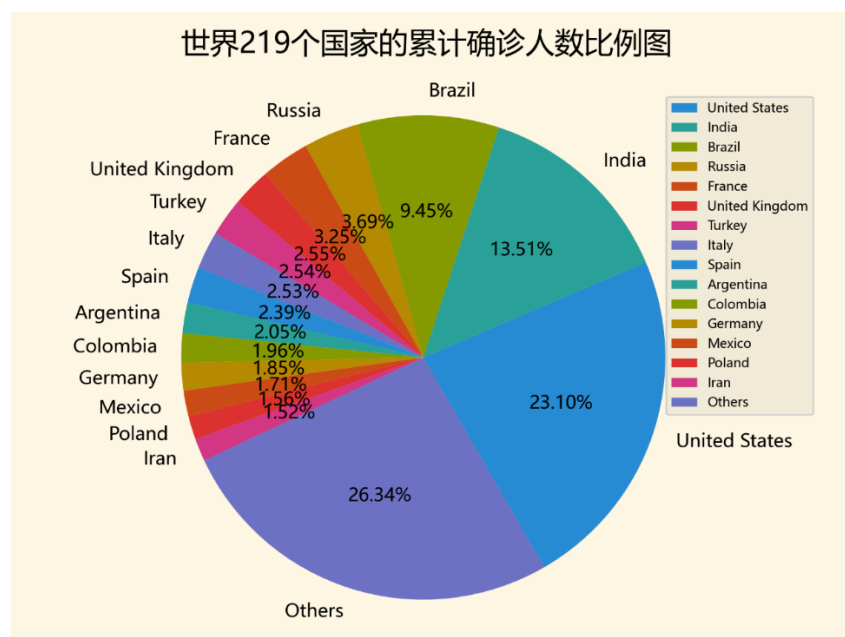


图 8 世界 219 个国家的累计确诊人数比例图

其中，确诊人数小于一百万的国家被归为其它。

3.7 展示全球各个国家累计确诊人数的箱型图

通过第 15 日数据分析做出确诊人数箱型图如下：

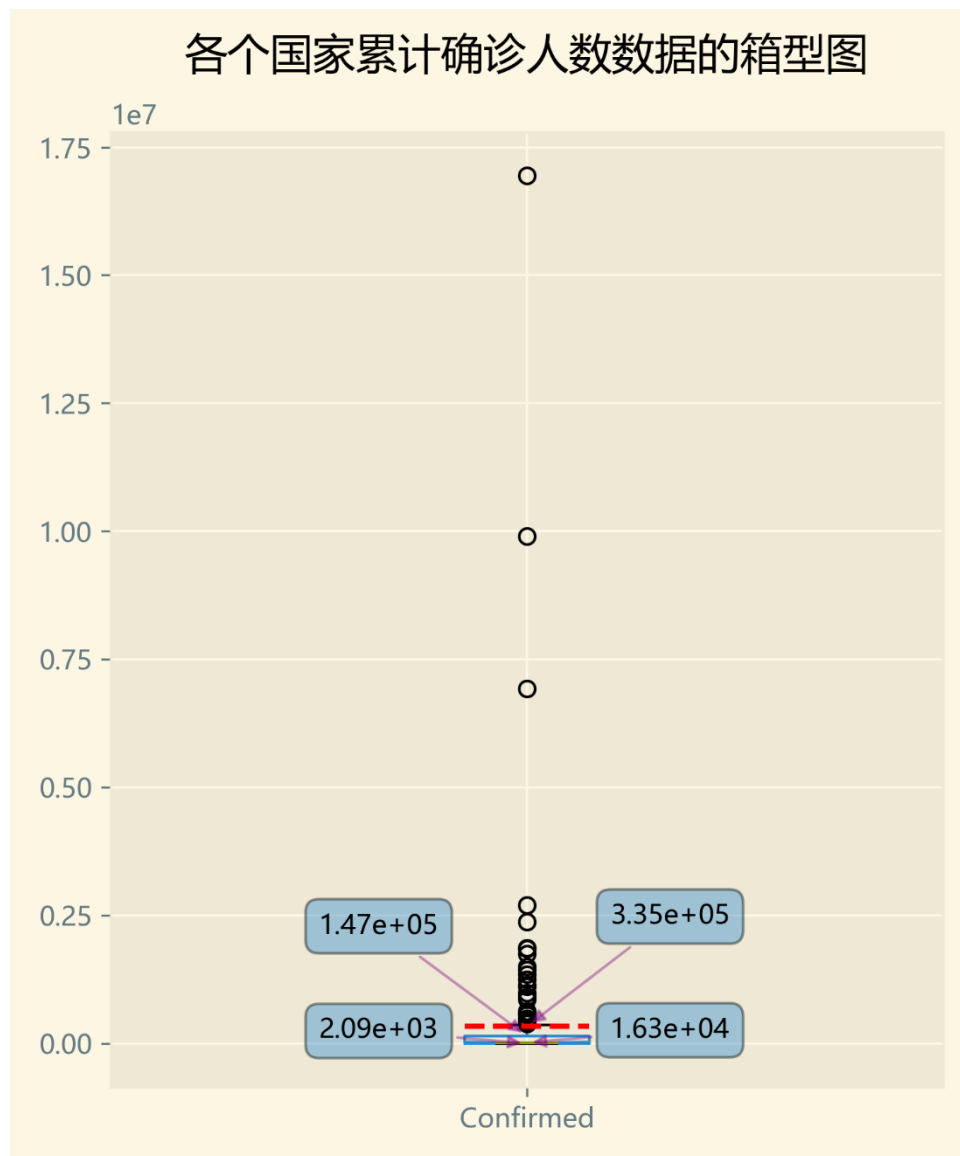


图 9 原始箱型图

可以看出，由于异常值过大，箱型图中的图例难以准确展示，且归一化后效果也不好。由于在此图中可以明显看出数据左偏较为严重，因此，删除确诊数据中最高的 5 条，然后重新作图如下：

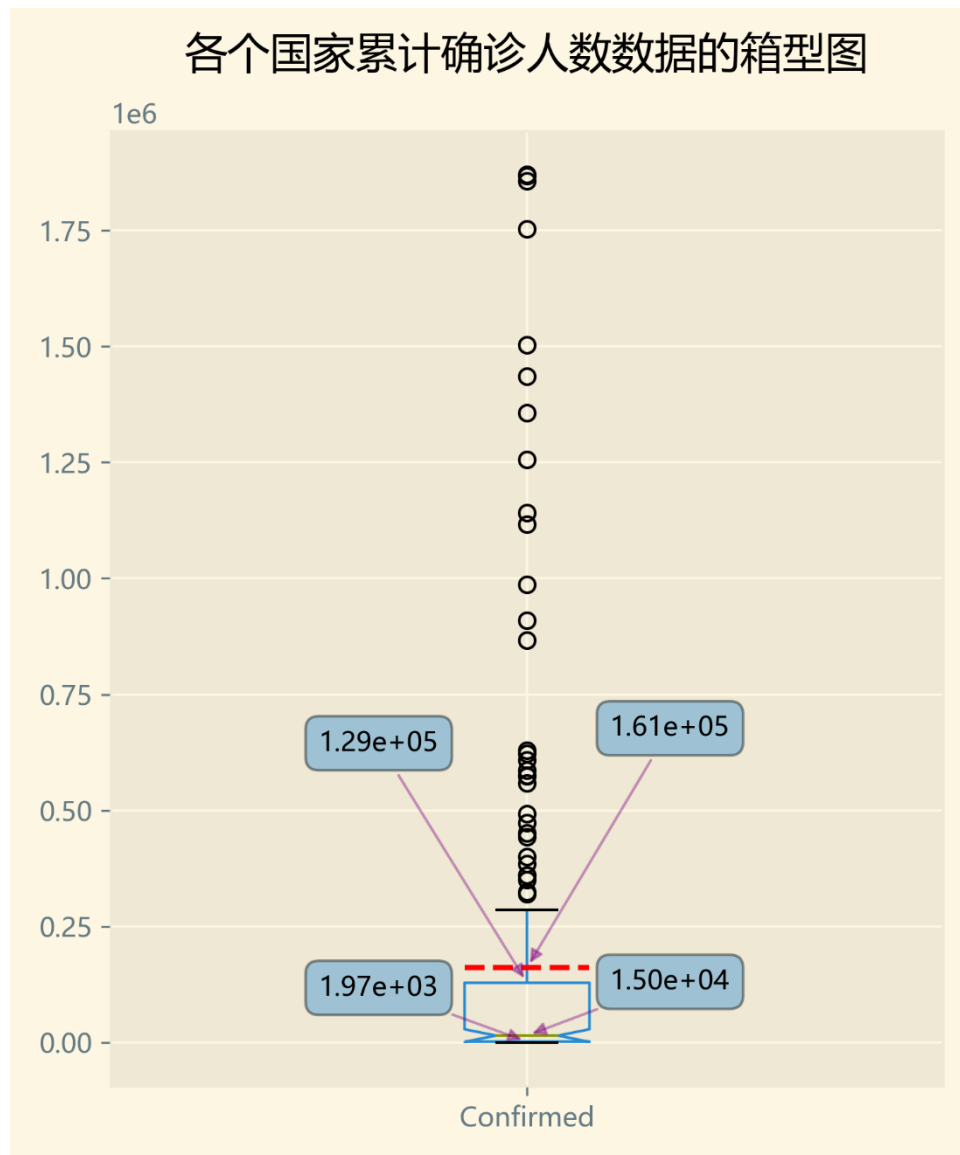


图 10 处理后的箱型图

此时箱型图的数据就相对明显了。为了方便观察，使用凹口的形式展现箱线图，均值用红色虚线标注。可以看出下限与下四分位数(1.97×10^3)十分接近（实际上下限为 1），中位数（ 1.50×10^4 ）和均值(1.61×10^5)偏差较大。这说明数据分布具有较高的偏态，偏态的形态表现为右偏，其实际意义是只有较少部分的国家疫情非常严重，但这些国家的疫情规模也往往远大于其它国家。

3.8 康复率最高的 10 个国家

为了后续分析的便利，额外分析各国新冠疫情存活率情况，得到康复率最高的 10 个国家数据，作图如下：

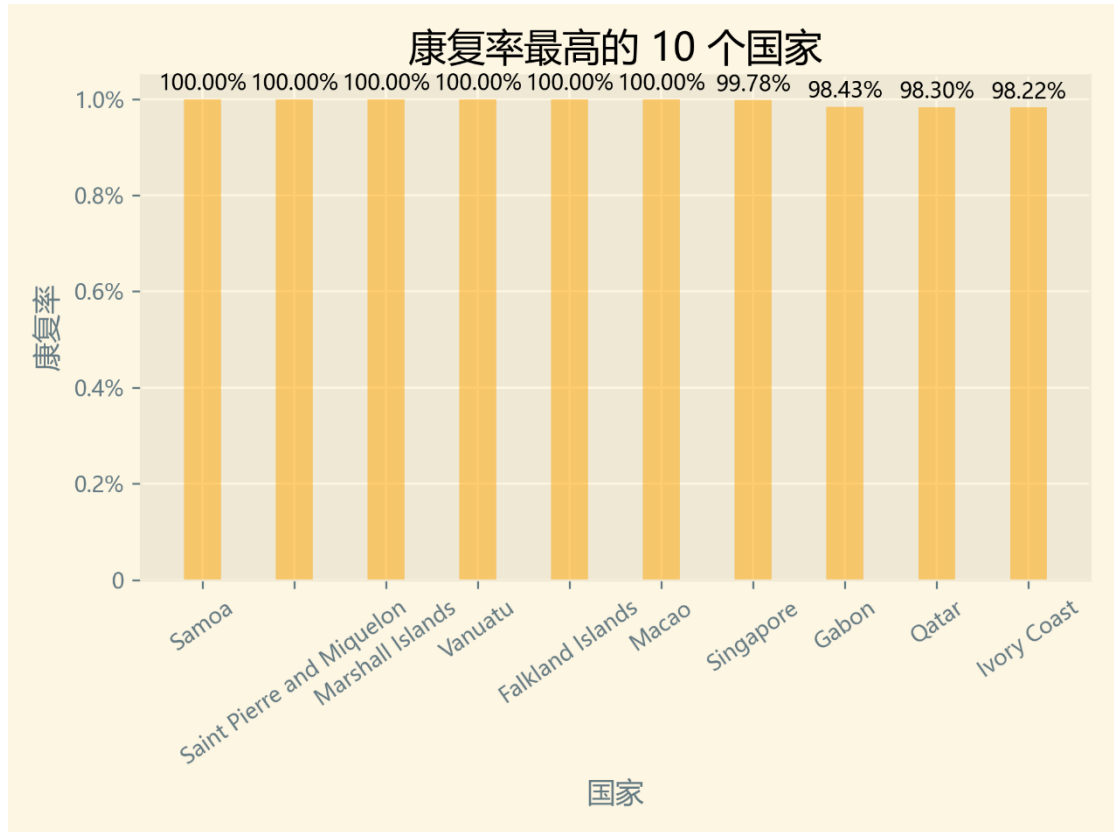


图 11 康复率最高的 10 个国家

可以看出，只有很少部分的国家实现了确诊人数完全康复，并且这些国家都是人口、面积较小的小国家。

4.分析全世界应对新冠疫情最好的 10 个国家

鉴于各个国家人口、医疗水平、检测人数等客观条件不同，故判断应对新冠疫情是否成功的关键因素及其权值如下：

1. 累计确诊人数占国家总人口比例，权值：0.25
2. 死亡率(死亡人数/确诊人数)，权值：0.3
3. 康复率(康复人数/确诊人数)，权值：0.3
4. 15 日内确诊人数日增长率((今日确诊人数-昨日确诊总人数)/昨日确诊总人数)平均值，权值：0.15

权值如此设置是因为在本模型中，新冠肺炎疫情对人的生命健康的影响是首要考虑因素，存活率和康复率是应对疫情是否成功的最关键指标。

累计确诊人数是第二重要的因素，因为这表明了某个国家疫情的规模，体现了该国家疫情的严重程度

日增长速率表明了新冠肺炎疫情在某个国家的肆虐速度，一定程度上能体现疫情在该国家的严重程度。但是，应当考虑部分国家由于应对措施得当导致潜在感染者被大量发现的情况，因此该指标的权值最小。

之后，应当设置一个判决值，其计算公式为：

$$\text{判决值} = (1 - \text{确诊率}) * 0.25 + (1 - \text{死亡率}) * 0.3 + \text{康复率} * 0.3 + (1 - \text{增长率}) * 0.15$$

判决值越高的国家，认为其应对新冠疫情做的最好。分析得到如下表格：

表格 1 判决值最优的 10 个国家

Name	Confirmed rate	Deceased rate	Recovered rate	Increase rate	Judgement Value
Singapore	0.009937	0.000497	0.997755	0.000138	0.996672
Ivory Coast	0.000814	0.006128	0.98217	0.001242	0.992423
Gabon	0.004157	0.006737	0.98428	0.001056	0.992065
Ghana	0.00169	0.006168	0.976365	0.001844	0.99036
Djibouti	0.005765	0.01064	0.980638	0.000676	0.989457
Equatorial Guinea	0.003643	0.016393	0.975506	0.000443	0.986757
Uzbekistan	0.002233	0.008134	0.963863	0.00207	0.98585
Comoros	0.000715	0.011146	0.964968	0.001972	0.985672
Madagascar	0.000628	0.014727	0.966168	0.001011	0.985124
Iceland	0.016257	0.005032	0.969087	0.002249	0.984815

5. 疫情预测

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，并与实际数据进行对比。在该部分，采用了三种预测方法进行预测分析，分别为

1. 霍尔特(Holt)线性趋势法：水平参数：1，趋势参数：0.2
2. 自回归移动平均模型 (ARIMA)：参数 p ， d ， q 分别为 2，1，7
3. 滑动窗口时间预测模型：窗口大小 2、3、4

选择霍尔特(Holt)线性趋势法是因为，累计确诊数数据没有季节性，但有递增趋势，该方法可以在无需假设的情况下，准确预测出数据趋势。

自回归移动平均模型 (ARIMA) 的目标是描述数据中彼此之间的关系，尽管常用来描述数据季节性特征，但同时也能处理具有趋势性的数据预测

滑动窗口模型则是经典的基于时间序列的预测方法。

利用前 10 日数据作为训练集，后 5 日作为测试集，得到如下分析图：

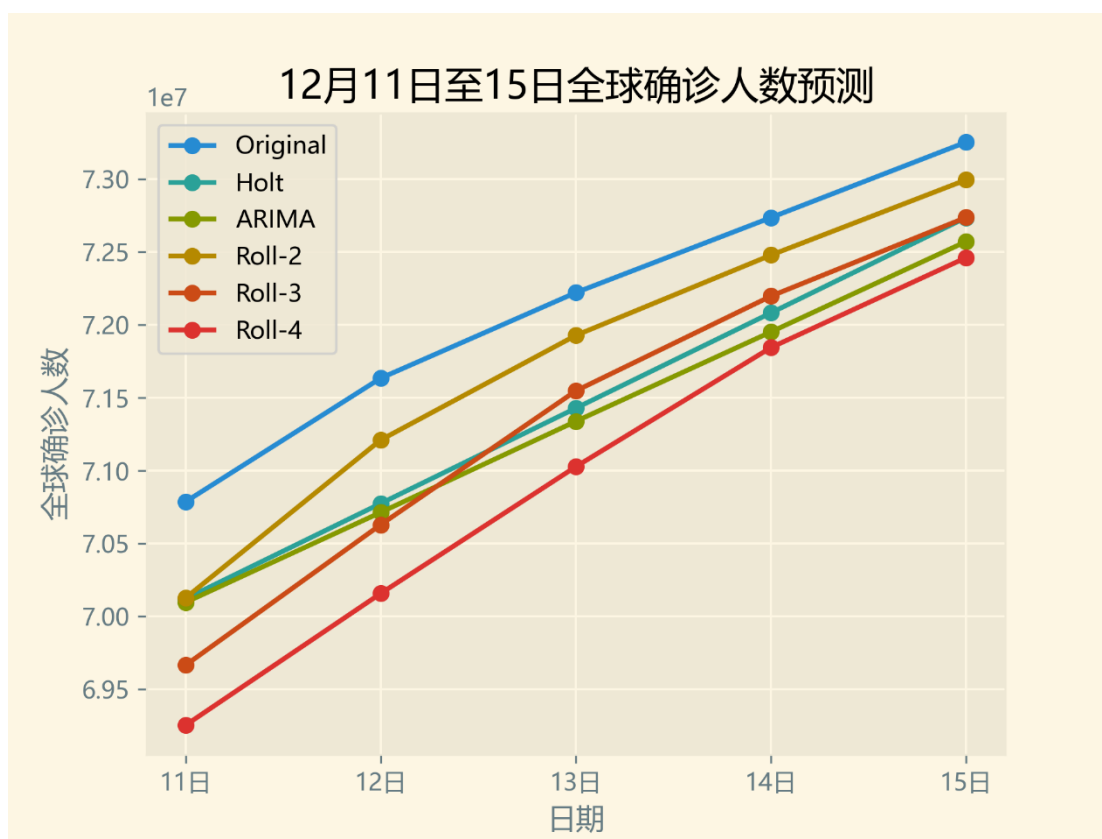


图 12 全球确诊人数预测

预测结果评价采用均方根误差，通过将预测结果和测试集数据进行分析计算，得到如下误差分析表

表格 2 均方根误差分析表

Method	Value
Holt	707708
ARIMA	797906
Roll-2	408442
Roll-3	809689
Roll-4	1214687

此处没有使用数据归一化的原因在于，由于数据间差距过大，在数据归一化后计算均方误差时将出现较为严重的精度问题，故不予考虑。

由表 2 及图 12 可知，滑动窗口数为 2 的时间预测方法拥有最好的预测效果，各个方案的详细预测数据如下表所示：

表格 3 详细预测数据

Confirmed	Holt_linear	ARIMA	Roll_2	Roll_3	Roll_4
63649292	63649292	63649292			
64327991	64327991	64327991	63988642		
64905069	64905069	64905069	64616530	64294117	
65744839	65744839	65744839	65324954	64992633	64656798
66294914	66294914	66294914	66019877	65648274	65318203
66936322	66936322	66936322	66615618	66325358	65970286
67493569	67493569	67493569	67214946	66908268	66617411
68017845	68017845	68017845	67755707	67482579	67185663
68741600	68741600	68741600	68379723	68084338	67797334
69466126	69466126	69466126	69103863	68741857	68429785
70785317	70119937	70095029	70125722	69664348	69252722
71633210	70773748	70712571	71209264	70628218	70156563
72220080	71427558	71337258	71926645	71546202	71026183
72734967	72081369	71950726	72477524	72196086	71843394
73252453	72735180	72570696	72993710	72735833	72460178

对于 Holt 和 ARIMA 预测方法来说，预测数据与实际数据存在差异的主要原因在于训练数据过少，并且没有很好地调整参数以适应新冠疫情的增长趋势。

对于基于时间序列的滑动窗口分析来说，其无法很好的应对疫情确诊数据的突发性，即当某些国家的确诊数据突然增大或是减少，该预测方案将不能很好的应对。然而由前文所述，12 月 1 日至 15 日的新冠疫情确诊数据是相对平稳的，因此滑动窗口预测的效果会好于其它两种预测方法。

6.源代码

6.1 爬虫

```
import csv
import time

import requests
from bs4 import BeautifulSoup

url = 'https://ncov2019.live/'

header = {
    'content-type': 'text/html;charset=UTF-8',
    'user-agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4280.66 Safari/537.36 Edg/87.0.664.41',
}

# 数据种类
data = {'Name': '', 'Confirmed': '', 'Confirmed Per Million': '', 'Confirmed Changes Today': '',
        'Confirmed Percentage Day Change': '', 'Critical': '', 'Deceased': '', 'Deceased Per Million': '',
        'Deceased Changes Today': '', 'Death Percentage Day Change': '', 'Tests': '', 'Active': '', 'Recovered': '',
        'Recovered Per Million': '', 'Population': ''}

# 配置输出的 csv 文件
csv_file = open('csvFile/Covid19Data' + time.strftime("%Y-%m-%d") + '.csv', "w", newline='', encoding="utf_8_sig")
csv_writer = csv.DictWriter(csv_file,
                            fieldnames=data.keys())

csv_writer.writeheader()

# 发送 get 请求, 得到 html 页面
r = requests.get(url=url, headers=header)

# 保存 html 文件以防数据丢失或错误
with open('htmlFile/page' + time.strftime("%Y-%m-%d-%H-%M") + '.html', 'w', encoding='utf-8') as f:
    f.write(r.text)

soup = BeautifulSoup(r.text, 'html5lib')
items = soup.find('table', id='sortable_table_world').find('tbody').find_all('tr')

for item in items:
    index = 0
    # 获得各种数据
```

```
for key in data.keys():
    s = item.select('td')[index].text
    # 处理特殊字符
    if '★' in s:
        s = ' '.join(s.split()[1:])
    s = s.strip()
    # 处理空数据
    if s == 'Unknown':
        s = 'NA'
    # 处理人口不足百万的国家
    if s == '0' and (
        key == 'Confirmed Per Million' or key == 'Deceased Per Million' or key == 'Recovered Per Million'):
        s = 'NA'
    data[key] = s
    index += 1
csv_writer.writerow(data)
print(data['Name'])

csv_file.close()
```

6.2 数据分析

6.2.1 Part-A

```
"""
15 天中，全球新冠疫情的总体变化趋势
"""

import matplotlib.pyplot as plt
import pandas as pd
from pandas import DataFrame

plt.rcParams['font.sans-serif'] = ['Microsoft YaHei'] # 用来正常显示中文标签
plt.rcParams['savefig.dpi'] = 300 # 图片像素
plt.rcParams['figure.dpi'] = 300 # 分辨率
plt.style.use('Solarize_Light2')

df = DataFrame()

for i in range(1, 16):
    if i >= 10:
        str_num = str(i)
```



```

else:
    str_num = '0' + str(i)

df[str(i)] = \
    pd.read_csv('csvFile/Covid19Data2020-12-' + str_num + '.csv',
                encoding='utf-8', thousands=',', nrows=1).loc[0] # 只读第一行的全球数据，并且去除千分位的逗号

print(df)
print(df.at['Confirmed', '1'])
print(df.loc['Confirmed'].to_list())

fig, ax = plt.subplots(nrows=2, ncols=3, figsize=(12, 8)) # nrows=2, ncols=2 figsize=(13, 13)

day_list = list(range(1, 16))

# 1.展示全球新冠疫情总确认数量变化
Confirmed_list = df.loc['Confirmed'].to_list()
ax[0, 0].plot(day_list, Confirmed_list, linewidth=2.0, marker='o')
ax[0, 0].text(day_list[0] - 1.5, Confirmed_list[0] + 3e5, '{:.2e}'.format(Confirmed_list[0]), color='r', size=13,
              weight='bold')
ax[0, 0].text(day_list[-1] - 1.5, Confirmed_list[-1] + 3e5, '{:.2e}'.format(Confirmed_list[-1]), color='r', size=13,
              weight='bold')
ax[0, 0].set_xticks(day_list)
ax[0, 0].set_xlabel('日期')
ax[0, 0].set_ylabel('确诊人数')
ax[0, 0].set_title('12月1日至15日全球新冠累计确诊人数变化', y=1.1, size=13)

# 2.展示全球新冠疫情增长速度变化
Confirmed_Percentage_list = df.loc['Confirmed Percentage Day Change'].to_list()
Confirmed_Percentage_list = [float(i[:-1]) for i in Confirmed_Percentage_list]
ax[0, 1].plot(day_list, Confirmed_Percentage_list, linewidth=2.0, marker='o')
ax[0, 1].set_xticks(day_list)
ax[0, 1].set_xlabel('日期')
ax[0, 1].set_yticks([0.10, 0.15, 0.20, 0.25, 0.30, 0.35])
ax[0, 1].set_yticklabels(['0.10%', '0.15%', '0.20%', '0.25%', '0.30%', '0.35%'])
ax[0, 1].set_ylabel('确诊人数日增长率')
ax[0, 1].set_title('12月1日至15日全球新冠总确诊人数增长率', y=1.1, size=13)

# 3.展示现存确诊人数
Active_list = df.loc['Active'].to_list()
ax[0, 2].plot(day_list, Active_list, linewidth=2.0, marker='o')
ax[0, 2].text(day_list[0] - 1.5, Active_list[0] + 8e4, '{:.2e}'.format(Active_list[0]), color='r', size=13,
              weight='bold')
ax[0, 2].text(day_list[-1] - 1.5, Active_list[-1] + 8e4, '{:.2e}'.format(Active_list[-1]), color='r', size=13,
              weight='bold')
ax[0, 2].set_xticks(day_list)
ax[0, 2].set_xlabel('日期')
ax[0, 2].set_ylabel('现存确诊人数')
ax[0, 2].set_title('12月1日至15日全球新冠现存确诊人数净变化', y=1.1, size=13)

```

```

# 4.展示全球新冠疫情死亡人数变化
Deceased_list = df.loc['Deceased'].to_list()
ax[1, 0].plot(day_list, Deceased_list, linewidth=2.0, marker='o')
ax[1, 0].text(day_list[0] - 1.5, Deceased_list[0] + 6e3, '{:.2e}'.format(Deceased_list[0]), color='r', size=13,
              weight='bold')
ax[1, 0].text(day_list[-1] - 1.5, Deceased_list[-1] + 6e3, '{:.2e}'.format(Deceased_list[-1]), color='r', size=13,
              weight='bold')
ax[1, 0].set_xticks(day_list)
ax[1, 0].set_xlabel('日期')
ax[1, 0].set_ylabel('死亡人数')
ax[1, 0].set_title('12月1日至15日全球新冠死亡人数变化', y=1.1, size=13)

# 5.展示全球新冠疫情康复人数变化
Recovered_list = df.loc['Recovered'].to_list()
ax[1, 1].plot(day_list, Recovered_list, linewidth=2.0, marker='o')
ax[1, 1].text(day_list[0] - 1.5, Recovered_list[0] + 3e5, '{:.2e}'.format(Recovered_list[0]), color='r', size=13,
              weight='bold')
ax[1, 1].text(day_list[-1] - 1.5, Recovered_list[-1] + 3e5, '{:.2e}'.format(Recovered_list[-1]), color='r', size=13,
              weight='bold')
ax[1, 1].set_xticks(day_list)
ax[1, 1].set_xlabel('日期')
ax[1, 1].set_ylabel('康复人数')
ax[1, 1].set_title('12月1日至15日全球新冠康复人数变化', y=1.1, size=13)

# 5.展示全球新冠疫情检测人数变化
Tests_list = df.loc['Tests'].to_list()
ax[1, 2].plot(day_list, Tests_list, linewidth=2.0, marker='o')
ax[1, 2].text(day_list[0] - 1.5, Tests_list[0] + 3e6, '{:.2e}'.format(Tests_list[0]), color='r', size=13,
              weight='bold')
ax[1, 2].text(day_list[-1] - 1.5, Tests_list[-1] + 3e6, '{:.2e}'.format(Tests_list[-1]), color='r', size=13,
              weight='bold')
ax[1, 2].set_xticks(day_list)
ax[1, 2].set_xlabel('日期')
ax[1, 2].set_ylabel('检测人数')
ax[1, 2].set_title('12月1日至15日全球新冠检测人数变化', y=1.1, size=13)

plt.tight_layout()
plt.subplots_adjust()
plt.savefig('imgResult/总体变化趋势.png')
plt.show()

```

6.2.2 Part-B

"""

15 天中，全球新冠疫情的总体变化趋势

```

"""

import matplotlib.pyplot as plt
import pandas as pd
from pandas import DataFrame

plt.rcParams['font.sans-serif'] = ['Microsoft YaHei'] # 用来正常显示中文标签
plt.rcParams['savefig.dpi'] = 300 # 图片像素
plt.rcParams['figure.dpi'] = 300 # 分辨率
plt.style.use('Solarize_Light2')

df = DataFrame()

for i in range(1, 16):
    if i >= 10:
        str_num = str(i)
    else:
        str_num = '0' + str(i)
    df[str(i)] = \
        pd.read_csv('csvFile/Covid19Data2020-12-' + str_num + '.csv',
                    encoding='utf-8', thousands=',', nrows=1).loc[0] # 只读第一行的全球数据，并且去除千分位的逗号

print(df)
print(df.at['Confirmed', '1'])
print(df.loc['Confirmed'].to_list())

fig, ax = plt.subplots(nrows=2, ncols=3, figsize=(12, 8)) # nrows=2, ncols=2 figsize=(13, 13)

day_list = list(range(1, 16))
# 1.展示全球新冠疫情总确认数量变化
Confirmed_list = df.loc['Confirmed'].to_list()
ax[0, 0].plot(day_list, Confirmed_list, linewidth=2.0, marker='o')
ax[0, 0].text(day_list[0] - 1.5, Confirmed_list[0] + 3e5, '{:.2e}'.format(Confirmed_list[0]), color='r', size=13,
              weight='bold')
ax[0, 0].text(day_list[-1] - 1.5, Confirmed_list[-1] + 3e5, '{:.2e}'.format(Confirmed_list[-1]), color='r', size=13,
              weight='bold')
ax[0, 0].set_xticks(day_list)
ax[0, 0].set_xlabel('日期')
ax[0, 0].set_ylabel('确诊人数')
ax[0, 0].set_title('12月1日至15日全球新冠累计确诊人数变化', y=1.1, size=13)
# 2.展示全球新冠疫情增长速度变化
Confirmed_Percentage_list = df.loc['Confirmed Percentage Day Change'].to_list()
Confirmed_Percentage_list = [float(i[:-1]) for i in Confirmed_Percentage_list]
ax[0, 1].plot(day_list, Confirmed_Percentage_list, linewidth=2.0, marker='o')
ax[0, 1].set_xticks(day_list)

```

```
ax[0, 1].set_xlabel('日期')
ax[0, 1].set_yticks([0.10, 0.15, 0.20, 0.25, 0.30, 0.35])
ax[0, 1].set_yticklabels(['0.10%', '0.15%', '0.20%', '0.25%', '0.30%', '0.35%'])
ax[0, 1].set_ylabel('确诊人数日增长率')
ax[0, 1].set_title('12月1日至15日全球新冠总确诊人数增长率', y=1.1, size=13)
# 3.展示现存确诊人数
Active_list = df.loc['Active'].to_list()
ax[0, 2].plot(day_list, Active_list, linewidth=2.0, marker='o')
ax[0, 2].text(day_list[0] - 1.5, Active_list[0] + 8e4, '{:.2e}'.format(Active_list[0]), color='r', size=13,
              weight='bold')
ax[0, 2].text(day_list[-1] - 1.5, Active_list[-1] + 8e4, '{:.2e}'.format(Active_list[-1]), color='r', size=13,
              weight='bold')
ax[0, 2].set_xticks(day_list)
ax[0, 2].set_xlabel('日期')
ax[0, 2].set_ylabel('现存确诊人数')
ax[0, 2].set_title('12月1日至15日全球新冠现存确诊人数净变化', y=1.1, size=13)
# 4.展示全球新冠疫情死亡人数变化
Deceased_list = df.loc['Deceased'].to_list()
ax[1, 0].plot(day_list, Deceased_list, linewidth=2.0, marker='o')
ax[1, 0].text(day_list[0] - 1.5, Deceased_list[0] + 6e3, '{:.2e}'.format(Deceased_list[0]), color='r', size=13,
              weight='bold')
ax[1, 0].text(day_list[-1] - 1.5, Deceased_list[-1] + 6e3, '{:.2e}'.format(Deceased_list[-1]), color='r', size=13,
              weight='bold')
ax[1, 0].set_xticks(day_list)
ax[1, 0].set_xlabel('日期')
ax[1, 0].set_ylabel('死亡人数')
ax[1, 0].set_title('12月1日至15日全球新冠死亡人数变化', y=1.1, size=13)
# 5.展示全球新冠疫情康复人数变化
Recovered_list = df.loc['Recovered'].to_list()
ax[1, 1].plot(day_list, Recovered_list, linewidth=2.0, marker='o')
ax[1, 1].text(day_list[0] - 1.5, Recovered_list[0] + 3e5, '{:.2e}'.format(Recovered_list[0]), color='r', size=13,
              weight='bold')
ax[1, 1].text(day_list[-1] - 1.5, Recovered_list[-1] + 3e5, '{:.2e}'.format(Recovered_list[-1]), color='r', size=13,
              weight='bold')
ax[1, 1].set_xticks(day_list)
ax[1, 1].set_xlabel('日期')
ax[1, 1].set_ylabel('康复人数')
ax[1, 1].set_title('12月1日至15日全球新冠康复人数变化', y=1.1, size=13)
# 5.展示全球新冠疫情检测人数变化
Tests_list = df.loc['Tests'].to_list()
ax[1, 2].plot(day_list, Tests_list, linewidth=2.0, marker='o')
ax[1, 2].text(day_list[0] - 1.5, Tests_list[0] + 3e6, '{:.2e}'.format(Tests_list[0]), color='r', size=13,
              weight='bold')
ax[1, 2].text(day_list[-1] - 1.5, Tests_list[-1] + 3e6, '{:.2e}'.format(Tests_list[-1]), color='r', size=13,
```

```

        weight='bold')
ax[1, 2].set_xticks(day_list)
ax[1, 2].set_xlabel('日期')
ax[1, 2].set_ylabel('检测人数')
ax[1, 2].set_title('12月1日至15日全球新冠检测人数变化', y=1.1, size=13)

plt.tight_layout()
plt.subplots_adjust()
plt.savefig('imgResult/总体变化趋势.png')
plt.show()

```

6.2.3 Part-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 matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from pandas import DataFrame

plt.rcParams['font.sans-serif'] = ['Microsoft YaHei'] # 用来正常显示中文标签
plt.rcParams['savefig.dpi'] = 300 # 图片像素
plt.rcParams['figure.dpi'] = 300 # 分辨率
plt.style.use('Solarize_Light2')

countries = pd.read_csv('csvFile/Covid19Data2020-12-15.csv', encoding='utf-8', skiprows=[1])['Name'].to_list()
df = DataFrame(columns=countries)

# 只读取每日确诊数据
for i in range(1, 16):
    if i >= 10:
        str_num = str(i)
    else:
        str_num = '0' + str(i)
    df_temp = pd.read_csv('csvFile/Covid19Data2020-12-' + str_num + '.csv',
                          encoding='utf-8', thousands=',', skiprows=[1], usecols=[0, 1])
    for tup in df_temp.itertuples():
        df.at[i, tup[1]] = tup[2]

df.to_csv('csvResult/所有国家15日确诊人数.csv', index=False)
for i in range(2, 16).__reversed__():

```

```

df.loc[i] -= df.loc[i - 1]

df.loc[str(i) + ' rate'] = df.loc[i] / df.loc[i - 1]

df = df.T

df['Name'] = np.array(countries)

df['Sum'] = df[df.columns[1:15]].sum(axis=1)

df['Rate Sum'] = df[df.columns[15:29]].mean(axis=1)

df.to_csv('csvResult/所有国家的新增确诊数数据日变化.csv', index=False)

df.sort_values(by='Sum', inplace=True, ascending=False)

df.reset_index(drop=True)

# 作图

# 取出累计确诊最多的 10 个国家

df_res = df[0:10]

fig, ax = plt.subplots()

day_list = list(range(2, 16))

for i in df_res.index.to_list():

    s = df_res.loc[i]

    ax.plot(day_list, s[[j for j in day_list]]) # 每个国家一条折线

ax.set_xticks(day_list)

ax.set_xlabel('日期')

ax.set_ylabel('新增确诊人数')

ax.set_title('12 月 1 日至 15 日每日新增确诊数累计排名前 10 的国家', size=13)

plt.legend(df_res['Name'].to_list(), loc='best') # 图例

plt.tight_layout()

plt.savefig('imgResult/新增确诊数最高的 10 个国家.png')

plt.show()

```

6.2.4 Part-D

"""

累计确诊人数占国家总人口比例最高的 10 个国家；

人口数量和累计确诊人数采用 12 月 15 日的数据

"""

```
import matplotlib.pyplot as plt
```

```
import pandas as pd
```

```
plt.rcParams['font.sans-serif'] = ['Microsoft YaHei'] # 用来正常显示中文标签
```

```
plt.rcParams['savefig.dpi'] = 300 # 图片像素
```

```
plt.rcParams['figure.dpi'] = 300 # 分辨率
```

```
plt.style.use('Solarize_Light2')
```

```
df = pd.read_csv('csvFile/Covid19Data2020-12-15.csv', encoding='utf-
```

```
8', skiprows=[1], thousands=',', usecols=[0, 1, 14])
```

```

df['Confirmed rate'] = df['Confirmed'] / df['Population']
print(df)

df.sort_values(by='Confirmed rate', inplace=True, ascending=False)

# 取出累计确诊人数占国家总人口比例最高的 10 个国家
df_res = df[0:10]
df_res = df_res.reset_index(drop=True) # 重置索引
print(df_res)

plt.bar(list(range(0, 50, 5)), df_res['Confirmed rate'].to_list(), width=2, alpha=0.5, color='r')
plt.xticks(list(range(0, 50, 5)), labels=df_res['Name'].to_list(), rotation=35)
plt.yticks([0.00, 0.02, 0.04, 0.06, 0.08, 0.10], ['0%', '2%', '4%', '6%', '8%', '10%'])
plt.tick_params(labelsize=11)
for a, b in zip(list(range(0, 50, 5)), df_res['Confirmed rate'].to_list()): # 在直方图上显示数字
    plt.text(a, b + 0.000001, '%.2f%' % (b * 100), ha='center', va='bottom', fontsize=10, color='black')
plt.title('累计确诊人数占国家总人口比例最高的 10 个国家')
plt.xlabel("国家")
plt.ylabel("确诊比例")

plt.tight_layout()
plt.savefig('imgResult/累计确诊人数占国家总人口比例最高的 10 个国家.png')
plt.show()

df_res.to_csv('csvResult/累计确诊人数占国家总人口比例最高的 10 个国家.csv', index=False)

```

6.2.5 Part-E

```

"""
死亡率（累计死亡人数/累计确诊人数）最低的 10 个国家：
"""

import matplotlib.pyplot as plt
import pandas as pd

plt.rcParams['font.sans-serif'] = ['Microsoft YaHei'] # 用来正常显示中文标签
plt.rcParams['savefig.dpi'] = 300 # 图片像素
plt.rcParams['figure.dpi'] = 300 # 分辨率
plt.style.use('Solarize_Light2')

df = pd.read_csv('csvFile/Covid19Data2020-12-15.csv', encoding='utf-8', skiprows=[1], thousands=',', usecols=[0, 1, 6])

df['Deceased rate'] = df['Deceased'] / df['Confirmed']
print(df)

```

```

df.sort_values(by='Deceased rate', inplace=True)

# 取出死亡率最低的 10 个国家
df_res = df[0:10]

df_res = df_res.reset_index(drop=True) # 重置索引
print(df_res)

plt.bar(list(range(0, 50, 5)), df_res['Deceased rate'].to_list(), width=2, alpha=0.5, color='y')
plt.xticks(list(range(0, 50, 5)), labels=df_res['Name'].to_list(), rotation=35)
plt.yticks([0.000, 0.001, 0.002, 0.003, 0.004, 0.005], ['0', '0.1%', '0.2%', '0.3%', '0.4%', '0.5%'])
plt.tick_params(labelsize=9)
for a, b in zip(list(range(0, 50, 5)), df_res['Deceased rate'].to_list()): # 在直方图上显示数字
    plt.text(a, b + 0.000001, '%.2f%%' % (b * 100), ha='center', va='bottom', fontsize=10, color='black')
plt.title('死亡率最低的 10 个国家')
plt.xlabel("国家")
plt.ylabel("死亡率")

plt.tight_layout()
plt.savefig('imgResult/死亡率最低的 10 个国家.png')
plt.show()

df_res.to_csv('csvResult/死亡率最低的 10 个国家.csv', index=False)

```

6.2.6 Part-F

```

"""
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）：
"""

import matplotlib.pyplot as plt
import pandas as pd

plt.rcParams['font.sans-serif'] = ['Microsoft YaHei'] # 用来正常显示中文标签
plt.rcParams['savefig.dpi'] = 300 # 图片像素
plt.rcParams['figure.dpi'] = 300 # 分辨率
plt.style.use('Solarize_Light2')

df = pd.read_csv('csvFile/Covid19Data2020-12-15.csv', encoding='utf-8', skiprows=[1], thousands=',', usecols=[0, 1])
df.sort_values(by='Confirmed', inplace=True, ascending=False)

df = df.reset_index(drop=True) # 重置索引

# print(df)
# print(df.info())
# print(df.head(20))

# 展示确诊人数大于一百万的国家，其它国家归为其它

```



```

df_show = df[df['Confirmed'] > 1000000]
df_other = df[df['Confirmed'] < 1000000]

new = pd.DataFrame({'Name': 'Others', 'Confirmed': df_other['Confirmed'].sum()}, index=[1])
df_show = df_show.append(new, ignore_index=True)
print(df_show)

patches, l_text, p_text = plt.pie(df_show['Confirmed'], startangle=300, labels=df_show['Name'],
                                   autopct='%1.2f%%', labeldistance=1.1, textprops={'fontsize': 10, 'color': 'black'})

plt.title('世界 219 个国家的累计确诊人数比例图', y=1.05)
plt.axis('equal')
# 图例
plt.legend(loc='upper right', fontsize=7)

plt.tight_layout()
plt.savefig('imgResult/世界 219 个国家的累计确诊人数比例图.png')
plt.show()
df_show.to_csv('csvResult/世界 219 个国家的累计确诊人数展示数据.csv', index=False)

```

6.2.7 Part-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 matplotlib.pyplot as plt
import pandas as pd
from sklearn.preprocessing import MinMaxScaler

plt.rcParams['font.sans-serif'] = ['Microsoft YaHei'] # 用来正常显示中文标签
plt.rcParams['savefig.dpi'] = 300 # 图片像素
plt.rcParams['figure.dpi'] = 300 # 分辨率
plt.style.use('Solarize_Light2')
plt.figure(figsize=(5, 6))
scaler = MinMaxScaler()

df = pd.read_csv('csvFile/Covid19Data2020-12-15.csv', encoding='utf-8', skiprows=[1], thousands=',', usecols=[0, 1])

# 数据右偏，去除最大的五个异常数据
for i in range(5):
    df = df[df['Confirmed'] != df['Confirmed'].max()]

print(df)

```

```
print(df.describe())

# 归一化
# x_reshape = df["Confirmed"].values.reshape(-1, 1)
# Confirmed = scaler.fit_transform(x_reshape) # 调用MinMaxScaler的fit_transform转换方法
# df['Fitted Confirmed'] = pd.DataFrame(Confirmed)
# print(df)

# 突出展示均值线
f = df.boxplot(column=['Confirmed'], meanline=True, showmeans=True, vert=True, notch=True,
               return_type='dict', grid=True)

for mean in f['means']:
    mean.set(color='r', linewidth=2)

# plt.text(1.1, df['Confirmed'].mean(), '{:.2e}'.format(df['Confirmed'].mean()))
# plt.text(1.1, df['Confirmed'].median(), '{:.2e}'.format(df['Confirmed'].median()))
# plt.text(0.75, df['Confirmed'].quantile(0.25), '{:.2e}'.format(df['Confirmed'].quantile(0.25)))
# plt.text(0.75, df['Confirmed'].quantile(0.75), '{:.2e}'.format(df['Confirmed'].quantile(0.75)))

plt.annotate(text='{:.2e}'.format(df['Confirmed'].mean()), xy=(1, df['Confirmed'].mean()),
            xytext=(1.1, df['Confirmed'].mean() + 5e5),
            arrowprops=dict(arrowstyle='->', connectionstyle='arc3', color='purple', alpha=0.4),
            bbox=dict(boxstyle='round,pad=0.5', ec='k', lw=1, alpha=0.4))
plt.annotate(text='{:.2e}'.format(df['Confirmed'].median()), xy=(1, df['Confirmed'].median()),
            xytext=(1.1, df['Confirmed'].median() + 1e5),
            arrowprops=dict(arrowstyle='->', connectionstyle='arc3', color='purple', alpha=0.4),
            bbox=dict(boxstyle='round,pad=0.5', ec='k', lw=1, alpha=0.4))
plt.annotate(text='{:.2e}'.format(df['Confirmed'].quantile(0.25)), xy=(1, df['Confirmed'].quantile(0.25)),
            xytext=(0.75, df['Confirmed'].quantile(0.25) + 1e5),
            arrowprops=dict(arrowstyle='->', connectionstyle='arc3', color='purple', alpha=0.4),
            bbox=dict(boxstyle='round,pad=0.5', ec='k', lw=1, alpha=0.4))
plt.annotate(text='{:.2e}'.format(df['Confirmed'].quantile(0.75)), xy=(1, df['Confirmed'].quantile(0.75)),
            xytext=(0.75, df['Confirmed'].quantile(0.75) + 5e5),
            arrowprops=dict(arrowstyle='->', connectionstyle='arc3', color='purple', alpha=0.4),
            bbox=dict(boxstyle='round,pad=0.5', ec='k', lw=1, alpha=0.4))

plt.title('各个国家累计确诊人数数据的箱型图', y=1.05)
plt.tight_layout()
# plt.savefig('imgResult/全球各个国家累计确诊人数的箱型图.png')
plt.savefig('imgResult/全球各个国家累计确诊人数的箱型图（去除最大5个异常数据后）.png')
plt.show()
```

6.2.8 Extra

```

"""
康复率(康复人数/确诊人数)最高的 10 个国家:
"""

import matplotlib.pyplot as plt
import pandas as pd

plt.rcParams['font.sans-serif'] = ['Microsoft YaHei'] # 用来正常显示中文标签
plt.rcParams['savefig.dpi'] = 300 # 图片像素
plt.rcParams['figure.dpi'] = 300 # 分辨率
plt.style.use('Solarize_Light2')

df = pd.read_csv('csvFile/Covid19Data2020-12-15.csv', encoding='utf-8', skiprows=[1], thousands=',', usecols=[0, 1, 12])

df['Recovered rate'] = df['Recovered'] / df['Confirmed']
print(df)
df.sort_values(by='Recovered rate', inplace=True, ascending=False)

# 取出死亡率最低的 10 个国家
df_res = df[0:10]
df_res = df_res.reset_index(drop=True) # 重置索引
print(df_res)

plt.bar(list(range(0, 50, 5)), df_res['Recovered rate'].to_list(), width=2, alpha=0.5, color='orange')
plt.xticks(list(range(0, 50, 5)), labels=df_res['Name'].to_list(), rotation=35)
plt.yticks([0.0, 0.2, 0.4, 0.6, 0.8, 1.0], ['0', '0.2%', '0.4%', '0.6%', '0.8%', '1.0%'])
plt.tick_params(labelsize=9)
for a, b in zip(list(range(0, 50, 5)), df_res['Recovered rate'].to_list()): # 在直方图上显示数字
    plt.text(a, b + 0.008, '%.2f%%' % (b * 100), ha='center', va='bottom', fontsize=9, color='black')
plt.title('康复率最高的 10 个国家')
plt.xlabel("国家")
plt.ylabel("康复率")

plt.tight_layout()
plt.savefig('imgResult/康复率最高的 10 个国家.png')
plt.show()
df_res.to_csv('csvResult/康复率最高的 10 个国家.csv', index=False)

```

6.3 最佳的 10 个国家

```

"""
列出全世界应对新冠疫情最好的 10 个国家
"""

import numpy as np
import pandas as pd

df = pd.read_csv('csvFile/Covid19Data2020-12-15.csv', encoding='utf-8', skiprows=[1], thousands=',')

df['Confirmed rate'] = df['Confirmed'] / df['Population']
df['Deceased rate'] = df['Deceased'] / df['Confirmed']
df['Recovered rate'] = df['Recovered'] / df['Confirmed']
df['Increase rate'] = pd.read_csv('csvResult/所有国家的新增确诊数数据日变化.csv', encoding='utf-8', usecols=[31])
print(df)

# 有空数据的国家不予考虑
df.drop(df[np.isnan(df['Confirmed rate'])].index, inplace=True)
df.drop(df[np.isnan(df['Deceased rate'])].index, inplace=True)
df.drop(df[np.isnan(df['Recovered rate'])].index, inplace=True)
df.drop(df[np.isnan(df['Increase rate'])].index, inplace=True)

# 判决值计算公式：
# (1-确诊率)*0.25+(1-死亡率)*0.3+康复率*0.3+(1-增长率)*0.15
df['Judgement Value'] = (1 - df['Confirmed rate']) * 0.25 + (1 - df['Deceased rate']) * 0.3 + df[
    'Recovered rate'] * 0.3 + (1 - df['Increase rate']) * 0.15

df.drop(labels=df.columns[1:15], axis=1, inplace=True)
df.sort_values(by='Judgement Value', inplace=True, ascending=False)
df.to_csv('csvResult/总体分析.csv', index=False)

```

6.4 预测分析

```

"""
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，并与实际数据进行对比。说明你预测的方法，并分析与实际数据的差距和原因。
"""

from math import sqrt

import numpy as np
import pandas as pd

import statsmodels.api as sm
from matplotlib import pyplot as plt
from sklearn.metrics import mean_squared_error

```

```
from sklearn.preprocessing import MinMaxScaler
from statsmodels.tsa.holtwinters import Holt

scaler = MinMaxScaler()

plt.rcParams['font.sans-serif'] = ['Microsoft YaHei'] # 用来正常显示中文标签
plt.rcParams['savefig.dpi'] = 300 # 图片像素
plt.rcParams['figure.dpi'] = 300 # 分辨率
plt.style.use('Solarize_Light2')

df = pd.read_csv('csvFile/Covid19Data2020-12-01.csv', encoding='utf-8', thousands=',', nrows=1, usecols=[0, 1])
# 均方根误差
df_error = pd.DataFrame({'Method': ['Holt', 'ARIMA', 'Roll-2', 'Roll-3', 'Roll-4'], 'Value': [0, 0, 0, 0, 0]})
print(df_error)
# 只读取全球总体数据
for i in range(2, 16):
    if i >= 10:
        str_num = str(i)
    else:
        str_num = '0' + str(i)
    df_temp = pd.read_csv('csvFile/Covid19Data2020-12-' + str_num + '.csv',
                          encoding='utf-8', thousands=',', nrows=1, usecols=[0, 1])
    df.loc[i - 1] = df_temp.loc[0]
# 数据归一化后计算均方误差时将出现精度问题, 故不予考虑
# # 将 Confirmed 列归一化
# x_reshape = df["Confirmed"].values.reshape(-1, 1)
# df["Confirmed"] = scaler.fit_transform(x_reshape) # 调用 MinMaxScaler 的 fit_transform 转换方法
print(df)

fig, ax = plt.subplots()

# 1. 原始数据
ax.plot(np.arange(5), df.loc[10:14, 'Confirmed'], marker='o')

# 2. 霍尔特(Holt)线性趋势法
df.loc[0:9, 'Holt_linear'] = df.loc[0:9, 'Confirmed']
fit = Holt(np.asarray(df.loc[0:9, 'Confirmed'])).fit(smoothing_level=1, smoothing_trend=0.2)
df.loc[10:14, 'Holt_linear'] = fit.forecast(5)
ax.plot(np.arange(5), df.loc[10:14, 'Holt_linear'], marker='o')
df_error.at[0, 'Value'] = sqrt(mean_squared_error(df.loc[10:14, 'Confirmed'], df.loc[10:14, 'Holt_linear']))

# 3. 自回归移动平均模型 (ARIMA)
df.loc[0:9, 'ARIMA'] = df.loc[0:9, 'Confirmed']
fit1 = sm.tsa.statespace.SARIMAX(df.loc[0:9, 'Confirmed'], order=(2, 1, 7)).fit()
```

```
df.loc[10:14, 'ARIMA'] = fit1.predict(start=10, end=14, dynamic=True)
ax.plot(np.arange(5), df.loc[10:14, 'ARIMA'], marker='o')
df_error.at[1, 'Value'] = sqrt(mean_squared_error(df.loc[10:14, 'Confirmed'], df.loc[10:14, 'ARIMA']))

# 4.生成滑动窗口为2的预测值
df['Roll_2'] = df['Confirmed'].rolling(window=2, center=False).mean()
ax.plot(np.arange(5), df.loc[10:14, 'Roll_2'], marker='o')
df_error.at[2, 'Value'] = sqrt(mean_squared_error(df.loc[10:14, 'Confirmed'], df.loc[10:14, 'Roll_2']))

# 5.生成滑动窗口为3的预测值
df['Roll_3'] = df['Confirmed'].rolling(window=3, center=False).mean()
ax.plot(np.arange(5), df.loc[10:14, 'Roll_3'], marker='o')
df_error.at[3, 'Value'] = sqrt(mean_squared_error(df.loc[10:14, 'Confirmed'], df.loc[10:14, 'Roll_3']))

# 6.生成滑动窗口为4的预测值
df['Roll_4'] = df['Confirmed'].rolling(window=4, center=False).mean()
ax.plot(np.arange(5), df.loc[10:14, 'Roll_4'], marker='o')
df_error.at[4, 'Value'] = sqrt(mean_squared_error(df.loc[10:14, 'Confirmed'], df.loc[10:14, 'Roll_4']))

ax.legend(['Original', 'Holt', 'ARIMA', 'Roll-2', 'Roll-3', 'Roll-4'])
ax.set_xticks([0, 1, 2, 3, 4])
ax.set_xticklabels(['11日', '12日', '13日', '14日', '15日'])
ax.set_xlabel('日期')
ax.set_ylabel('全球确诊人数')
ax.set_title('12月11日至15日全球确诊人数预测')

df.to_csv('csvResult/预测分析.csv', index=False)
df_error.to_csv('csvResult/预测误差.csv', index=False)

plt.savefig('imgResult/12月11日至15日全球确诊人数预测.png')
plt.show()
```