

疫情数据追踪分析

随着新型冠状病毒肺炎疫情持续蔓延，大众对疫情的关注逐步增加。在这种背景下，明确疫情的走向，每日的动态尤为重要。数据中包含确诊，疑似，死亡，治愈等数据能直观地了解疫情的变化。在疫情爆发一段时间之后，各个省市行政机构，开始将疫情数据按照地域逻辑进行梳理，比如将全国各个省市、或是某地级市各个行政区域，用表格的方式呈现出来，用户在阅读体验上有所提升，能更加直观清晰地获取关键信息，因此我们对它做了数据可视化的分析。数据可视化是指将数据通过与地图、折线图、散点图等多种图表形式结合，在特定的数字场景中，让观众对问题有直观的视觉思维。

学习目标

- (1) 分析指定网页上所需要爬取的数据
- (2) 掌握爬取数据的基本思路和方法
- (3) 掌握数据预处理的基本方法和相关函数
- (4) 掌握数据可视化的基本原理与方法

1 分析指定网页上的疫情数据

任务描述

本次项目根据分析腾讯新闻 <https://news.qq.com/zt2020/page/feiyun.html> 网页上的数据进行爬取。首先通过 Chrome 浏览器登录数据来源页，使用 Chrome 的检查功能，多次刷新先分析要爬取的数据的规律，搜索全国确诊人数，地区名称，其它国家名称等信息，然后爬取所需页面的信息进行解析，获取日期，确诊，疑似，死亡，治愈等数据存储到数据库或者其它形式的文件。

1.1 了解数据来源页的数据

本任务主要针对腾讯新闻网页上的数据进行分析（以国内数据为例）。通过 Chrome 的检查功能可得出国内疫情数据存储页面为：
https://view.inews.qq.com/g2/getOnsInfo?name=disease_h5。部分数据参数如下：

“ lastUpdataTime: 最后更新时间、chinaTotal: 中国累计值、chinaAdd: 每日新增值、isShowAdd: 判断的状态码、showAddSwitch: 判断的状态码、areaTree: 数据树”

其中 chinaTotal 与 chinaAdd 字段一样:

chinaTotal / chinaAdd

confirm: 确诊
heal: 治愈
dead: 死亡
nowConfirm: 现有确诊
suspect: 疑似
nowSevere: 重症

areaTree (部分)

name
today
total
suspect
dead
deadRate: 死亡率
heal
healRate: 治愈率
children: 这个字典里面包含的是具体省份数据

2 获取疫情数据（全球、全国以及各地区）

任务描述

本任务需要从页面上获取相关的数据，涉及到的主要技术有 python 爬虫技术，主要使用 request 库爬取数据。

任务分析

对疫情数据的获取可以分为以下 2 个步骤。

- (1) 了解本项目数据获取的方法。
- (2) 获取数据的代码实现

2.1 了解本案例数据获取的方法

这里以获取全球疫情数据为例子：

打开腾讯新闻官网，点击抗肺炎，找到旁边的疫情实时追踪数据来源，也可直接打开网址：<https://news.qq.com/zt2020/page/feiyang.htm>



接下来，我们分别将全球数据、全国以及各地区数据和中国每日疫情数据爬取下来。一般情况下，爬取某个网页数据时，会首先使用 Python 获取某个网页的源代码，查看所需要的数据是否可以通过解析网页源代码获取。经过测试发现本案例所需要的数据无法通过读取网页源代码获取，因此可以借助 Google Chrome 的 Web 开发者工具找到疫情数据存放的位置。打开【网络（Network）】面板，可以看到从网络请求下载资源的实时信息，它是一个查看 JavaScript 触发的 XHR 对象所发起资源请求的理想工具。打开网络面板之后，当点击网页中【海外疫情】，可以发现网络面板会显示加载出来的资源，网络面板下的第一列会显示所有请求的文件名。如图 2-1 所示：

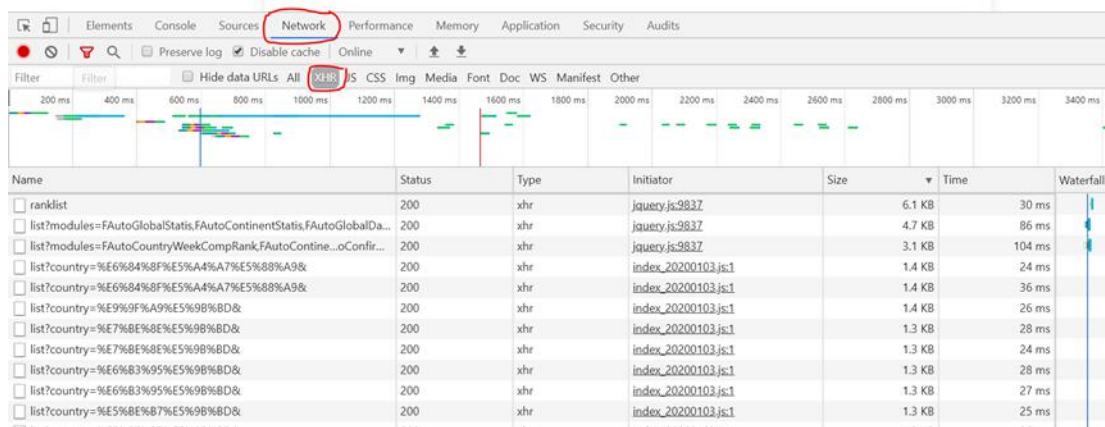


图 2-1 网络面板界面

左键单击第一个资源，会在右侧预览（Preview）标签下显示该资源的内容。如图 2-2 所示。

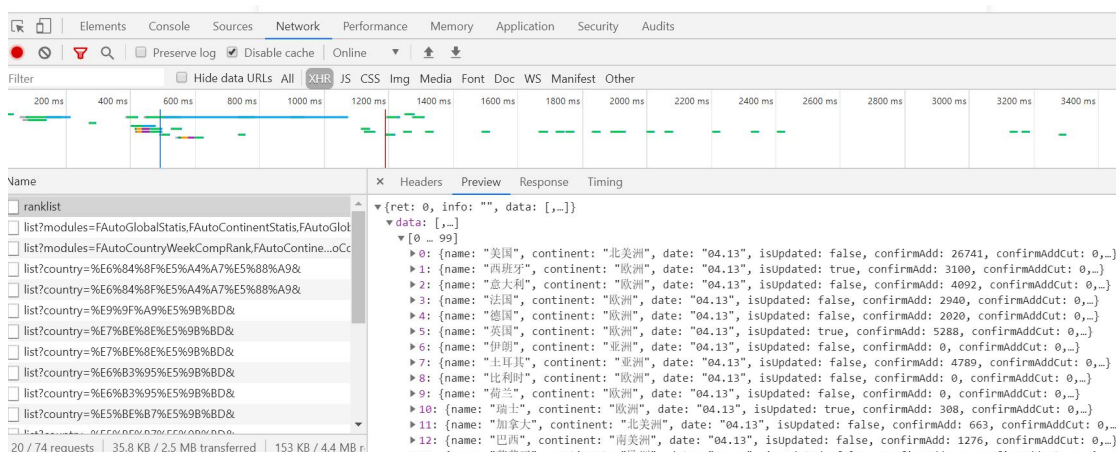
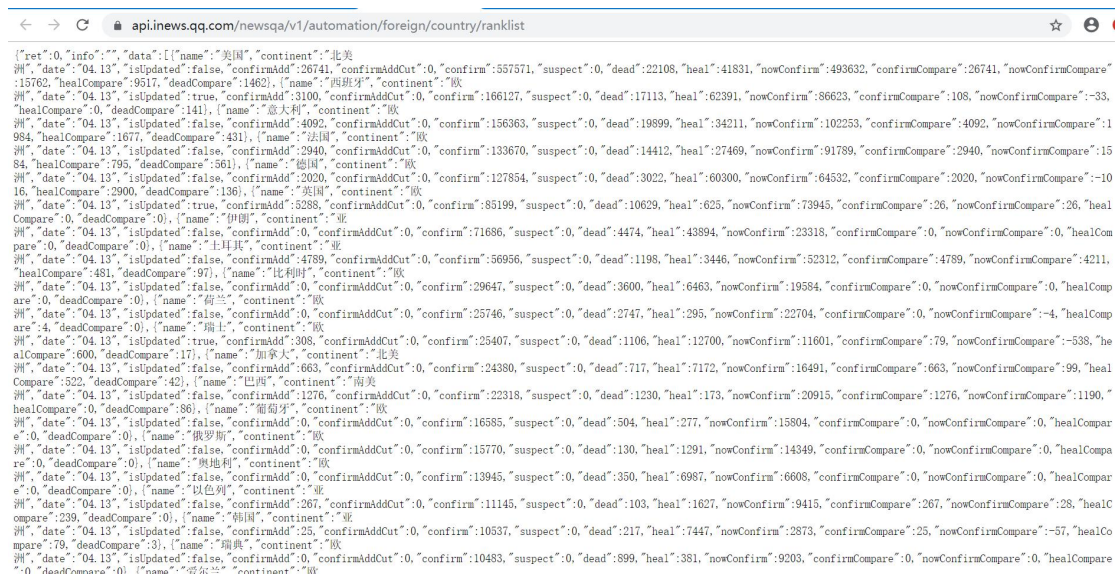


图 2-2 资源预览界面

想要更加仔细的查看该资源的信息，可以右键单击该资源，选择在新的标签页打开此连接【Open link in new tab】。弹出的页面如图 2-3 所示



```
{
  "ret": 0,
  "info": {
    "data": [
      {
        "name": "美国",
        "continent": "北美",
        "date": "04.13",
        "isUpdated": false,
        "confirmAdd": 26741,
        "confirmAddCut": 0,
        "confirm": 557571,
        "suspect": 0,
        "dead": 22108,
        "heal": 41831,
        "nowConfirm": 493632,
        "confirmCompare": 26741,
        "nowConfirmCompare": 15762,
        "deadCompare": 9517,
        "healCompare": 1462,
        "name": "西班牙",
        "continent": "欧洲",
        "date": "04.13",
        "isUpdated": true,
        "confirmAdd": 3100,
        "confirmAddCut": 0,
        "confirm": 166127,
        "suspect": 0,
        "dead": 17113,
        "heal": 62391,
        "nowConfirm": 86623,
        "confirmCompare": 108,
        "nowConfirmCompare": -33,
        "healCompare": 0,
        "deadCompare": 1411,
        "name": "意大利",
        "continent": "欧洲",
        "date": "04.13",
        "isUpdated": false,
        "confirmAdd": 4092,
        "confirmAddCut": 0,
        "confirm": 156363,
        "suspect": 0,
        "dead": 19899,
        "heal": 34211,
        "nowConfirm": 102253,
        "confirmCompare": 4092,
        "nowConfirmCompare": 1984,
        "healCompare": 1677,
        "deadCompare": 431,
        "name": "法国",
        "continent": "欧洲",
        "date": "04.13",
        "isUpdated": false,
        "confirmAdd": 2940,
        "confirmAddCut": 0,
        "confirm": 133670,
        "suspect": 0,
        "dead": 14412,
        "heal": 27469,
        "nowConfirm": 91789,
        "confirmCompare": 2940,
        "nowConfirmCompare": 1584,
        "healCompare": 795,
        "deadCompare": 561,
        "name": "德国",
        "continent": "欧洲",
        "date": "04.13",
        "isUpdated": false,
        "confirmAdd": 2020,
        "confirmAddCut": 0,
        "confirm": 127854,
        "suspect": 0,
        "dead": 3022,
        "heal": 60300,
        "nowConfirm": 64532,
        "confirmCompare": 2020,
        "nowConfirmCompare": -1016,
        "healCompare": 2900,
        "deadCompare": 136,
        "name": "英国",
        "continent": "欧洲",
        "date": "04.13",
        "isUpdated": true,
        "confirmAdd": 5288,
        "confirmAddCut": 0,
        "confirm": 85199,
        "suspect": 0,
        "dead": 10629,
        "heal": 625,
        "nowConfirm": 73945,
        "confirmCompare": 26,
        "nowConfirmCompare": 26,
        "healCompare": 0,
        "deadCompare": 0,
        "name": "伊朗",
        "continent": "亚洲",
        "date": "04.13",
        "isUpdated": false,
        "confirmAdd": 0,
        "confirmAddCut": 0,
        "confirm": 71686,
        "suspect": 0,
        "dead": 4474,
        "heal": 43894,
        "nowConfirm": 23318,
        "confirmCompare": 0,
        "nowConfirmCompare": 0,
        "healCompare": 0,
        "deadCompare": 0,
        "name": "土耳其",
        "continent": "亚洲",
        "date": "04.13",
        "isUpdated": false,
        "confirmAdd": 4789,
        "confirmAddCut": 0,
        "confirm": 56956,
        "suspect": 0,
        "dead": 1198,
        "heal": 3446,
        "nowConfirm": 52312,
        "confirmCompare": 4789,
        "nowConfirmCompare": 4211,
        "healCompare": 481,
        "deadCompare": 97,
        "name": "比利时",
        "continent": "欧洲",
        "date": "04.13",
        "isUpdated": false,
        "confirmAdd": 0,
        "confirmAddCut": 0,
        "confirm": 29647,
        "suspect": 0,
        "dead": 3600,
        "heal": 6463,
        "nowConfirm": 19584,
        "confirmCompare": 0,
        "nowConfirmCompare": 0,
        "healCompare": 0,
        "deadCompare": 0,
        "name": "荷兰",
        "continent": "欧洲",
        "date": "04.13",
        "isUpdated": false,
        "confirmAdd": 0,
        "confirmAddCut": 0,
        "confirm": 25746,
        "suspect": 0,
        "dead": 2747,
        "heal": 295,
        "nowConfirm": 22704,
        "confirmCompare": 0,
        "nowConfirmCompare": -4,
        "healCompare": 4,
        "deadCompare": 0,
        "name": "瑞士",
        "continent": "欧洲",
        "date": "04.13",
        "isUpdated": true,
        "confirmAdd": 308,
        "confirmAddCut": 0,
        "confirm": 25407,
        "suspect": 0,
        "dead": 1106,
        "heal": 12700,
        "nowConfirm": 11601,
        "confirmCompare": 79,
        "nowConfirmCompare": -538,
        "healCompare": 600,
        "deadCompare": 17,
        "name": "加拿大",
        "continent": "北美",
        "date": "04.13",
        "isUpdated": false,
        "confirmAdd": 683,
        "confirmAddCut": 0,
        "confirm": 24380,
        "suspect": 0,
        "dead": 717,
        "heal": 7172,
        "nowConfirm": 16491,
        "confirmCompare": 683,
        "nowConfirmCompare": 99,
        "healCompare": 522,
        "deadCompare": 42,
        "name": "巴西",
        "continent": "南美",
        "date": "04.13",
        "isUpdated": false,
        "confirmAdd": 1276,
        "confirmAddCut": 0,
        "confirm": 22318,
        "suspect": 0,
        "dead": 1230,
        "heal": 173,
        "nowConfirm": 20915,
        "confirmCompare": 1276,
        "nowConfirmCompare": 1190,
        "healCompare": 0,
        "deadCompare": 86,
        "name": "葡萄牙",
        "continent": "欧洲",
        "date": "04.13",
        "isUpdated": false,
        "confirmAdd": 0,
        "confirmAddCut": 0,
        "confirm": 16585,
        "suspect": 0,
        "dead": 504,
        "heal": 277,
        "nowConfirm": 15804,
        "confirmCompare": 0,
        "nowConfirmCompare": 0,
        "healCompare": 0,
        "deadCompare": 0,
        "name": "俄罗斯",
        "continent": "欧洲",
        "date": "04.13",
        "isUpdated": false,
        "confirmAdd": 0,
        "confirmAddCut": 0,
        "confirm": 15770,
        "suspect": 0,
        "dead": 130,
        "heal": 1291,
        "nowConfirm": 14349,
        "confirmCompare": 0,
        "nowConfirmCompare": 0,
        "healCompare": 0,
        "deadCompare": 0,
        "name": "奥地利",
        "continent": "欧洲",
        "date": "04.13",
        "isUpdated": false,
        "confirmAdd": 0,
        "confirmAddCut": 0,
        "confirm": 13945,
        "suspect": 0,
        "dead": 350,
        "heal": 6987,
        "nowConfirm": 16608,
        "confirmCompare": 0,
        "nowConfirmCompare": 0,
        "healCompare": 0,
        "deadCompare": 0,
        "name": "以色列",
        "continent": "亚洲",
        "date": "04.13",
        "isUpdated": false,
        "confirmAdd": 267,
        "confirmAddCut": 0,
        "confirm": 11145,
        "suspect": 0,
        "dead": 103,
        "heal": 1627,
        "nowConfirm": 9415,
        "confirmCompare": 267,
        "nowConfirmCompare": 28,
        "healCompare": 239,
        "deadCompare": 0,
        "name": "韩国",
        "continent": "亚洲",
        "date": "04.13",
        "isUpdated": false,
        "confirmAdd": 25,
        "confirmAddCut": 0,
        "confirm": 10537,
        "suspect": 0,
        "dead": 217,
        "heal": 7447,
        "nowConfirm": 2873,
        "confirmCompare": 25,
        "nowConfirmCompare": -57,
        "healCompare": 79,
        "deadCompare": 3,
        "name": "瑞典",
        "continent": "欧洲",
        "date": "04.13",
        "isUpdated": false,
        "confirmAdd": 0,
        "confirmAddCut": 0,
        "confirm": 10483,
        "suspect": 0,
        "dead": 899,
        "heal": 381,
        "nowConfirm": 9203,
        "confirmCompare": 0,
        "nowConfirmCompare": 0,
        "healCompare": 0,
        "deadCompare": 0,
        "name": "爱尔兰",
        "continent": "欧洲"
      }
    ]
  }
}
```

图 2-3 带有全球疫情数据的 JSON 文件预览

通过图 2-3 可以看出，里面包含了这个页面的全球疫情数据，并且该数据以 JSON 格式进行存储。将 URL 复制下来，全球疫情数据的获取就完成了。

2.2 任务代码实现

2.2.1 全球数据获取

利用上面获取到的 URL，使用 `request.get()` 读取该网页下的数据，将数据转换成 json 数据类型，如图 2-4 所示

```
1 import requests
2 import json
3 import pandas as pd
4 #目标网站
5 data=requests.get('https://api.inews.qq.com/newsqa/v1/automation/foreign/country/ranklist')
6 #转换成json数据类型
7 data=data.json()
```

图 2-4

获取所需要的数据并将数据保存为 csv 文件，如图 2-5 所示

```
1 global_data = pd.DataFrame(data['data'])
2 global_data['confirm'] = global_data['confirm']
3 global_data['suspect'] = global_data['suspect']
4 global_data['dead'] = global_data['dead']
5 global_data['heal'] = global_data['heal']
6 global_data['addconfirm'] = global_data['confirmAdd']
7 global_data['name'] = global_data['name']
8 global_data = global_data[['name', 'confirm', 'suspect', 'dead', 'heal', 'addconfirm']]
9 global_data=pd.DataFrame(global_data)
10 global_data.to_csv("全球的数据.csv", index = False) #把数据保存为csv文件（去掉索引）
```


图 2-5

2.2.2 全国以及各地区数据获取

将利用上面方法获取的全国数据 URL 爬取全国数据。查看爬取下来的数据是否为标准的 JSON 格式, 如果格式不够标准, 则将符合标准格式的数据提取出来。接着使用 `json.loads()` 将 JSON 格式数据转为字典类型, 并提取评论数据, 如图 2-6 所示

```
1 import requests
2 import json
3 import pandas as pd
4 def getData():
5     url = 'https://view.inews.qq.com/g2/getOnsInfo?name=disease_h5'
6     headers = {
7         'user-agent': 'Mozilla/5.0 (iPhone; CPU iPhone OS 11_0 like Mac OS X) AppleWebKit/604.1.38
8     }
9     r = requests.get(url, headers)
10    if r.status_code == 200:
11        return json.loads(json.loads(r.text)['data'])
12 data = getData()
```

图 2-6 爬取中国省份数据

利用 for 循环爬取全国以及各地区数据, 如图 2-7 所示

```
1 province_list = list()
2 for province in data_dict.get('areaTree')[0]['children']:
3     province_info = province['total']
4     province_info['name'] = province['name']
5     province_list.append(province_info)
6 province_df = pd.DataFrame(province_list)
7 province_df = province_df.select_dtypes(exclude=['bool']) #清除bool类型数据
8 province_df.to_csv('中国省份的疫情数据.csv', index=False, encoding="utf-8")
```

图 2-7 处理并保存为中国省份数据

利用这数据做“中国总确诊地图”, 如图 2-8 所示

```
1 china_total_data = province_df.groupby("name")["confirm"].sum().reset_index()
2 china_total_data.columns = ["province", "confirm"]
3 china_total_data.to_csv('中国总确诊数据.csv', index=False)
```

图 2-8 处理并保存为中国总确诊数据

利用这数据做“中国现有确诊地图”, 如图 2-9 所示

```
1 china_now_data = province_df.groupby("name")["nowConfirm"].sum().reset_index()
2 china_now_data.columns = ["province", "nowConfirm"]
3 china_now_data.to_csv('中国现有确诊数据.csv', index=False)
```

图 2-9 处理并保存为中国现有确诊数据

利用这数据做“广东地图”, 因为获取数据里没有云浮市的数据, 所以手动添加上去了, 第 3 行代码是去掉“境外输入”和“无症状感染者”, 第 4 行代码是改变每个市的名称, 比如:

广州—>广州市（可视化时需要），如图 2-10 所示

```
1 gd_confirm_data=city_df.loc[city_df.province=='广东',['city','confirm']]
2 gd_confirm_data.loc['86']=['云浮','0']
3 gd_confirm_data.drop(labels=[64,75],axis=0,inplace=True)#行标签,删除2和4行
4 gd_confirm_data['city']=gd_confirm_data['city'].map(lambda x:x+"市")
5 gd_confirm_data.to_csv('广东各城市确诊的人数.csv',index=False,encoding='utf-8')
```

图 2-10 处理并保存为广东各城市确诊人数

利用这数据做“湖北地图”第 3、4 行代码修改湖北地图中的名称，如图 2-11 所示

```
1 hubei_confirm_data=city_df.loc[city_df.province=='湖北',['city','confirm']]
2 hubei_confirm_data['city']=hubei_confirm_data['city'].map(lambda x:x+"市")
3 hubei_confirm_data.loc[32,'city']='神农架林区'
4 hubei_confirm_data.loc[22,'city']='恩施土家族苗族自治州'
5 hubei_confirm_data.to_csv('湖北各城市确诊的人数.csv',index=False,encoding='utf-8')
```

图 2-11 处理并保存为湖北各城市确诊人数

2.2.3 每日疫情变化数据获取

获取国内每日的疫情数据变化，如图 2-12 所示

```
1 data=requests.get('https://view.inews.qq.com/g2/getOnsInfo?name=disease_other')
2 #转换成json数据类型
3 data = data.json()
4 processing_1 = data['data']
5 # eval作用是, 返回传入字符串的表达式的结果。即变量赋值时, 等号右边的表示是写成字符串的格式, 返回值就是这个表达式的结果。
6 processing_2 = eval('{' + processing_1 + '}')
7 processing_3 = processing_2['chinaDayList']
8 chinaDayList_data = pd.DataFrame(processing_3)
9 chinaDayList_data['date'] = chinaDayList_data['date']
10 chinaDayList_data['confirm'] = chinaDayList_data['confirm']
11 chinaDayList_data['suspect'] = chinaDayList_data['suspect']
12 chinaDayList_data['dead'] = chinaDayList_data['dead']
13 chinaDayList_data['heal'] = chinaDayList_data['heal']
14 chinaDayList_data['deadRate'] = chinaDayList_data['deadRate']
15 chinaDayList_data['healRate'] = chinaDayList_data['healRate']
16 chinaDayList_data['importedCase'] = chinaDayList_data['importedCase']
17 chinaDayList_data['noInfect'] = chinaDayList_data['noInfect']
18 chinaDayList_data = chinaDayList_data[['date','confirm','suspect','dead','heal','deadRate','healRate','importedCase','noInfect']]
19 chinaDayList_data = pd.DataFrame(chinaDayList_data)
20 chinaDayList_data.to_csv('中国每天的数据变化.csv',index=False)#把数据保存为csv文件(去掉索引)
```

图 2-12 处理并保存为中国每天疫情数据变化

获取国外每天的数据变化，如图 2-13 所示

```
1 import requests
2 import json
3 import pandas as pd
4 data=requests.get('https://api.inews.qq.com/newsqa/v1/automation/modules/list?modules=FAutoGlobalStatis,')
5 #转换成json数据类型
6 data=data.json()
7 global_list = list()
8 for date in data.get('data')['FAutoGlobalDailyList']:
9     global_info = date['all']
10    global_info['date'] = date['date']
11    global_list.append(global_info)
12 globalDayList_data = pd.DataFrame(global_list)
13 globalDayList_data.to_csv('国外每天的数据变化.csv',index=False,encoding='utf-8')
```

图 2-13 处理并保存为国外每天数据变化

3 对疫情数据进行预处理

任务描述

由于获取到的数据格式不是我们需要的，所以要对数据做预处理。本任务所用到的是 pandas 库。

任务分析

对疫情数据预处理可以分为以下 2 个步骤。

- (1) 了解 pandas
- (2) 任务代码实现

3.1.1 Pandas 基本介绍

Pandas 是基于 NumPy 的一种工具，该工具是为了解决数据分析任务而创建的。Pandas 纳入了大量库和一些标准的数据模型，提供了高效地操作大型数据集所需的工具。pandas 提供了大量能使我们快速便捷地处理数据的函数和方法。同时它是使 Python 成为强大而高效的数据分析环境的重要因素之一。

Pandas 是 python 的一个数据分析包，最初由 AQR Capital Management 于 2008 年 4 月开发，并于 2009 年底开源出来，目前由专注于 Python 数据包开发的 PyData 开发 team 继续开发和维护，属于 PyData 项目的一部分。Pandas 最初被作为金融数据分析工具而开发出来，因此，pandas 为时间序列分析提供了很好的支持。Pandas 的名称来自于面板数据(panel data)和 python 数据分析(data analysis)。panel data 是经济学中关于多维数据集的一个术语，在 Pandas 中也提供了 panel 的数据类型。

数据结构：

Series：一维数组，与 Numpy 中的一维 array 类似。二者与 Python 基本的数据结构 List 也很相近。Series 如今能保存不同种数据类型，字符串、boolean 值、数字等都能保存在 Series 中。

Time- Series：以时间为索引的 Series。

DataFrame：二维的表格型数据结构。很多功能与 R 中的 data.frame 类似。可以将 DataFrame 理解为 Series 的容器。

Panel：三维的数组，可以理解为 DataFrame 的容器。

Panel4D: 是像 Panel 一样的 4 维数据容器。

PanelND: 拥有 factory 集合, 可以创建像 Panel4D 一样 N 维命名容器的模块。

而在本项目中我们主要是使用了 Pandas 库中的 DataFrame 对数据进行处理。

3.1.2 任务代码实现（截取部分代码为例）

导入 pandas 库并改名为 pd, 如 图 3-1:

```
1 import pandas as pd
```

将数据转换成 dataframe 对象, 并清除爬取到数据的 bool 类型数据, 如图 3-2 所示

```
1 province_df = pd.DataFrame(province_list)
2 province_df = province_df.select_dtypes(exclude=['bool']) #清除bool类型数据
```

图 3-2

将数据按照 name 进行分类, 将同一类的 name 对应的 confirm 进行汇总, 利用 to_csv 方法保存为 csv 文件, 如图 3-3 所示

```
1 china_total_data = province_df.groupby("name")["confirm"].sum().reset_index()
2 china_total_data.columns = ["province", "confirm"]
3 china_total_data.to_csv('中国总确诊数据.csv', index=False)
```

图 3-3

利用 pandas 的 loc 函数添加数据, drop 函数删除数据, 如图 3-4 所示

```
1 gd_confirm_data=city_df.loc[city_df.province=='广东',['city','confirm']]
2 gd_confirm_data.loc['86']='云浮','0' #添加一行数据
3 gd_confirm_data.drop(labels=[64,75],axis=0,inplace=True) #行标签, 删除2和4行
```

图 3-4

4 数据可视化分析

任务描述

使用 Pyecharts 等库绘制全国或者某省份和地级市的疫情变化趋势图、疫情分布地图。

本任务主要涉及技术为 pyechart 数据可视化工具。

任务分析

对疫情数据可视化分析可以分为以下 3 个步骤。

- (1) 了解 pyecharts 的基本原理与使用
- (2) 项目用到 pyecharts 的部分说明

(3) 可视化代码实现及根据图表进行分析

4.1 Pyecharts 的基本使用

Pyecharts 是一个用于生成 Echarts 图表的类库。echarts 是百度开源的一个数据可视化 JS 库，主要用于数据可视化。pyecharts 是一个用于生成 Echarts 图表的类库。实际上就是 Echarts 与 Python 的对接。

相关函数、参数说明

`add()` 主要方法，用于添加图表的数据和设置各种配置项。

`show_config()` 打印输出图表的所有配置项。

`render()` 默认将会在根目录下生成一个 `render.html` 的文件，支持 `path` 参数，设置文件保存位置，如 `render(r'e:my_first_chart.html')`，文件用浏览器打开。

基本上所有的图表类型都是这样绘制的：

`chart_name = Type()` 初始化具体类型图表。

`add()` 加数据及配置项。

`render()` 显示图表

4.2 项目用到 pyecharts 的部分说明

功能	代码
决定画布大小	<code>Line(init_opts=opts.InitOpts(width='1000px',height='500px'))</code>
控制是否在线上显示各个点的值	<code>.set_series_opts(label_opts=opts.LabelOpts(is_show=False))</code>
x 轴坐标名称	<code>xaxis_opts=opts.AxisOpts(name="日期")</code> ，x 轴坐标名称
y 轴坐标名称	<code>yaxis_opts=opts.AxisOpts(name="人数")</code>
标题	<code>title_opts=opts.TitleOpts(title="中国疫情数据变化图")</code>
x 轴滚动条	<code>datazoom_opts=opts.DataZoomOpts()</code>
工具栏	<code>toolbox_opts=opts.ToolboxOpts(is_show=True)</code>

4.3 任务代码实现及分析

4.3.1 全国以及地区可视化地图

首先我们通过导入 csv 文件引入数据，利用 for 循环将字符串转换成相对应的数据类型，如图 4-1 所示（这里以中国总确诊数据为例）

```
1 import csv
2 china_total_data='中国总确诊数据.csv'
3 with open(china_total_data,'r',encoding="utf-8") as file:
4     #1. 创建阅读器对象
5     reader=csv.reader(file)
6     #2. 读取文件头信息
7     header_row=next(reader)
8     #3. 保存数据
9     china_province=[]
10    china_confirm=[]
11    for row in reader:
12        #4. 将字符串转换为整型数据
13        china_province.append((row[0]))
14        china_confirm.append(int(row[1]))
```

图 4-1 导入中国总确诊数据

导入所需要的图表

```
1 from pyecharts import options as opts
2 from pyecharts.charts import Line, Pie, Tab, Gauge, Geo, Map
3 from pyecharts.commons.utils import JsCode
4 from pyecharts.globals import GeoType, RenderType
```

图 4-2 导入所需要的图表

接下来通过 Pyecharts 进行中国疫情地图的绘制，分为动态与静态。动态地图代码如图 4-3 所示，成果如图 4-4 所示；静态地图代码和动态的差不多就不做展示了，成果如图 4-5 所示

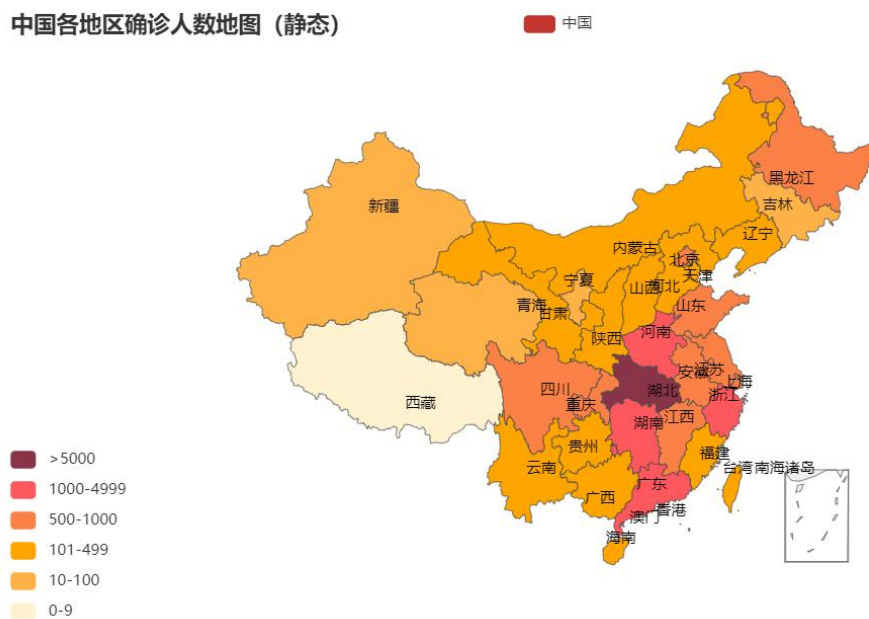
```
1 def china_confirm_geo() -> Geo:
2     c = (
3         Geo(init_opts=opts.InitOpts(width="1000px", height="500px"))
4         .add_schema(maptype="china", itemstyle_opts=opts.ItemStyleOpts(color="rgb(49, 60, 72)", border_color="rgb(0, 0, 0)"))
5         .add_series_name("中国", data_pair=[(i, j) for i, j in zip(china_province, china_confirm)], type_=GeoType.EFFECT_SCATTER)
6         .set_series_opts(label_opts=opts.LabelOpts(is_show=False), effect_opts=opts.EffectOpts(scale=6))
7         .set_global_opts(title_opts=opts.TitleOpts(title="全国各省份总确诊人数地图"), visualmap_opts=opts.VisualMapOpts(is_piecewise=True,
8             pieces = [
9                 {"min": 5000, "label": ">5000", "color": "#893448"}, #不指定 max, 表示 max 为无限大
10                {"min": 1000, "max": 4999, "label": "1000-4999", "color": "#ff585e"},
11                {"min": 500, "max": 999, "label": "500-1000", "color": "#fb8146"},
12                {"min": 101, "max": 499, "label": "101-499", "color": "#ffa500"},
13                {"min": 0, "max": 100, "label": "0-99", "color": "#ff2d11"}]))
14    )
15    return c
```

图 4-3 动态中国地图可视化代码

全国各省份总确诊人数地图



中国各地区确诊人数地图（静态）



除了中国地图以外，我们还可以进行地区地图绘制
方法与中国地图一致（这里以湖北省为例），代码如图 4-6，成果如图 4-7 所示

```

1 def hubei_confirm_map() -> Map:
2     c = (
3         Map(init_opts=opts.InitOpts(width="1000px",height="500px"))
4         .add('湖北',[i,j] for i,j in zip(hubei_city,hubei_confirm)), '湖北' )
5     .set_global_opts(title_opts=opts.TitleOpts(title="湖北各城市总确诊人数地图"),visualmap_opts=opts.VisualMapOpts(is_piecewise=True,
6         pieces = [
7             {"min": 5000, "label": ">5000", "color": "#893448"}, #不指定 max, 表示 max 为无限大
8             {"min": 1000, "max": 4999, "label": "1000-4999", "color": "#ff585e"},
9             {"min": 500, "max": 999, "label": "500-1000", "color": "#fb8146"},
10            {"min": 101, "max": 499, "label": "101-499", "color": "#ffa500"},
11            {"min": 0, "max": 100, "label": "0-9", "color": "#fff2d1"}]))
12
13 )
14 return c

```

图 4-6 湖北各城市总确诊人数可视化代码

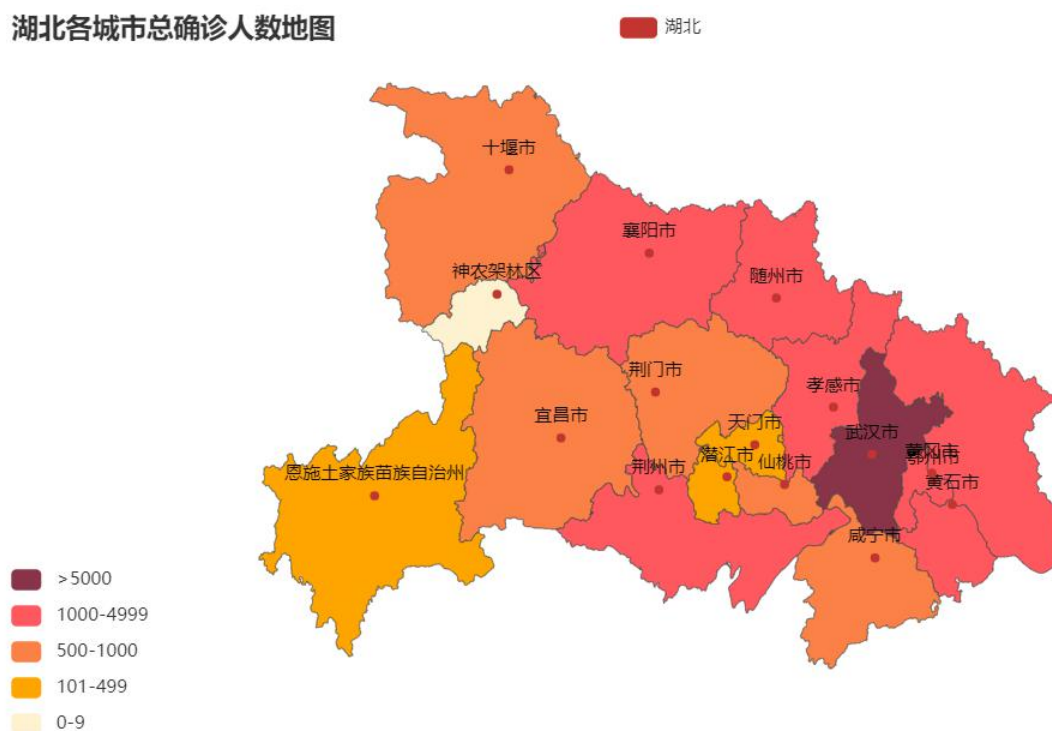


图 4-7 湖北各城市总确诊可视化图表

疫情地图是最常见的疫情数据可视化的方式。我们可以清晰直观的看到各个省，各个市确诊人数，大到全国各省市，小到街道社区，都可以通过地图结合数据的形式来呈现。

4.3.2 每日疫情变化可视化分析

首先我们通过读取 csv 文件，再将字符串转换成相对应的数据类型，如图 4-8 所示（以中国每天新增数据为例）


```

1 # 读取《中国每天的新增数据变化》的数据并把数据处理成列表类型
2 china_add_data='中国每天的新增数据变化.csv'
3 with open(china_add_data,'r') as file:
4     #1. 创建阅读器对象
5     reader=csv.reader(file)
6     #2. 读取文件头信息
7     header_row=next(reader)
8     #3. 保存数据
9     date_addconfirm=[]
10    country_addconfirm=[]
11    hubei_addconfirm=[]
12    for row in reader:
13        #4. 将字符串转换为整型数据
14        date_addconfirm.append(row[0])
15        hubei_addconfirm.append(int(row[1]))
16        country_addconfirm.append(int(row[2]))

```

图 4-8 读取中国每天的新增数据变化的代码

接着我们对疫情数据每日的变化新增人数，治愈人数，死亡人数的变化绘制折线图，中国疫情数据变化代码如图 4-9 所示，可视化图表如图 4-10 所示；国外疫情数据变化代码如图 4-11 所示，可视化图表如图 4-12 所示；

```

1 def chinaDayList_line() -> Line:
2     c = (
3         Line(init_opts=opts.InitOpts(width='1000px',height='500px')) #画布大小
4         .add_xaxis(xaxis_data=chinaDayList_date)
5         .add_yaxis(series_name='确诊', y_axis=chinaDayList_confirm)
6         .add_yaxis(series_name='疑似', y_axis=chinaDayList_suspect)
7         .add_yaxis(series_name='死亡', y_axis=chinaDayList_dead)
8         .add_yaxis(series_name='治愈', y_axis=chinaDayList_heal)
9         .set_series_opts(label_opts=opts.LabelOpts(is_show=False)) #控制是否在线上显示各个点的值
10        .set_global_opts(
11            xaxis_opts=opts.AxisOpts(name="日期"), #x轴坐标名称
12            yaxis_opts=opts.AxisOpts(name="人数"), #y轴坐标名称
13            title_opts=opts.TitleOpts(title="中国疫情数据变化图"), #标题
14            datazoom_opts=opts.DataZoomOpts(), #x轴滚动条
15            toolbox_opts=opts.ToolboxOpts(is_show=True) #工具栏
16        )
17    )
18    return c

```

图 4-9 中国疫情数据变化图代码

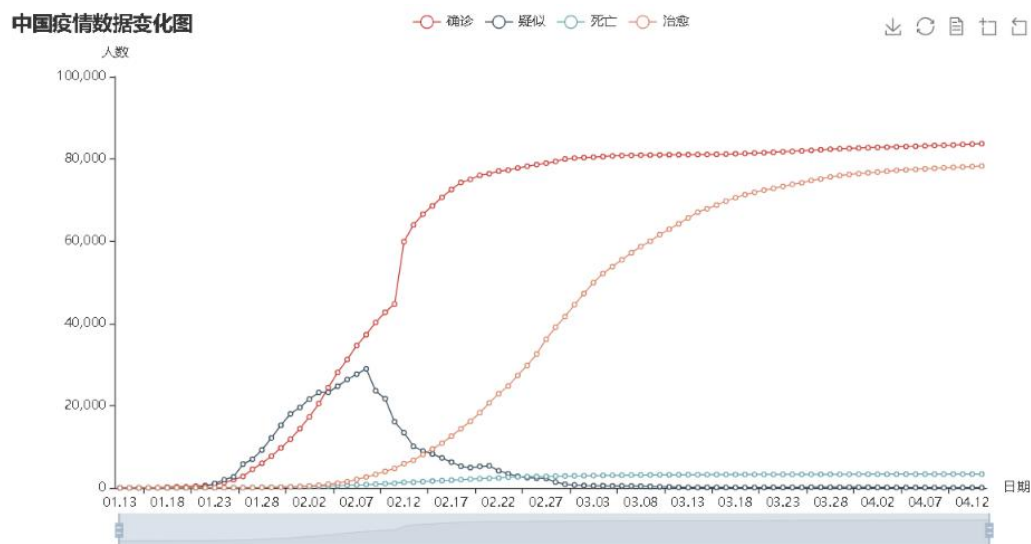


图 4-10 中国疫情数据变化可视化图表

从上图我们可以直观看到中国的总确诊人数上升的越来越缓慢，疑似人数的变化趋势从上升到下降，说明中国疫情得到了控制。

```

1 def globalDayList_line() -> Line:
2     c =(
3         Line(init_opts=opts.InitOpts(width='1000px', height='500px'))
4         .add_xaxis(xaxis_data=globalDayList_date)
5         .add_yaxis(series_name='总确诊', y_axis=globalDayList_confirm)
6         .add_yaxis(series_name='新增确诊', y_axis=globalDayList_newAddConfirm)
7         .add_yaxis(series_name='死亡', y_axis=globalDayList_dead)
8         .add_yaxis(series_name='治愈', y_axis=globalDayList_heal)
9         .set_series_opts(label_opts=opts.LabelOpts(is_show=False))
10        .set_global_opts(
11            xaxis_opts=opts.AxisOpts(name="日期"),
12            yaxis_opts=opts.AxisOpts(name="人数"),
13            title_opts=opts.TitleOpts(title="国外疫情数据变化图"),
14            toolbox_opts=opts.ToolboxOpts(is_show=True),
15            datazoom_opts=opts.DataZoomOpts(),
16        )
17    )
18    return c

```

图 4-11 国外疫情数据变化代码

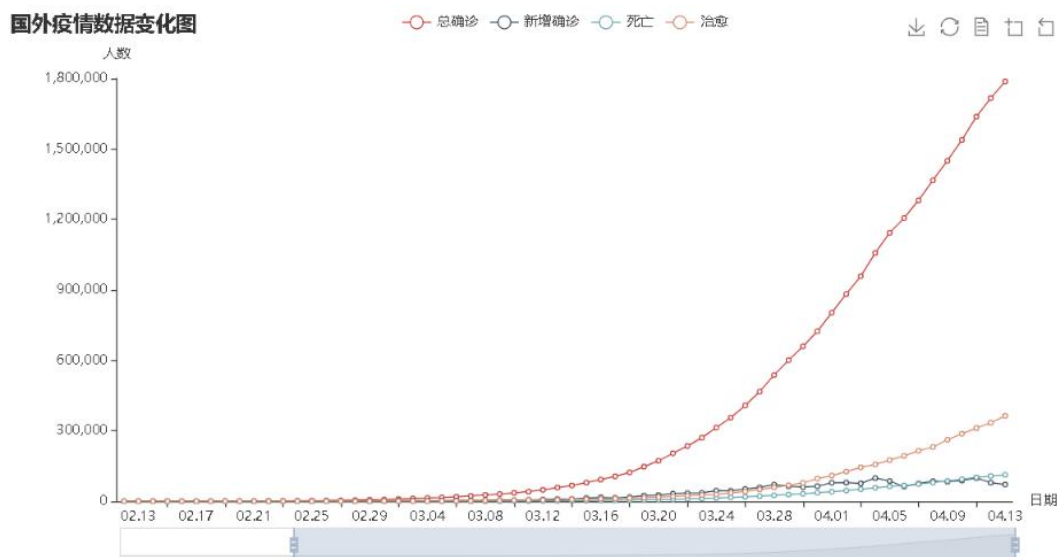


图 4-12 国外疫情数据变化可视化图表

折线图常用于绘制和表达连续的数据。从以上的图我们可以清晰直观的对比到过往与现在的数据分析，了解到国内以及国外的确诊人数，治愈人数，和死亡人数。

4.3.3 确诊人数 top10 可视化

首先通过 pandas 的 `sort_values` 方法对数据进行处理

```
1 #处理top10的数据
2 china_confirm = pd.read_csv('中国总确诊数据.csv', encoding='utf-8') #csv的编码格式, 大多数为utf-8
3 china_top10 = china_confirm.sort_values(["confirm"], ascending=False).head(11)
4 gd_confirm = pd.read_csv('广东各城市确诊的人数.csv', encoding='utf-8') #csv的编码格式, 大多数为utf-8
5 gd_top10 = gd_confirm.sort_values(["confirm"], ascending=False).head(10)
6 global_confirm = pd.read_csv('全球的数据.csv', encoding='utf-8') #csv的编码格式, 大多数为utf-8
7 global_top10 = global_confirm.sort_values(["confirm"], ascending=False).head(10)
```

图 4-13 处理 top10 的代码

```

1 def china_top10_pie() -> Pie:
2     c = (
3         Pie(init_opts=opts.InitOpts(width='1000px', height='500px'))
4         .add(
5             "",
6             [list(z) for z in zip(list(china_top10["province"][1:10]), list(china_top10["confirm"][1:10]))],
7             radius=["40%", "55%"],
8             label_opts=opts.LabelOpts(
9                 position="outside",
10                formatter="{a|{a}}{abg|}\n{hr|}\n{b|{b}}:{c} {per|{d}%} ",
11                background_color="#eee",
12                border_color="#aaa",
13                border_width=1,
14                border_radius=4,
15                rich={
16                    "a": {"color": "#999", "lineHeight": 22, "align": "center"},
17                    "abg": {"backgroundColor": "#e3e3e3",
18                        "width": "100%", "align": "right", "height": 22,
19                        "borderRadius": [4, 4, 0, 0],
20                    },
21                    "hr": {
22                        "borderColor": "#aaa", "width": "100%", "borderWidth": 0.5, "height": 0,
23                    },
24                    "b": {"fontSize": 16, "lineHeight": 33},
25                    "per": {
26                        "color": "#eee", "backgroundColor": "#334455",
27                        "padding": [2, 4], "borderRadius": 2,
28                    },
29                },
30            ),
31        )
32        .set_global_opts(title_opts=opts.TitleOpts(title="全国（除湖北）确诊人数Top10"), legend_opts=opts.LegendOpts(
33            pos_top="10%", pos_left="0%", orient="vertical"))
34    )
35    return c

```

图 4-14 全国（除湖北）确诊人数 Top10 图代码

全国（除湖北）确诊人数Top10

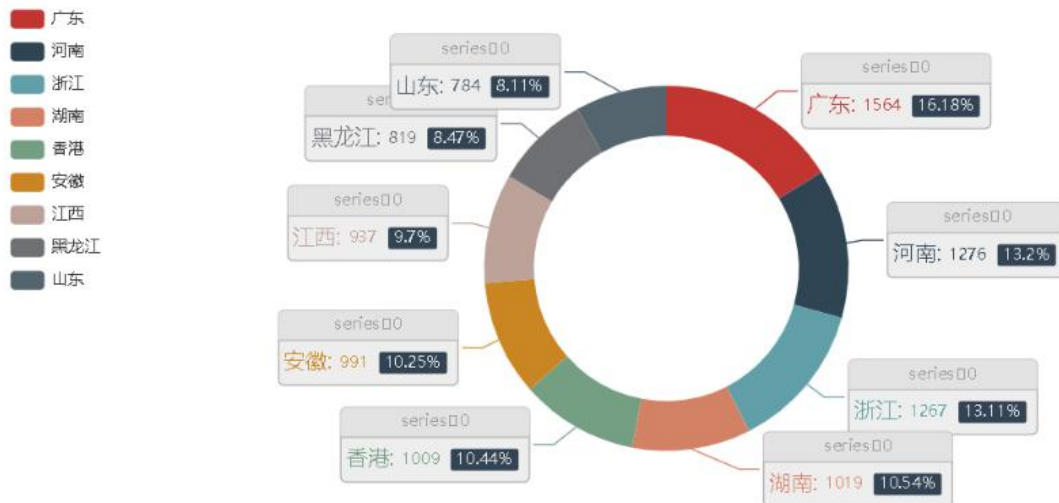


图 4-15 全国（除湖北）确诊人数 Top10 图

4.3.4 全球疫情地图可视化分析

首先我们通过数据获取到世界疫情的数据（手动添加中国的数据）

```
1 import time
2 import json
3 import csv
4 import requests
5 from datetime import datetime
6 import pandas as pd
7 data = requests.get('https://api.inews.qq.com/newsqa/v1/automation/foreign/country/ranklist')
8 data=data.json()
9 # 生成更新日期
10 update_date = date.today()
11 global_data = pd.DataFrame(data['data'])
12 global_data['confirm'] = global_data['confirm']
13 global_data['suspect'] = global_data['suspect']
14 global_data['dead'] = global_data['dead']
15 global_data['heal'] = global_data['heal']
16 global_data['addconfirm'] = global_data['confirmAdd']
17 global_data['name'] = global_data['name']
18 world_name = pd.read_excel("世界各国中英文对照.xlsx")
19 global_data = pd.merge(global_data, world_name, left_on="name", right_on="中文", how="inner")
20 global_data = global_data[["name", "英文", "confirm", "suspect", "dead", "heal", "addconfirm"]]
21 global_data=pd.DataFrame(global_data)
22 global_data.loc[155]='中国', 'China', '83696', '72', '3351', '77262', '99']
```

图 4-16 获取到世界疫情的数据

接着根据获取到的数据以地图的形式做成可视化图表，代码如图 4-17 所示，可视化图表如图 4-18 所示

```
1 from pyecharts.charts import *
2 from pyecharts import options as opts
3 from pyecharts.globals import ThemeType
4 import datetime
5 world_map = (
6     Map(init_opts=opts.InitOpts(theme='dark', width='400'))
7     .add("", [list(z) for z in zip(list(global_data["英文"]), list(global_data["confirm"]))], "world",
8           is_map_symbol_show=False)
9     .set_series_opts(label_opts=opts.LabelOpts(is_show=False),
10                     toolbox_opts=opts.ToolboxOpts(orient='vertical', pos_right="10%"))
11     .set_global_opts(title_opts=opts.TitleOpts(title="新型冠状病毒全球疫情地图",
12                                                subtitle="更新日期: {}".format(update_date)),
13                     visualmap_opts=opts.VisualMapOpts(is_piecewise=True, background_color="transparent",
14                                                         textstyle_opts=opts.TextStyleOpts(color="#F66FFA")),
15                     pieces=[
16                         {"max":0, "label": "0人", "color": "#FFFFFF"},
17                         {"min":1, "max":9, "label": "1-9人", "color": "#FFEB3D"},
18                         {"min":10, "max":99, "label": "10-99人", "color": "#FFA07A"},
19                         {"min":100, "max":499, "label": "100-499人", "color": "#FF7F50"},
20                         {"min":500, "max":999, "label": "500-999人", "color": "#CD4F39"},
21                         {"min":1000, "max":10000, "label": "1000-10000人", "color": "#CD3333"},
22                         {"min":10000, "label": ">10000人", "color": "#8B0000"} #不指定 max, 表示 max 为无限大
16                     ])
23     ))
24 world_map.render_notebook()
```

图 4-17

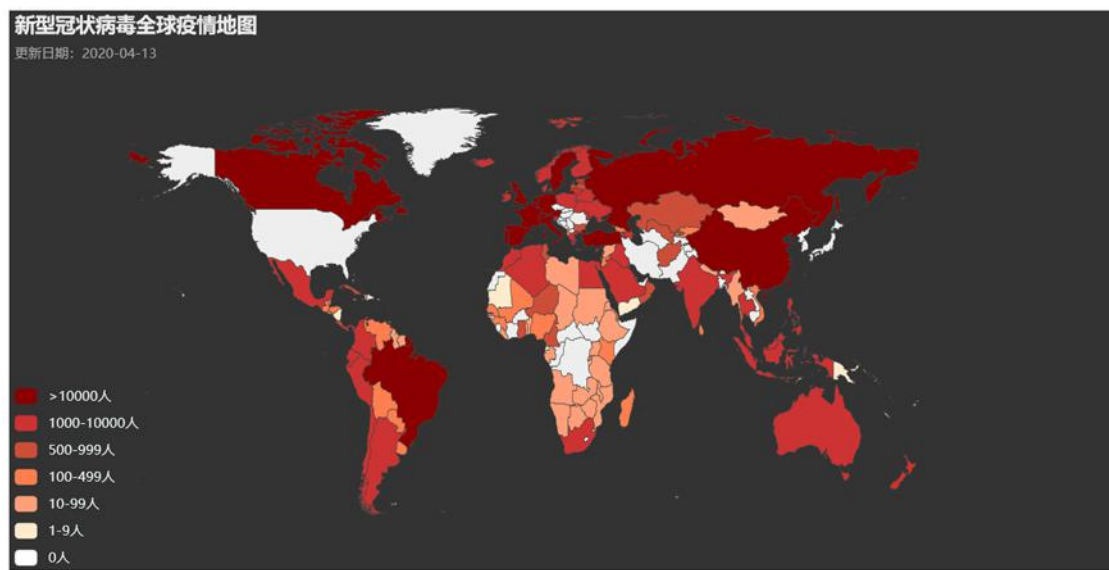


图 4-18

5 小结

本项目通过 Python 进行项目数据爬取，对文本数据进行预处理，最后使用 Pyecharts 对数据进行可视化分析。