北京郵電大學

Beijing University of Posts and Telecommunications

Python 程序设计

实验报告

——分析新冠疫情数据

姓 名：崔 思 颖

学 号：2018211290

学 院：计算机学院

班 级：2018211305

数据爬取

疫情数据

本次实验中的疫情数据来源于搜狗的“全国疫情地图：新冠肺炎疫情实时追踪”网站。

◆ 网址：

https://sa.sogou.com/new-weball/page/sgs/epidemic?type_page=VR

◆ 网页截图：

◆ 爬虫程序源码：

```python
from selenium import webdriver
import time
import pandas as pd
import datetime

'''''根据日期给 csv 文件命名'''
today = datetime.date.today()
fileNameStr = 'epidemic' + str(today) + '.csv'

'''定义爬虫类'''
class mySpider(object):
    '''''设置爬取网址'''
    def __init__(self):
        self.url = "https://sa.sogou.com/new-weball/page/sgs/epidemic?type_page=VR"
        self.browser = webdriver.Edge("E:\CODE\msedgedriver.exe")

    '''转到要爬取的网页'''
    def get_html(self):
        self.browser.get(self.url)
        self.browser.find_element_by_xpath(
            '//*[@id="async"]/div/div[1]/div[3]/div[1]/div/div[1]/a[2]').click()  # 点击“国外疫情”按钮
        self.browser.find_element_by_xpath('//*[@id="tab-pane-column_0"]/div[3]/a').click()  # 点击“查看更多”按钮
        time.sleep(3)

    '''单独爬取中国数据'''
    def China(self, file):
        self.browser.execute_script("arguments[0].click();", self.browser.find_element_by_xpath(
            '//*[@id="async"]/div/div[1]/div[3]/div[1]/div/div[1]/a[1]'))  # 点击“国内疫情”按钮
        time.sleep(3)
        file.write('中国,')
        file.write(self.browser.find_element_by_xpath(
            '//*[@id="async"]/div/div[1]/div[3]/div[1]/div/ul/li[1]/span[1]/strong/span').text)
        file.write(',')
        file.write(self.browser.find_element_by_xpath('//*[@id="async"]/div/div[1]/div[3]/div[1]/div/ul/li[1]/em').text)
        file.write(',')
        file.write(self.browser.find_element_by_xpath('//*[@id="async"]/div/div[1]/div[3]/div[1]/div/ul/li[2]/em').text)
```

```
37.         file.write(',')
38.         file.write(self.browser.find_element_by_xpath('//*[@id="async"]/div/
    div[1]/div[3]/div[1]/div/ul/li[4]/em').text)
39.         file.write('\n')
40.
41.     def parse(self):
42.         item = {}
43.         result = []
44.         file = open(fileNameStr, mode='w', encoding='utf-8')
45.
46.         '''爬取国外数据所有条目'''
47.         li = self.browser.find_elements_by_xpath('//*[@id="overseasChart"]/d
    iv')
48.         for each in li:
49.             result_item = each.text.split(',')
50.             info_list = each.text.split('\n')
51.             item['country'] = info_list[0]
52.             item['new'] = info_list[1]
53.             item['total'] = info_list[2]
54.             item['cured'] = info_list[3]
55.             item['death'] = info_list[4]
56.             # print(item)
57.             result.append(result_item)
58.         for items in result:
59.             file.write("".join(items).replace('\n', ','))
60.             file.write('\n')
61.
62.         '''爬取中国数据'''
63.         self.China(file)
64.
65.         file.close
66.
67.     def main(self):
68.         self.get_html()
69.         self.parse()
70.
71.
72. if __name__ == '__main__':
73.     spider = mySpider()
74.     spider.main()
```

人口数据

本次实验中的人口数据来源于百度的“世界人口排名_百度百科”网站。

◆ 网址

https://baike.baidu.com/item/%E4%B8%96%E7%95%8C%E5%90%84%E5%9B%BD%E4%BA%BA%E5%8F%A3%E6%8E%92%E5%90%8D/23333799

◆ 网页截图

◆ 爬虫程序源码

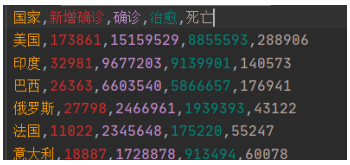
```
from selenium import webdriver
import pandas as pd
import time

fileNameStr = 'population.csv'

'''定义爬虫类'''
```

```
class mySpider(object):
    '''设置爬取网址'''
    def __init__(self):
        self.url = "https://baike.baidu.com/item/%E4%B8%96%E7%95%8C%E5%90%84%E5%9B%BD%E4%BA%BA%E5%8F%A3%E6%8E%92%E5%90%8D/23333799"
        self.browser=webdriver.Edge("E:\CODE\msedgedriver.exe")

    '''打开网页'''
    def get_html(self):
        self.browser.get(self.url)
        time.sleep(3)

    '''爬取'''
    def parse(self):
        item={}
        result=[]
        file = open(fileNameStr, mode='w', encoding='utf-8')
        flag=0;

        '''找到人口数据栏'''
        li=self.browser.find_elements_by_xpath('/html/body/div[4]/div[2]/div/div[2]/table/tbody/tr')
        for each in li:
            flag+=1
            if flag==1:
                continue    #表头不读取
            info_list=each.text.split('\n')

            '''单独处理中国一项，将大陆及港澳台人口合并'''
            if info_list[1]=='中华人民共和国':
                item['country'] = '中国'
                item['population'] = int(info_list[3].replace(',',''))
            elif info_list[0]=='中国香港' or info_list[0]=='中国澳门' or info_list[0]=='中国台湾':
                item['country'] = '中国'
                item['population'] += int(info_list[1].replace(',', ''))
            else:
                '''其余国家正常读取'''
                item['country'] = info_list[1]
                item['population'] = int(info_list[2].replace(',', ''))
            print(item)
            result.append(str(item['country'])+','+str(item['population']))

        '''写入文件'''
```

```
        for items in result:
            file.write(items)
            file.write('\n')
        file.close

    def main(self):
        self.get_html()
        self.parse()

if __name__ == '__main__':
    spider=mySpider()
    spider.main()
```

数据处理

处理 12-1 至 12-15 如图所示的数据




```
import numpy as np
import pandas as pd
import time

#打开 CSV 文件
for i in range(1,16):
    fileNameStr = 'epidemic2020-12-'
    if i<10:
        fileNameStr+='0'+str(i)+'.csv'
    else:
        fileNameStr+=str(i)+'.csv'
    df = pd.read_csv(fileNameStr, encoding='utf-8',dtype=np.str)

    #2.查看数据集的基本情况
    print("2:head=============================================================
==================")
    print(df.head())
    print("2:describe=========================================================
======================")
    print(df.describe())
```

```
    print("3:info=============================================================
    ==================")
    print(df.info())

    #3.查看是否有缺失值
    print("4=================================================================
    ============")
    print(df.isnull().sum().sort_values(ascending=False))

    #将'-'换成'0'
    for i in df.columns:
        df[i]=df[i].str.replace('-','0')

    #将"确诊"变为整型
    df['确诊'] = df['确诊'].astype(np.float).astype(np.int)

    #按照累计确诊倒序排序
    df=df.sort_values(by='确诊',ascending=False)

    #输出到文件
    df.to_csv('result1/'+fileNameStr, encoding="utf-8",index=False)
```

使用 12-15 的数据，添加确诊率、死亡率、治愈率三列，输出为“total.csv”

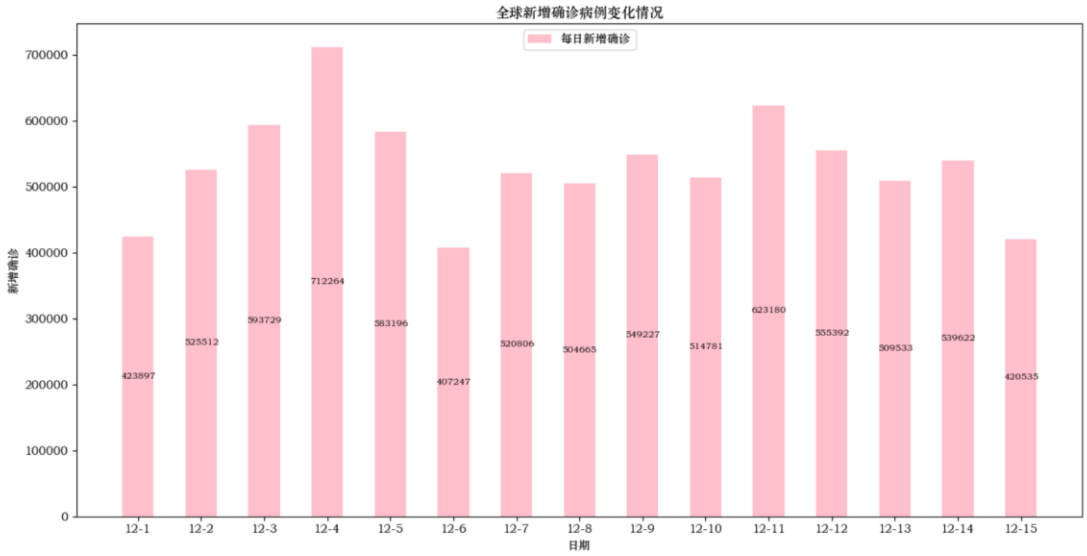
```
import pandas as pd

#处理 12-15 号数据
epdf=pd.read_csv('result1/epidemic2020-12-15.csv', encoding='utf-8')
podf=pd.read_csv('population.csv', encoding='utf-8')

#与人口表左连接
result=pd.merge(epdf, podf, how='left', on='国家
',copy=False, indicator=False,left_index=False)
temp=pd.merge(podf,epdf, how='left', on='国家
',copy=False, indicator=False,left_index=False)
print('======================疫情表空值======================')
print(result[result['人口'].isnull().values==True])
print('======================人口表空值======================')
print(temp[temp['确诊'].isnull().values==True])
result=result.dropna(how = 'any')

#添加”确诊率“、”死亡率“、“治愈率”三列
ratio_di=round(100*result['确诊']/result['人口'],4)
ratio_de=round(100*result['死亡']/result['确诊'],4)
```

```
ratio_cu=round(100*result['治愈']/result['确诊'],4)
result = pd.concat([result['国家'],result['确诊'],result['治愈'],result['死亡'],result['人口'],ratio_de,ratio_di,ratio_cu],axis=1,ignore_index=True)
result.columns = ['国家','确诊','治愈','死亡','人口','死亡率（%）','确诊率（%）','治愈率（%）']

#输出到 csv 文件
result.to_csv('total.csv',encoding='utf-8',index=False)
```

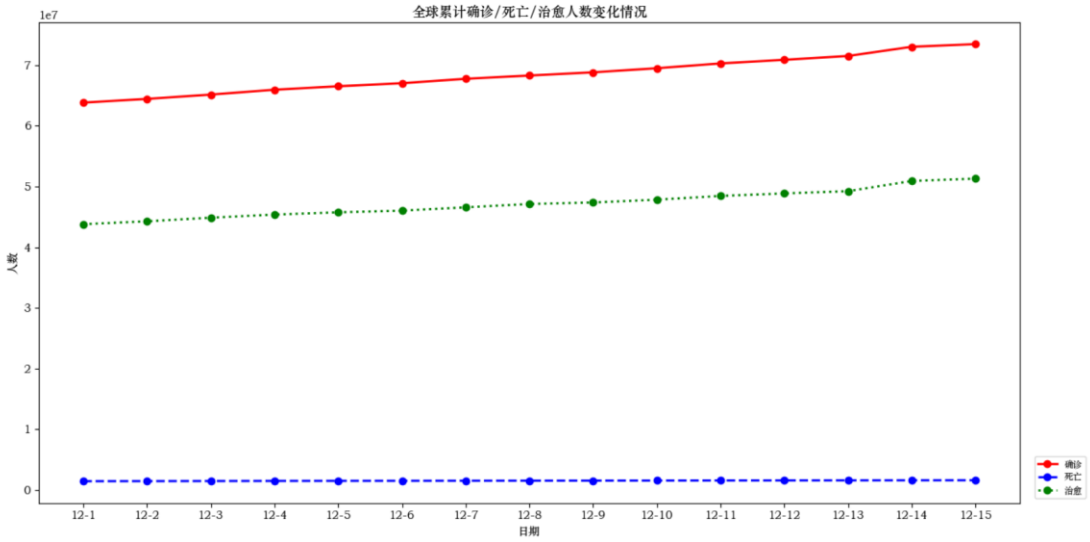
数据展示

◆ 15 天中，全球新冠疫情的总体变化趋势

15 天中，全球每日新增病例数变化不定，大多数处于 50 万人左右，



15 天中，累计确诊病例数稳步上升，累计治愈人数上升较为缓慢，而死亡人数变化不大。



由此可见，各国疫情防控措施有效地控制了死亡人数，治愈人数也在增加，但累计确诊人数依然明显呈逐日递增趋势，疫情形势较为严峻。

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from pylab import mpl
mpl.rcParams['font.sans-serif'] = ['STZhongsong']     # 指定默认字体：解决 plot 不能显示中文问题
mpl.rcParams['axes.unicode_minus'] = False            # 解决保存图像是负号'-'显示为方块的问题

newcase=[]
case=[]
death=[]
cured=[]

'得到每个 csv 中每一天的各个国家各种数值总计'
def readData():
    '''依次读取 csv 文件'''
    for i in range(1,16):
        fileNameStr = 'result1/epidemic2020-12-'
        if i<10:
            fileNameStr+='0'+str(i)+'.csv'
        else:
            fileNameStr+=str(i)+'.csv'
        df = pd.read_csv(fileNameStr, encoding='utf-8',dtype=np.str)

```

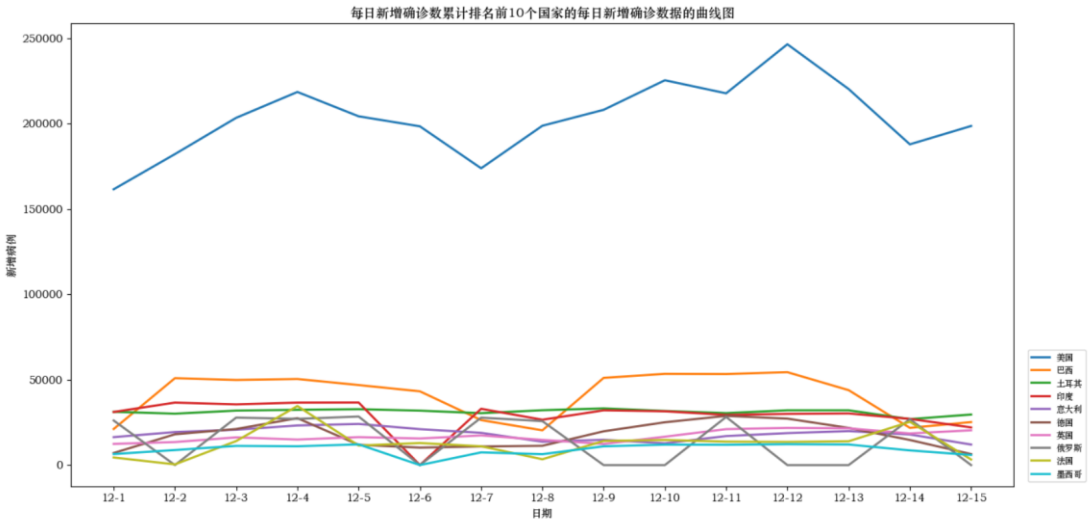
```
        newcase.append(df['新增确诊
    '].astype(np.float).astype(np.int).sum())
        case.append(df['确诊'].astype(np.float).astype(np.int).sum())
        death.append(df['死亡'].astype(np.float).astype(np.int).sum())
        cured.append(df['治愈'].astype(np.float).astype(np.int).sum())

readData()
#绘制每日新增条形图
plt.style.use('seaborn-muted')  # 设置图像风格
fig,ax = plt.subplots()
ax.set_title("全球新增确诊病例变化情况")
ax.set_xlabel("日期")
ax.set_ylabel("新增确诊")
x = np.array(['12-'+str(one) for one in range(1,16)])  # 创建一个numpy数组x
y = np.array(newcase) # 创建一个numpy数组y，内容为x中数据的平方值
plt.bar(x, y, color='pink',width=0.5)  # bar的颜色改为黄色
for a, b in zip(x, y):  # 在直方图上显示数字
    plt.text(a, b / 2, '%d' % b, ha='center', va='center', fontsize=8)
plt.legend(["每日新增确诊"],loc='upper center')
plt.show()

#绘制确诊、死亡、治愈折线图
plt.plot(x, case, 'bo',color="red", linewidth=2.0, linestyle="-", label='确诊
    ')
plt.plot(x, death, 'bo', color="b", linewidth=2.0, linestyle="--",label='死亡
    ')
plt.plot(x, cured, 'bo', color="g", linewidth=2.0, linestyle=":",label='治愈
    ')
plt.title('全球累计确诊/死亡/治愈人数变化情况')
plt.xlabel('日期')
plt.ylabel('人数')
plt.legend(["确诊","死亡","治愈
    "],bbox_to_anchor=(1.01, 0), loc=3, prop={'size': 8})
plt.show()
```

◆ 累计确诊数排名前 20 的国家名称及其数量

从数据处理生成的“total.csv”文件中，可以看到累计确诊数排名前 20 的国家和确诊数量如下：

```
国家,确诊,治愈,死亡,
美国,16942822,987
印度,9906165,9422
巴西,6927145,6158
俄罗斯,2656601,210
法国,2433859,1833
英国,1874867,4048
土耳其,1866345,163
意大利,1855737,111
西班牙,1762036,196
阿根廷,1503222,134
哥伦比亚,1434516,1
德国,1357261,1012
墨西哥,1255974,927
波兰,1140572,8691
伊朗,1115770,8232
秘鲁,984973,91835
乌克兰,933737,5422
南非,866127,76274
荷兰,632085,7944,
印度尼西亚,623309,
```

◆ **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**



从该曲线图上可以明显看到，美国每日新增病例数“一超多强”，远远高于其他各国。剩下九个国家中，巴西的每日新增病例数较多。各国在 15 天内每日新增病例数量变化不定。

```
1. import numpy as np
2. import pandas as pd
3. import matplotlib.pyplot as plt
4. from pylab import mpl
```

```
mpl.rcParams['font.sans-serif'] = ['STZhongsong']    # 指定默认字体：解决 plot 不能显示中文问题
mpl.rcParams['axes.unicode_minus'] = False           # 解决保存图像是负号'-'显示为方块的问题

total={}
everyday={}

'''将所有国家 15 天的新增病例总数转化为字典'''
def readData():
    global total
    '''''依次读取 csv 文件'''
    for i in range(1,16):
        fileNameStr = 'result1/epidemic2020-12-'
        if i<10:
            fileNameStr+='0'+str(i)+'.csv'
        else:
            fileNameStr+=str(i)+'.csv'
        df = pd.read_csv(fileNameStr, encoding='utf-8',dtype=np.str,usecols=[0,1])
        df['新增确诊']=df['新增确诊'].astype(np.float).astype(np.int)

        for j in range(0, len(df)):
            if not df.iloc[j]['国家'] in everyday.keys():
                everyday[df.iloc[j]['国家']]=[]
            everyday[df.iloc[j]['国家']].append(df.iloc[j]['新增确诊'])

        if i==1:
            total=df.set_index('国家').T.to_dict('int')['新增确诊']
        else:
            for j in range(0, len(df)):
                if df.iloc[j]['国家'] in total.keys():
                    total[df.iloc[j]['国家']] += df.iloc[j]['新增确诊']

readData()

'''取新增病例总数前 10 的国家'''
top=sorted(total.items(),key=lambda x:x[1],reverse=True)[0:10]

'''绘制折线图'''
fig,ax = plt.subplots()
x = np.array(['12-'+str(one) for one in range(1,16)])
plt.title('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')
plt.xlabel('日期')
```

```
plt.ylabel('新增病例')
country=[]
for each in top:
    plt.plot(x, everyday[each[0]], linewidth=2.0,label=each[0])
    country.append(each[0])
plt.legend(country,bbox_to_anchor=(1.01, 0), loc=3, prop={'size': 8})
plt.show()
```

◆ 累计确诊人数占国家总人口比例最高的 10 个国家

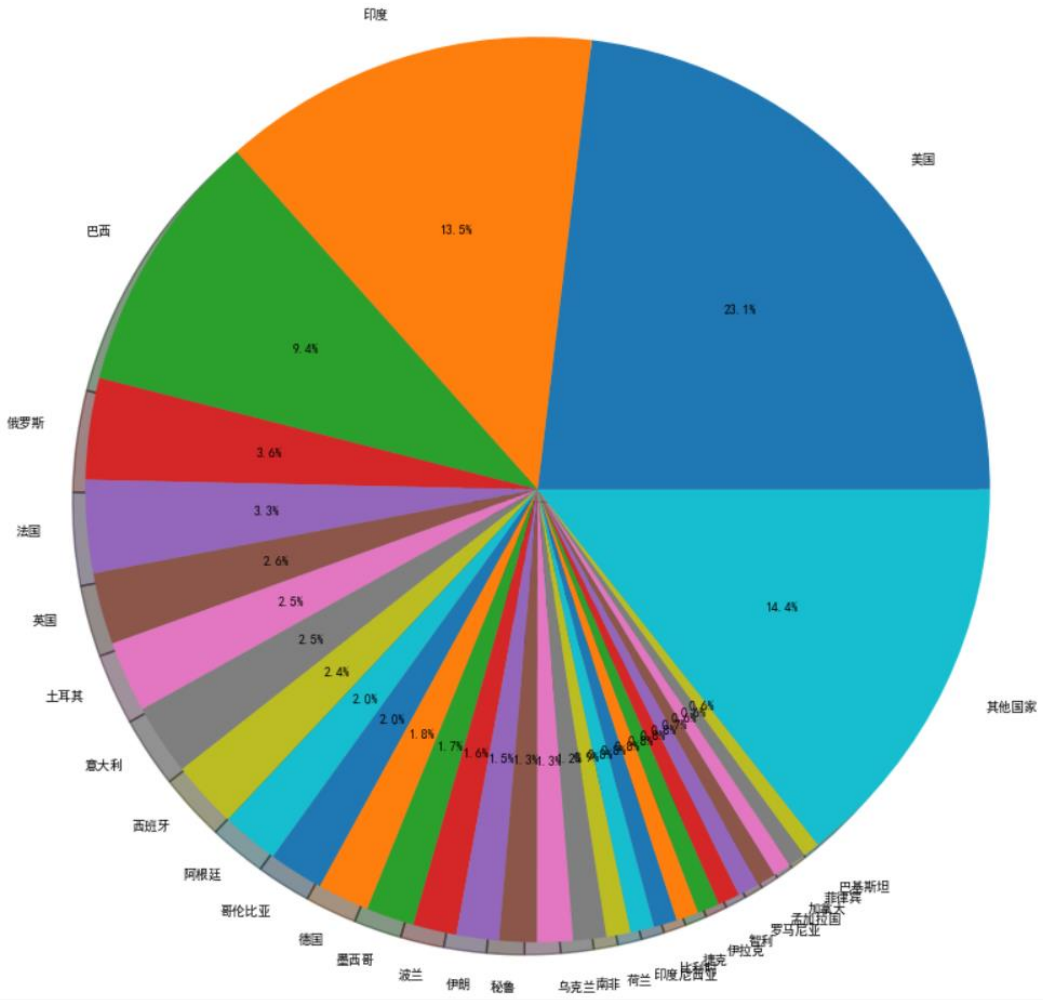
	国家	确诊率（%）
0	安道尔	9.8698
1	卢森堡	6.9601
2	黑山	6.7169
3	圣马力诺	5.7871
4	巴林	5.7842
5	捷克	5.511
6	比利时	5.3181
7	格鲁吉亚	5.2258
8	美国	5.1529
9	卡塔尔	5.1011

◆ 死亡率（累计死亡人数/累计确诊人数）最低的 10 个国家

	国家	死亡率（%）
0	梵蒂冈	0.0
1	老挝	0.0
2	东帝汶	0.0
3	圣文森特和格林纳丁斯	0.0
4	塞舌尔	0.0
5	柬埔寨	0.0
6	蒙古	0.0
7	不丹	0.0
8	厄立特里亚	0.0
9	新加坡	0.0497

```
import pandas as pd
import numpy as np

df = pd.read_csv('total.csv', encoding='utf-8',dtype=np.str)
df1 = df.sort_values(by="确诊率（%）
",ascending=False,ignore_index=True)[0:10]
print(df1[["国家","确诊率（%）"]])

df2 = df.sort_values(by="死亡率(%)",ascending=True,ignore_index=True)[0:10]
print(df2[["国家","死亡率（%）"]])
```

◆ 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）

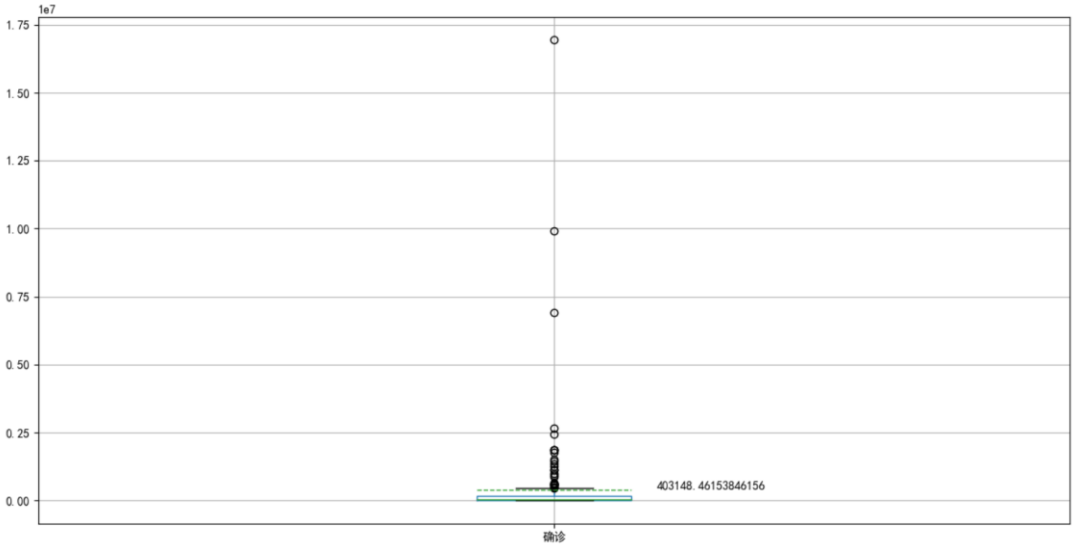


其中，其他国家为确诊人数排在第 30 位及以后的国家。

```
1. import matplotlib.pyplot as plt
2. import matplotlib
3. from matplotlib import font_manager as fm
4. import numpy as np
5. import pandas as pd
6.
7. matplotlib.rcParams['font.sans-serif'] = ['SimHei'] #支持中文的细黑体
8.
9. df = pd.read_csv('total.csv', encoding='utf-8',dtype=np.str)
10. label_list =df['国家'][:29].to_list()# 各国标签
11. size = df['确诊'][:29].astype(np.float).astype(np.int).to_list()# 各国确诊人数
12.
```

```
'''将确诊人数排在 29 位之后的国家进行合并处理'''
label_list.append('其他国家')
size.append(df['确诊'][29:].astype(np.float).astype(np.int).sum())

patches, texts, autotexts= plt.pie(size, labels=label_list,
labeldistance=1.1, autopct="%1.1f%%", shadow=True, startangle=0,
pctdistance=0.6)

#调整字体大小
proptease = fm.FontProperties()
proptease.set_size('x-small')
plt.setp(texts, fontproperties=proptease)
plt.setp(autotexts, fontproperties=proptease)
plt.show()
```

◆ 展示全球各个国家累计确诊人数的箱型图，要有平均值




```
import numpy as np
import pandas as pd
from pandas import Series,DataFrame
import matplotlib.pyplot as plt
from matplotlib import font_manager as fm
fm.rcParams['font.sans-serif'] = ['SimHei'] #支持中文的细黑体

#打开 CSV 文件
fileNameStr = 'total.csv'
df = pd.read_csv(fileNameStr,encoding='utf-8')

#绘制
ax = df.boxplot(column=['确诊'],meanline=True,showmeans=True,vert=True)
```

```
ax.text(1.1,df['确诊'].mean(),df['确诊'].mean())  #标注均值
plt.show()
```

◆ 全世界应对新冠疫情最好的 10 个国家

因为各国的人口基数（分母）不一样，不可直接比较死亡率、确诊率、治愈率。既然是全球统考评比，就应该以全球人口，全球感染人数和死亡人数作为共同基数（分母）来衡量：某个国家人口占全球人口的比例 A，该国感染病例占全球感染病例的比例 B，以及该国死亡人数占全球死亡总数的比例 C，如果 B 和 C 都是接近于或小于 A，那么该国应该说考评及格。

由此推出衡量各国应对疫情考评指标=（死亡人数全球占比+确诊人数全球占比*（1-治愈率））/人口全球占比

指标越小，代表应对疫情防控措施越好。

最终得到的应对疫情最好的 10 个国家如下：

	国家	指标	人口占比	死亡占比	确诊占比
179	老挝	0.000103	0.097438	0.000000	0.000059
170	柬埔寨	0.000323	0.222820	0.000000	0.000521
160	越南	0.001877	1.304342	0.002224	0.002019
168	坦桑尼亚	0.002359	0.764527	0.001334	0.000733
143	泰国	0.004672	0.907406	0.003812	0.006114
163	巴布亚新几内亚	0.005866	0.117075	0.000508	0.001044
73	中国	0.015976	19.512889	0.302467	0.137030
148	贝宁	0.018475	0.160496	0.002795	0.004449
112	刚果（金）	0.020823	1.214773	0.022617	0.020897
151	尼日尔	0.021916	0.305242	0.005209	0.003343

```
import pandas as pd
import numpy as np

df = pd.read_csv('total.csv', encoding='utf-8')
df=df.sort_values(by='人口',ascending=False)[0:100]#选取人口数前 100 的国家
df[['确诊率（%）','死亡率（%）','治愈率（%）']]=df[['确诊率（%）','死亡率（%）','治愈率（%）']].astype('float')

'''计算总人口、总感染数、总死亡数'''
total_pop=df['人口'].sum()
total_confirm=df['确诊'].sum()
```

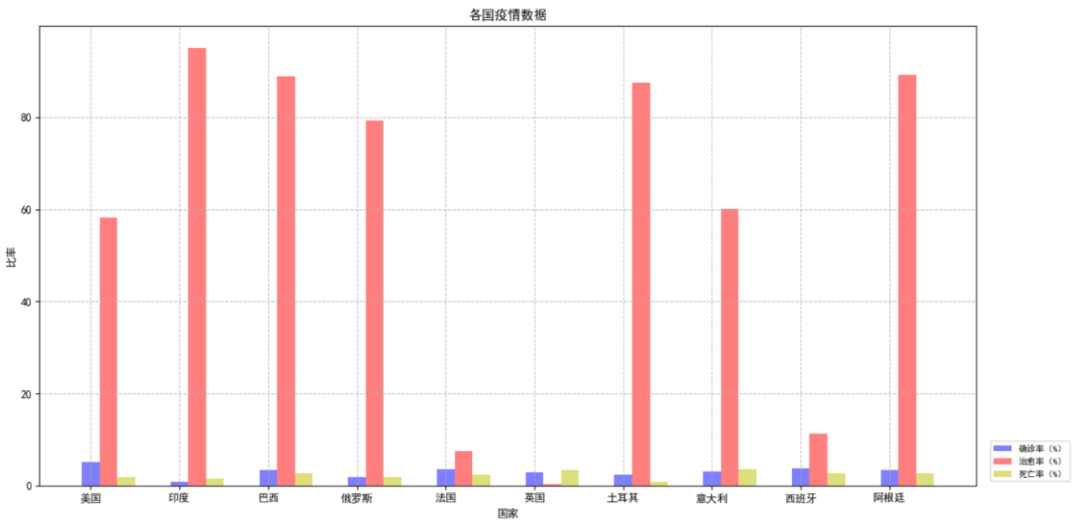
```
total_death=df['死亡'].sum()
total_cured=df['治愈'].sum()

'''计算各国人口、确诊、死亡全球占比'''
each_pop=round(100*df['人口']/total_pop,8)
each_confirm=round(100*df['确诊']/total_confirm,8)
each_death=round(100*df['死亡']/total_death,8)
'''计算指标，进行排名'''
index=(each_confirm*(1-df['治愈率（%）']/100)+each_death)/each_pop
result = pd.concat([df['国家
'],index,each_pop,each_death,each_confirm],axis=1,ignore_index=True)
result.columns = ['国家','指标','人口占比','死亡占比','确诊占比']
result=result.sort_values(by='指标',ascending=True)

print(result[0:10])
```

◆ 其他分析

1.治愈率（治愈人数/确诊人数）最高的 10 个国家

	国家	治愈率（%）
0	新加坡	99.8028
1	加蓬	98.4280
2	卡塔尔	98.2979
3	科特迪瓦	98.2170
4	吉布提	98.0638
5	巴林	97.8312
6	加纳	97.6365
7	赤道几内亚	97.5506
8	苏里南	97.5369
9	沙特阿拉伯	97.4387

2.确诊率前十名国家确诊率、死亡率、治愈率情况



可以看到，英国与法国的治愈率显著低于其他各国，印度的治愈率最高；美国的确诊率最高，印度的确诊率最低；英国和意大利的死亡率较高，土耳其的死亡率最低。

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

df = pd.read_csv('total.csv', encoding='utf-8')
result=df[0:10]

'''输出治愈率前 10'''
df1 = df.sort_values(by="治愈率（%）",ascending=False,ignore_index=True)
print(df1[0:10][["国家","治愈率（%）"]])

'''绘制确诊数前 10 国家确诊率、死亡率、治愈率图形'''
fig,ax=plt.subplots()
plt.rcParams['font.sans-serif'] = ['SimHei'] #添加对中文字体的支持

x=np.arange(1,11) #生成横轴数据
fig, ax = plt.subplots()
ax.set_title("各国疫情数据")
ax.set_xlabel("国家")
ax.set_ylabel("比率")
ax.set_xticklabels(result[0:10]['国家'].to_list())

y1=result[0:10]["确诊率（%）"].to_list()
y2=result[0:10]["治愈率（%）"].to_list()
```

```
y3=result[0:10]["死亡率（%）"].to_list()
plt.xticks(range(1,11))
plt.bar(x,y1,0.2,alpha=0.5,color='b')
plt.bar(x+0.2,y2,0.2,alpha=0.5,color='r')
plt.bar(x+0.4,y3,0.2,alpha=0.5,color='y')

plt.legend(["确诊率（%）","治愈率（%）","死亡率（%）
"],bbox_to_anchor=(1.01, 0), loc=3, prop={'size': 8})
plt.grid(True, linestyle='--', alpha=0.8) #设置网格线
plt.show()
```

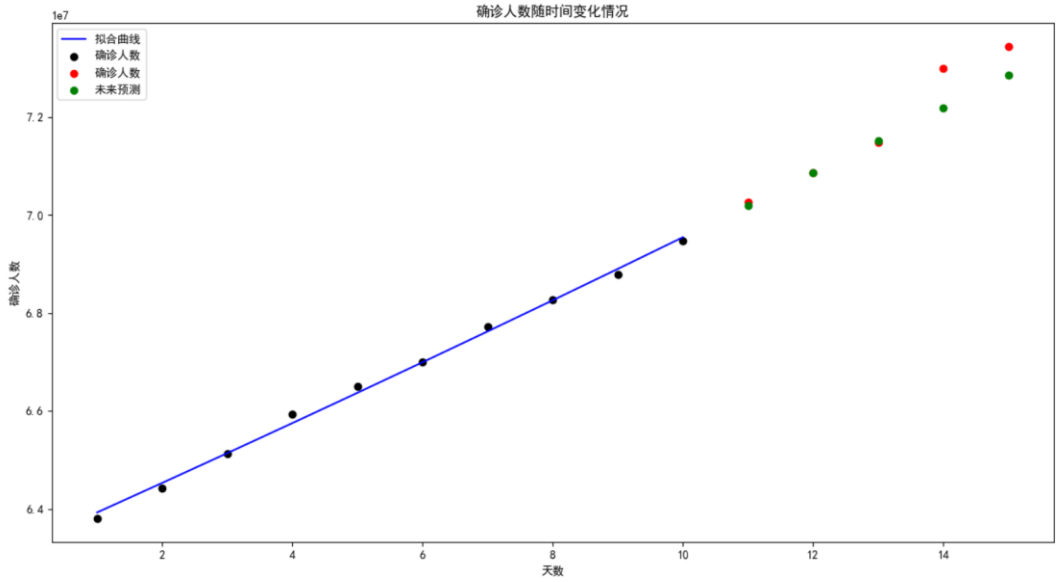
数据预测

◆ **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**

一般疾病的传播是 S 型增长的过程，因此此处利用 logistic 增长函数进行预测：

$$P(t) = \frac{KP_0e^{rt}}{K + P_0(e^{rt} - 1)}$$

作图如下，全球累计确诊数稳步递增。黑色点为 12-1 至 12-10 的真实数据，红色点为 12-11 至 12-15 的真实数据，绿色点为 12-11 至 12-15 的预测数据。



由于天数较少，因此这条曲线没有反映出 logistic 增长曲线的 S 形状特征，属于 S 形曲线的后期部分，近似一条斜率为正的直线。预测结果数据如下，12-11 至 12-13 与实际数据偏差较小，12-14 与 12-15 两日偏差较大。原因在于前 10 天数据拟合的曲线斜率较为稳定，而 14、15 号确诊人数变化较大，与曲线偏离。

```
============predict=============
[70198002.70149402 70855460.31812479 71518621.65213977 72187527.65878104
 72862219.42647265]
============real=============
[70259557 70855698 71486717 73001562 73437267]
```

```
1.  import numpy as np  # 导入数值计算模块
2.  import pandas as pd  # 导入数据处理模块
3.  import matplotlib.pyplot as plt  # 导入绘图模块
4.  from scipy.optimize import curve_fit  # 导入拟合模块
5.
6.  plt.rcParams["font.sans-serif"] = "SimHei"  # 黑体中文
7.  plt.rcParams["axes.unicode_minus"] = False  # 显示负号
8.
9.  confirm=[]
10.
11. '''依次读取 csv 文件'''
12. for i in range(1,16):
13.         fileNameStr = 'result1/epidemic2020-12-'
14.         if i<10:
```

```
            fileNameStr+='0'+str(i)+'.csv'
        else:
            fileNameStr+=str(i)+'.csv'
        df = pd.read_csv(fileNameStr, encoding='utf-8')

        confirm.append(df['确诊'].astype(np.float).astype(np.int).sum())

t = [1,2,3,4,5,6,7,8,9,10] # 构造横轴
t=np.array(t)
confirm=np.array(confirm)

# 散点图
fig = plt.figure(figsize=(16, 8))  # 建立画布
ax = fig.add_subplot(1, 1, 1)
ax.scatter(t, confirm[0:10], color="k", label="确诊人数")  # 真实数据散点图
ax.scatter([11,12,13,14,15], confirm[10:15], color="r", label="确诊人数
")  # 真实数据散点图

ax.set_xlabel("天数")  # 横坐标
ax.set_ylabel("确诊人数")  # 纵坐标
ax.set_title("确诊人数随时间变化情况")  # 标题

# 拟合
def logistic_increase_function(t,K,P0,r):
    # t:time   t0:initial time    P0:initial_value    K:capacity  r:increase
_rate
    r=0.01
    exp_value=np.exp(r*(t+330))   # 疫情约开始 330 天
    return (K*exp_value*P0)/(K+(exp_value-1)*P0)

coef, pcov = curve_fit(logistic_increase_function, t, confirm[0:10])  # 拟
合
print(coef)  # logistic 函数参数
y_values = logistic_increase_function(t, coef[0], coef[1], coef[2])  # 拟合 y
值
ax.plot(t, y_values, color="blue", label="拟合曲线")  # 画出拟合曲线

x = np.array([11,12,13,14,15])
y_predict = logistic_increase_function(x, coef[0], coef[1], coef[2])  # 未来
预测
ax.scatter(x, y_predict, color="green", label="未来预测")  # 未来预测散点
ax.legend()  # 加标签

print("============predict=============")
```

```
print(y_predict)
print("============real=============")
print(confirm[10:15])

plt.show()
```