

北京邮电大学



Python 爬虫实验

姓名：_____江姝潼_____

班级：_____2019211314_____

学号：_____2019211653_____

学院：_____计算机学院_____

专业：_____网络工程_____

目录

一、	实验要求	3
二、	实验准备	错误! 未定义书签。
1、	安装 scrapy 库	错误! 未定义书签。
2、	项目构建	错误! 未定义书签。
3、	Scrapy 库执行流程	错误! 未定义书签。
三、	爬取链家	错误! 未定义书签。
1、	Item.py	4
2、	spider.py	错误! 未定义书签。
3、	pipelines.py	错误! 未定义书签。
4、	settings.py	错误! 未定义书签。
5、	运行结果	错误! 未定义书签。
四、	爬取学堂在线	错误! 未定义书签。
1、	实验准备	12
2、	Item.py	14
3、	spider.py	16
4、	pipelines.py	18
5、	settings.py	20
6、	运行结果	22
五、	实验感想	错误! 未定义书签。
六、	附录：代码	错误! 未定义书签。
1、	爬取链家代码	29
2、	爬取学堂在线代码	32

一、 实验要求

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 2021 年 12 月 5 日开始的连续 15 天的数据，国家数不少于 100 个。

1. 标明数据来源：包括网址和首页截图
2. 数据分析和展示应包括：
 - 1) 15 天中，全球新冠疫情的总体变化趋势；
 - 2) 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；
 - 3) 累计确诊数排名前 10 的国家名称及其数量；
 - 4) 用饼图展示各个国家的累计确诊人数的比例；
 - 5) 累计确诊人数占国家总人口比例最高的 10 个国家；
 - 6) 疫苗接种情况（至少接种了一针及以上），请用地图形式展示；
 - 7) 疫苗接种率（累计疫苗接种人数/国家人数）最低的 10 个国家；
 - 8) 全球 GDP 前十名国家的累计确诊人数箱型图，要有平均值；
 - 9) 死亡率最高的 10 个国家；
 - 10) 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。

二、 爬取实验数据

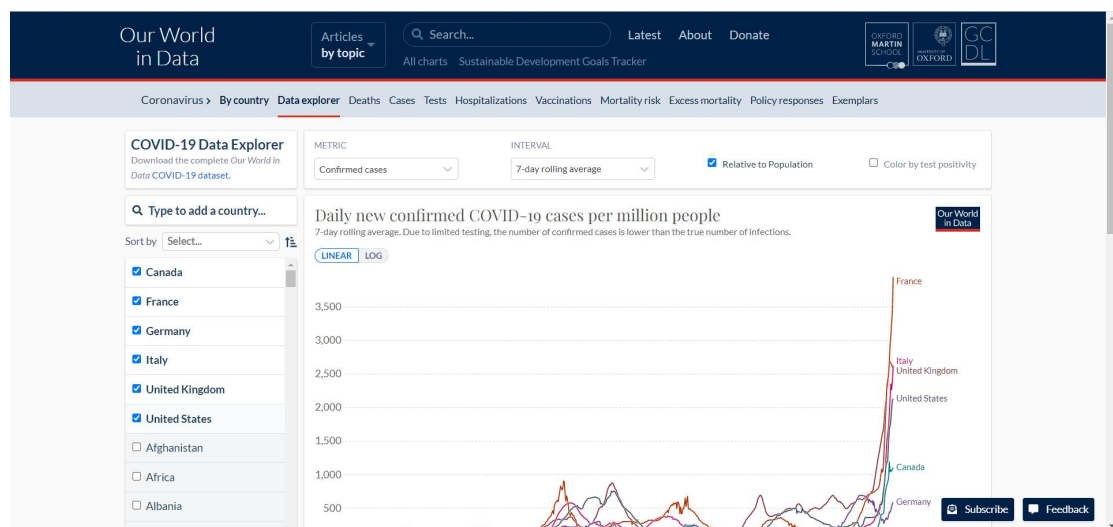
1、数据来源

经过查找，找到了 OurWorldinData 网站，该网站是由英国牛津大学的 Leszeli 创建的网站，他致力于研究几十年来各国关于人类生活水平的数据，以揭示全世界的生活状况是如何潜移默化地发生变化的，并对未来产生了什么影响，网站上展示了当今世界各方面的数据。

网址为：<https://ourworldindata.org/>

我们要爬取的是关于新冠疫情的数据，故进入 COVID-19 数据页面，url 为：
<https://ourworldindata.org/explorers/coronavirus-data-explorer>

其首页如图：



2、 爬取数据

2.1 items.py

首先需要在 items.py 这个文件中定义需要爬取的数据变量，这里根据需求定义了国家名、日期、新增病例数、确诊人数、每百万的确诊人数、注射疫苗人数、接种超过一针的人数以及死亡人数等变量，便于后续的数据处理。统一存在 NcovspiderItem 这个类里。

```

import scrapy

class NcovspiderItem(scrapy.Item):
    # define the fields for your item here like:
    country = scrapy.Field()           #表示国家名
    time = scrapy.Field()              #表示爬取数据的时间
    new_cases = scrapy.Field()         #新增病例数
    confirmed_cases = scrapy.Field()   #确诊人数
    confirmed_cases_per_million = scrapy.Field() #每百万的确诊人数
    people_vaccinated = scrapy.Field() #注射疫苗的人数
    people_fully_vaccinated = scrapy.Field() #接种超过一针的人数
    confirmed_deaths = scrapy.Field()  #死亡人数
    pass

```

2.2 ourworld.py

下面编写爬虫过程的主要代码，在 spiders 文件夹下创建一个新文件

ourworld.py:

首先定义爬虫的三个强制属性:

name = "" : 这个爬虫的识别名称，必须唯一，这里定义为 ourworld

allow_domains = [] 是搜索的域名范围，也就是爬虫的约束区域，规定爬虫只爬取这个域名下的网页，本题中为“ourworldindata.org”。

start_urls = () : 为要爬取的 URL 列表。爬虫从这里开始抓取数据，这里定义了新增病例数、确诊人数、注射疫苗人数、接种超过一针的人数以及死亡人数等的网页。

```

class OurworldSpider(scrapy.Spider):
    name = 'ourworld'
    allowed_domains = ['ourworldindata.org']
    start_urls = [
        'https://ourworldindata.org/explorers/coronavirus-data-explorer?tab=table&zoomToSelection=true&facet=none&uniformYAxis=0&pickerSort=asc&pickerMetric=total_cases&Metric=Confirmed+cases&Interval=New+per+day&Relative+to+Population=false&Color+by+test+positivity=false&time=',
        'https://ourworldindata.org/explorers/coronavirus-data-explorer?tab=table&facet=none&Interval=Cumulative&Relative+to+Population=false&Color+by+test+positivity=false&Metric=Confirmed+cases&time=earliest...',
    ]

```

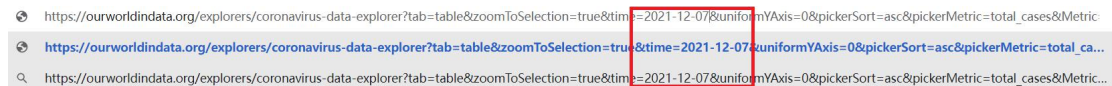
```

        'https://ourworldindata.org/explorers/coronavirus-data-explorer?tab=table&facet=none&Interval=Cumulative&Relative+to+Population=true&Color+by+test+positivity=false&Metric=Confirmed+cases&time=earliest..',
        'https://ourworldindata.org/explorers/coronavirus-data-explorer?tab=table&facet=none&Metric=People+vaccinated&Interval=Cumulative&Relative+to+Population=true&Color+by+test+positivity=false&time=earliest..',
        'https://ourworldindata.org/explorers/coronavirus-data-explorer?tab=table&facet=none&Metric=People+fully+vaccinated&Interval=Cumulative&Relative+to+Population=true&Color+by+test+positivity=false&time=earliest..',
        'https://ourworldindata.org/explorers/coronavirus-data-explorer?tab=table&facet=none&uniformYAxis=0&Metric=Confirmed+deaths&Interval=Cumulative&Relative+to+Population=false&Color+by+test+positivity=false&time=earliest..',
    ]

```

确定好参数后，下面来编写相关的操作函数，首先编写 `start_requests` 函数，更新每次提交的 url：

观察 ourworldindata 的，可以发现在其中 “time=” 后面定义的是日期，所以可以通过更改 url 上的数字连续的爬取从 12 月 5 号到 12 月 25 号的数据。



这里使用的方法是定义 `starttime` 为 5，`endtime` 为 20，从 `start time` 和 `end time` 中不断的循环，在每一个循环中，都在以上的六个 url 中加上日期并提交爬取要求。这里使用了 `cb_kwargs` 参数，可以直接在构造 `Request` 时指定一个 dict 对象，然后可以直接向 `cb_kwargs` 中添加其他参数，这些参数会被逐一追加到后续回调函数的参数列表中，便于后续 `parse` 的处理：

具体代码实现如下：

```

def start_requests(self):
    self.logger.info("Start requests")
    starttime = 5
    endtime = 20
    while starttime <= endtime:
        yield scrapy.Request(self.start_urls[0] + "2022-01-" +
                               (str)starttime, callback=self.parse, cb_kwargs=dict(date="2022-01-" +

```

```

(str)starttime, type="new_cases"))
        yield scrapy.Request(self.start_urls[1] + "2022-01-" +
(str)starttime, callback=self.parse, cb_kwargs=dict(date="2022-01-" +
(str)starttime, type="confirmed_cases"))
        yield scrapy.Request(self.start_urls[2] + "2022-01-" +
(str)starttime, callback=self.parse, cb_kwargs=dict(date="2022-01-" +
(str)starttime, type="confirmed_cases_per_million"))
        yield scrapy.Request(self.start_urls[3] + "2022-01-" +
(str)starttime, callback=self.parse, cb_kwargs=dict(date="2022-01-" +
(str)starttime, type="people_vaccinated"))
        yield scrapy.Request(self.start_urls[4] + "2022-01-" +
(str)starttime, callback=self.parse, cb_kwargs=dict(date="2022-01-" +
(str)starttime, type="people_fully_vaccinated"))
        yield scrapy.Request(self.start_urls[5] + "2022-01-" +
(str)starttime, callback=self.parse, cb_kwargs=dict(date="2022-01-" +
(str)starttime, type="confirmed_deaths"))
        starttime += 1

```

下面编写对爬取的 response 进行信息获取操作，即编写 parse 函数。

先在网页中找到要爬取的数据的位置，并对应地复制它的 xpath，对比多个 url 可以发现，网页的结构大致是相似的，所以数据所在的 xpath 也是固定的，为“/html/body/main/div/div[3]/div/div[1]/div/table/tbody/*”，来遍历表格的每一行。通过上述 kwargs 返回的字典数据，找到对应的 key 就可以获取了，并保存在 item 中。

值得注意的是，此处有一个地方需要特殊处理，就是在爬取新病例的数量和死亡的数量中发现，与其他网页都只有两列（左边是国家，右边是数据）的结构不同，这两项数据显示在同一页的（如图 2 所示），所以需要特判。如果是 new cases，查看 xpath 可以得到此时的 td 中数字为 2，其他情况下均为 3。

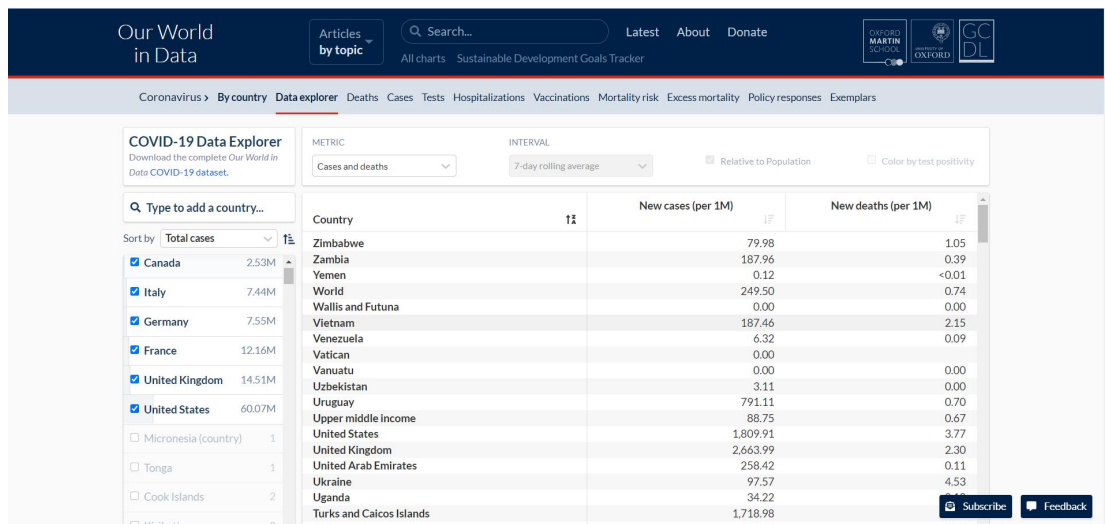


图 1 new cases and new deaths 的数据

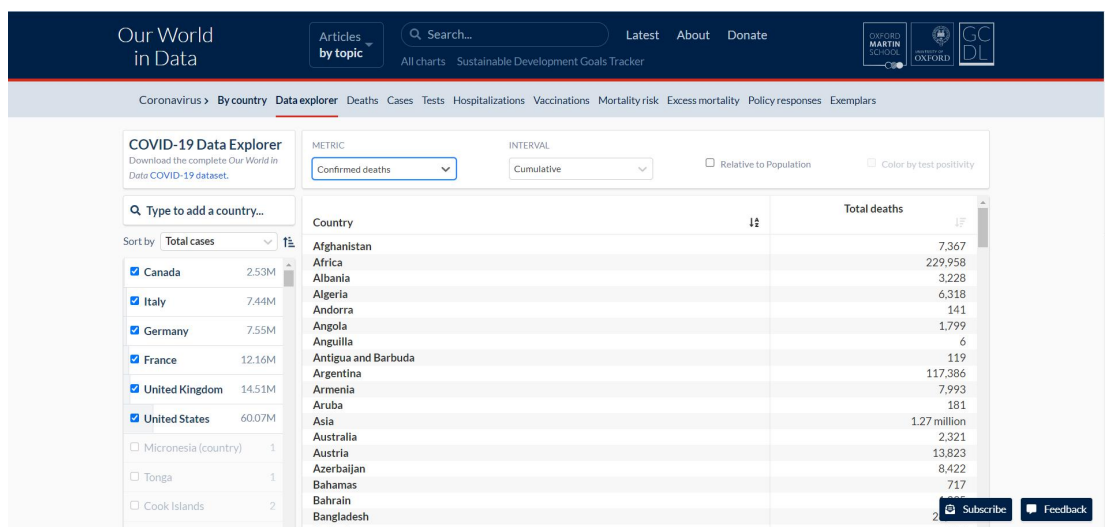


图 2 大多数数据的页面

将上述信息保存在之前定义的 `N covspiderItem` 类中，下一步提交给 pipeline 处理。

```
def parse(self, response, *args, **kwargs):
    item = NcovspiderItem()
    for keyword in keywordlist:
        item[keyword] = ""
    col = 2
    if (kwargs["type"] != "new_cases"):
        col = 3

    item['time'] = kwargs['date']
    for row in response.xpath("/html/body/main/div/div[3]/div/div[1]/div/table/tbody/*"):

```



```

        item['country'] = row.xpath("./td[1]/text()").get()
        item[kwargs['type']] =
row.xpath('./td[{}]/text()'.format(col)).get()
        yield item
        item[kwargs["type"]] = "" #方便 for 循环的后续处理

```

2.3 pipelines.py

在 pipelines.py 文件中定义对数据的处理方法，此处定义 open_spider、process_item 和 close_spider 三个函数。

在本实验中，需要将爬取的数据存在 csv 文件中，故以 dataframe 的方式存储，并定义数据列为 'country'，'time'，'new_cases'，'confirmed_cases'，'confirmed_cases_per_million'，'people_vaccinated'，'people_fully_vaccinated'，'confirmed_deaths'。

在 pipelines 中对要写入 CSV 的文件进行一个初步的处理，去除一些不好的数据，例如爬出来的数据中存在有空值的情况，便将空值删去。在写回 CSV 文件中，用 sort_values 函数，让其按照优先级从高到低为 country 和 time 的顺序进行排序，对数据做了初步的处理，让结果更为清晰直观。

Pipelines.py 的具体代码如下：

```

class NcovspiderPipeline:
    def open_spider(self, spider):
        self.df = pd.DataFrame(columns=['country', 'time',
'new_cases', 'confirmed_cases', 'confirmed_cases_per_million',
'people_vaccinated', 'people_fully_vaccinated', 'confirmed_deaths'])

    def process_item(self, item, spider):
        dict_item = ItemAdapter(item).asdict()
        if (self.df[(self.df['country'] == dict_item['country']) &
(self.df['time'] == dict_item['time'])].empty):
            self.df = self.df.append(dict_item, ignore_index=True)
        else:
            index = self.df[(self.df['country'] ==
dict_item['country']) & (self.df['time'] ==
dict_item['time'])].index[0]
            for (key, value) in dict_item.items():
                #这里对空值进行处理

```

```

        if value != "":
            self.df.at[index, key] = value
        return item

def close_spider(self, spider):
    self.df.sort_values(by=['country', 'time'], inplace=True)
    self.df.to_csv('data.csv', index=False)

```

2.4 middlewares.py

下面在 middlewares.py 文件中做中间件的处理。这里主要修改 process_request 函数的内容。经和同学讨论发现，该网页有的时候会遇到反爬的措施，所以这里需要用 selenium 去模拟操作。这里使用了 webdriver.Chrome 来打开谷歌浏览器，并设置 “--headless” 为默认不打开页面，且在每爬取一次页面之后都 sleep 5 秒来防止网页的反爬。上述操作虽然降低了爬虫的速度和效率，但是成功的爬取了所有的数据。

```

def process_request(self, request, spider):
    # Called for each request that goes through the downloader
    # middleware.

    option = webdriver.ChromeOptions()
    option.add_argument('--headless')
    option.add_experimental_option('excludeSwitches', ['enable-
automation'])
    driver = webdriver.Chrome(chrome_options=option)
    driver.get(request.url)
    # 停 5s, 防止网页的反爬
    sleep(5)
    html = driver.page_source
    driver.quit()
    return HtmlResponse(url=request.url, body=html, request=request,
encoding='utf-8')

# Must either:
# - return None: continue processing this request
# - or return a Response object
# - or return a Request object
# - or raise IgnoreRequest: process_exception() methods of
#   installed downloader middleware will be called
return None

```

2.5 settings.py

最后一步，在 `settings.py` 中设置刚刚写的中间件和 `pipelines` 即可，代码如下：

```
DOWNLOADER_MIDDLEWARES = {
    'ncovspider.middlewares.NcovspiderDownloaderMiddleware': 543,
}

# Enable or disable extensions
# See https://docs.scrapy.org/en/latest/topics/extensions.html
EXTENSIONS = {
    'scrapy.extensions.telnet.TelnetConsole': None,
}

# Configure item pipelines
# See https://docs.scrapy.org/en/latest/topics/item-pipeline.html
ITEM_PIPELINES = {
    'ncovspider.pipelines.NcovspiderPipeline': 300,
}
```

3、运行结果

运行爬虫，将结果保存到 data.csv 文件中，打开 CSV 文件部分，结果展示如下。

可以看到，上述爬虫基本实现了爬取的功能，完成了数据的爬取，并以国家和时间的顺序排列，但是只是非常粗略的，例如在确诊的病例数中，这里这里面有数字的形式，也有用 million 和 billion 为单位的形式，原因是原网页中的数据就是这么展示的，所以这里在后续数据处理中需要做进一步的处理。

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
1	Country	Time	new_cases confirmed deaths people_ku confirmed_deaths																				
2	Afghanistan	2021/12/5		157,454	3,952.61	8.86%	8.00%	7,312															
3	Afghanistan	2021/12/6	45	157,499	3,953.74	8.86%	8.00%	7,316															
4	Afghanistan	2021/12/7	9	157,508	3,953.97	8.86%	8.00%	7,317															
5	Afghanistan	2021/12/8	34	157,542	3,954.82	8.86%	8.00%	7,319															
6	Afghanistan	2021/12/9	43	157,585	3,955.90	8.86%	8.00%	7,321															
7	Afghanistan	2021/12/10	18	157,603	3,956.35	8.86%	8.00%	7,324															
8	Afghanistan	2021/12/11	8	157,611	3,956.55	8.86%	8.00%	7,325															
9	Afghanistan	2021/12/12	22	157,633	3,957.11	8.86%	8.00%	7,328															
10	Afghanistan	2021/12/13	15	157,648	3,957.48	8.86%	8.00%	7,328															
11	Afghanistan	2021/12/14	12	157,660	3,957.78	8.86%	8.00%	7,329															
12	Afghanistan	2021/12/15	5	157,665	3,957.91	8.86%	8.00%	7,331															
13	Afghanistan	2021/12/16	60	157,725	3,959.41	8.86%	8.00%	7,332															
14	Afghanistan	2021/12/17	9	157,734	3,959.64	8.86%	8.00%	7,332															
15	Afghanistan	2021/12/18	11	157,745	3,959.92	8.86%	8.00%	7,333															
16	Afghanistan	2021/12/19	42	157,787	3,960.97	8.86%	8.00%	7,335															
17	Afghanistan	2021/12/20	10	157,797	3,961.22	8.86%	8.00%	7,335															
18	Africa	2021/12/5		8.74 million	6,362.47	11.54%	7.66%	223,492															
19	Africa	2021/12/6		11,032.875 million	6,370.50	11.74%	7.73%	223,635															
20	Africa	2021/12/7		20,150.877 million	6,385.17	11.78%	7.74%	223,763															
21	Africa	2021/12/8		33,732.880 million	6,409.73	11.92%	7.82%	223,923															
22	Africa	2021/12/9		32,459.884 million	6,433.37	12.25%	7.93%	224,032															
23	Africa	2021/12/10		27,165.886 million	6,453.14	12.28%	7.96%	224,145															
24	Africa	2021/12/11		32,984.890 million	6,477.16	12.41%	8.03%	224,299															
25	Africa	2021/12/12		45,379.894 million	6,510.20	12.49%	8.06%	224,409															
26	Africa	2021/12/13		17,062.896 million	6,522.62	12.53%	8.10%	224,485															
27	Africa	2021/12/14		38,980.900 million	6,551.00	12.64%	8.17%	224,684															
28	Africa	2021/12/15		44,313.904 million	6,583.26	12.88%	8.37%	224,890															
29	Africa	2021/12/16		14,222.000 million	6,615.53	13.03%	8.40%	225,073															

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
673	Chad	2021/12/20	0	5.703	337.16	1.69%	0.48%	181															
674	Chile	2021/12/5		1.77	millio	92.260.75	87.99%	84.46%	38.501														
675	Chile	2021/12/6		1.501	1.77	millio	92.338.88	88.97%	84.58%	38.531													
676	Chile	2021/12/7		1.164	1.78	millio	92.399.47	88.16%	84.70%	38.535													
677	Chile	2021/12/8		1.387	1.78	millio	92.471.66	88.16%	84.70%	38.541													
678	Chile	2021/12/9		1.771	1.78	millio	92.563.84	88.36%	84.83%	38.600													
679	Chile	2021/12/10		0	1.78	millio	92.563.84	88.56%	84.93%	38.600													
680	Chile	2021/12/11		2.857	1.78	millio	92.712.55	88.58%	84.95%	38.646													
681	Chile	2021/12/12		1.577	1.78	millio	92.794.63	88.58%	84.95%	38.686													
682	Chile	2021/12/13		1.361	1.78	millio	92.885.47	88.75%	85.07%	38.716													
683	Chile	2021/12/14		959	1.79	millio	92.915.38	88.91%	85.18%	38.723													
684	Chile	2021/12/15		1.060	1.79	millio	92.970.56	89.05%	85.28%	38.727													
685	Chile	2021/12/16		1.492	1.79	millio	93.048.22	89.18%	85.36%	38.773													
686	Chile	2021/12/17		1.433	1.79	millio	93.122.80	89.30%	85.44%	38.812													
687	Chile	2021/12/18		1.415	1.79	millio	93.196.45	89.31%	85.45%	38.840													
688	Chile	2021/12/19		1.252	1.79	millio	93.261.62	89.31%	85.45%	38.864													
689	Chile	2021/12/20		1.126	1.79	millio	93.320.23	89.42%	85.53%	38.885													
690	China	2021/12/5		99.203	68.69	84.82%	77.90%	4.636															
691	China	2021/12/6	94	99.297	68.75	84.82%	77.90%	4.636															
692	China	2021/12/7	74	99.371	68.81	84.82%	77.90%	4.636															
693	China	2021/12/8	83	99.454	68.86	84.82%	77.90%	4.636															
694	China	2021/12/9	63	99.517	68.91	84.82%	77.90%	4.636															
695	China	2021/12/10	87	99.604	68.97	84.82%	80.49%	4.636															
696	China	2021/12/11	75	99.679	69.02	84.82%	80.49%	4.636															
697	China	2021/12/12	101	99.780	69.09	84.82%	80.49%	4.636															
698	China	2021/12/13	76	99.856	69.14	84.82%	80.49%	4.636															
699	China	2021/12/14	69	99.925	69.19	84.82%	80.49%	4.636															
700	China	2021/12/15	75	100.000	69.24	84.82%	80.49%	4.636															
701	China	2021/12/16	76	100.076	69.29	84.82%	80.49%	4.636															
702	China	2021/12/17	125	100.201	69.38	84.82%	82.14%	4.636															
703	China	2021/12/18	83	100.284	69.44	84.82%	82.14%	4.636															
704	China	2021/12/19	102	100.386	69.51	84.82%	82.64%	4.636															
705	China	2021/12/20	81	100.467	69.56	84.82%	82.64%	4.636															
706	Colombia	2021/12/5		5.08	millio	99.112.08	58.85%	49.49%	128.780														
707	Colombia	2021/12/6		1.698	5.08	millio	99.145.20	58.85%	49.78%	128.821													
708	Colombia	2021/12/7		1.704	5.08	millio	99.178.44	58.85%	50.03%	128.874													
709	Colombia	2021/12/8		1.915	5.09	millio	99.215.79	58.85%	50.03%	128.874													
710	Colombia	2021/12/9		1.627	5.09	millio	99.247.53	58.85%	50.45%	128.969													
711	Colombia	2021/12/10		1.687	5.09	millio	99.280.43	58.85%	50.80%	129.011													
712	Colombia	2021/12/11		1.813	5.09	millio	99.315.80	58.85%	51.04%	129.056													
713	Colombia	2021/12/12		2.036	5.09	millio	99.355.33	58.85%	51.10%	129.107													

三、 数据处理

1、 总体变化趋势

首先要观察这 15 天来全球的总体病例数的变化趋势。在 CSV 中可以看到，当 country 为 world 的时候是表示全球的数据，所以只需要通过 loc 函数提取出为 world 的项目，然后并进行处理和绘制曲线图就可以了，选用曲线图是因为曲线图可以很好地反映出趋势的变化。但是观察 CSV 可以知道，里面的数据是通过逗号来分隔开的，所以这里要对逗号进行一个处理，把逗号给去掉并返回成数据的值即可，处理代码如下：

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from pylab import mpl

plt.rcParams['font.sans-serif'] = ['SimHei']

def num_transform(data):
    y = str(data)
```

```

y = y.replace(',', '')
if y.isdigit():
    return int(y)
else:
    return 0

df = pd.read_csv('data.csv')

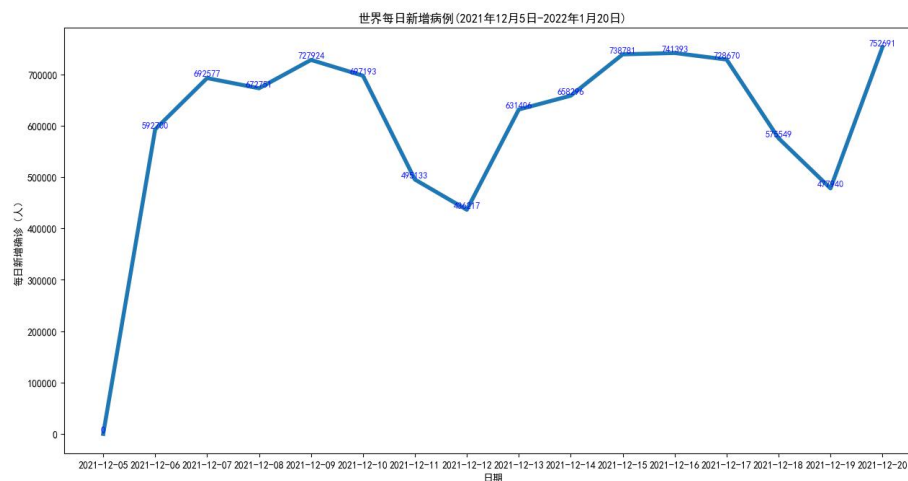
Y =
df.loc[df['country']=='World']['new_cases'].apply(num_transfor
m)
X = df.loc[df['country']=='World']['time']

for a,b in zip(X,Y):
    plt.text(a, b, b,
             ha='center',va='bottom',fontsize=9,color='b',alpha=0.9)
plt.plot(X, Y, linewidth=4)
plt.xlabel("日期")
plt.ylabel("每日新增确诊（人）")
plt.title('世界每日新增病例(2021年12月5日-2022年1月20日)')

plt.show()

```

运行出的结果如下：



观察趋势可以发现，在12月5日处有一个突然的暴增，查看原数据发现此处的数据是空，故不具备参考价值。看剩下几天的变化趋势，可以发现基本是以七

天的为一个周期的规律，先递增再逐渐的递减，猜想可能是因为周六日人群的频繁外出，使得疫情有进一步的传播，从而导致数量激增。

46	World	2021/12/5		265.89 mil	33,763.59	55.18%	44.66%	5.26 million
47	World	2021/12/6	592,700	266.48 mil	33,838.86	55.28%	44.77%	5.26 million
48	World	2021/12/7	692,577	267.17 mil	33,926.80	55.35%	44.90%	5.27 million
49	World	2021/12/8	672,751	267.85 mil	34,012.23	55.47%	45.07%	5.28 million
50	World	2021/12/9	727,924	268.57 mil	34,104.67	55.61%	45.28%	5.29 million
51	World	2021/12/10	697,193	269.27 mil	34,193.20	55.92%	45.86%	5.30 million
52	World	2021/12/11	495,133	269.77 mil	34,256.07	56.08%	46.09%	5.30 million
53	World	2021/12/12	436,217	270.20 mil	34,311.47	56.16%	46.25%	5.31 million
54	World	2021/12/13	631,406	270.83 mil	34,391.65	56.21%	46.37%	5.31 million
55	World	2021/12/14	658,296	271.49 mil	34,475.24	56.33%	46.55%	5.32 million
56	World	2021/12/15	738,781	272.23 mil	34,569.05	56.44%	46.71%	5.33 million
57	World	2021/12/16	741,393	272.97 mil	34,663.20	56.50%	46.80%	5.34 million
58	World	2021/12/17	728,670	273.70 mil	34,755.73	56.61%	47.27%	5.35 million
59	World	2021/12/18	575,549	274.28 mil	34,828.82	56.77%	47.39%	5.35 million
60	World	2021/12/19	477,940	274.75 mil	34,889.51	56.85%	47.58%	5.36 million
61	World	2021/12/20	752,691	275.51 mil	34,985.09	56.97%	47.77%	5.36 million

2、每日新增确诊数据的曲线图

下面需要绘制在 15 天内累计排名前十国家的每日新增确诊数据的曲线图，这个处理比上面的要复杂一些，有几部分的要求。首先在爬取的数据中，有一部分数据并不是国家的数据，它包括了一些综合的数据，比如洲、高收入阶层等

(eg: world、Africa、high income)，把这些特殊数据全部存在

NotCountryList 字符串数组，并用 isin 函数筛选掉非国家的数据。其次需要把 15 天内的新增确诊全部求和再排序获得的前十个国家的名单，把它保存在一个 list 变量中，然后遍历列表中每一项，分别绘制十个国家的变化曲线图。和上一个处理类似，同样需要用一个 num_transform 函数把逗号给去掉，并转化为数字，同时在使用 text 函数标明数据值，再使用一个 legend 函数来标明各个图例，防止混淆，最终用 show 绘制整个图片。代码如下：

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

plt.rcParams['font.sans-serif'] = ['SimHei']

NotCountryList=['World','Africa','North America','South
America','Europe','Asia','Oceania','High income','European
```

```

Union','Upper middle income','Lower middle income','Low
income']

def num_transform(data):
    y = str(data)
    y = y.replace(',','')
    if y.isdigit():
        return int(y)
    else:
        return 0

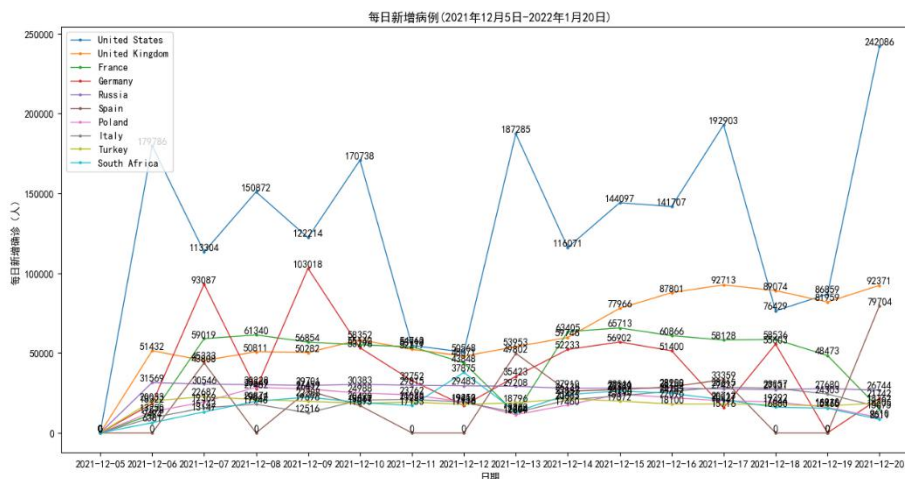
df = pd.read_csv('data.csv')
df['new_cases'] = df['new_cases'].apply(num_transform)
#筛掉不是非国家的数据
df = df.loc[df['country'].isin(NotCountryList) == False]
#选取总和为前十的国家
list =
df.groupby('country')['new_cases'].sum().sort_values(ascending
=False).head(10).index.tolist()

plt.xlabel("日期")
plt.ylabel("每日新增确诊（人）")
plt.title('2021 年 12 月 5 日-2022 年 1 月 20 日每日新增病例')
for i in list:
    Y =
df.loc[df['country']==i]['new_cases'].apply(num_transform)
    X = df.loc[df['country']==i]['time']
    plt.plot(X, Y, linewidth=1, marker='o', label=i,
markersize=2)
    for a, b in zip(X, Y):
        plt.text(a, b, b, ha='center', va='bottom',
fontsize=10)

plt.legend(list, loc='upper left', fontsize=10)
plt.show()

```

最终绘制结果如下：



可以看到，排名前十的国家的每日确诊数据变化趋势和全球总体变化趋势基本相同，都是有一个周期波动的过程。但是美国的确诊数据遥遥领先，基本是后面几个国家的基本两倍之多，可以看到美国的疫情形势还是非常严峻的。

3、 累计确诊数排名前 10 的国家名称及其数量

下面需要展示累计确诊排名前十的国家名称及数量，前面的处理和上一个实验类似，不同的是由于累计确诊数较多，有的国家的数据是以逗号的形式分隔的，而有的国家的数据后面则出现了 million，由于没有出现 billion 的情况，所以统一对百万进行处理：在去掉逗号后，如果还是为数字，那么统一除以 1000000，把单位统为统一化成百万人，然后再进行排序，选取出排名前十的国家，并且将前十的国家以柱状图的形式绘制，代码如下：

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

plt.rcParams['font.sans-serif'] = ['SimHei']

NotCountryList=['World','Africa','North America','South America','Europe','Asia','Oceania','High income','European Union','Upper middle income','Lower middle income','Low income']

def num_transform(data):
```



```

    y = str(data)
    y = y.replace(',', '')
    if y.isdigit():
        return int(y) / 1000000
    elif len(y) > 0:
        return float(y.split(' ')[0])
    else:
        return 0

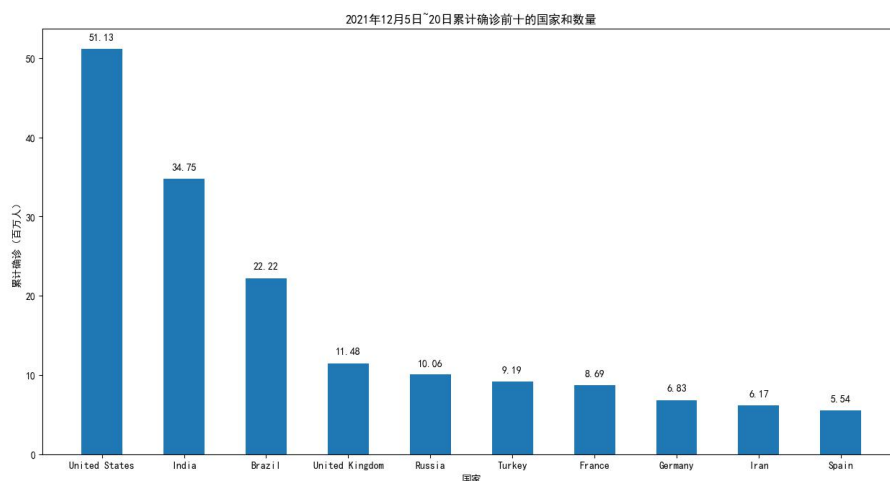
df = pd.read_csv('data.csv')
#筛掉非国家的数据
df = df.loc[df['country'].isin(NotCountryList) == False]
#选取总和为前十的国家
df['confirmed_cases'] =
df['confirmed_cases'].apply(num_transform)
list =
df.groupby('country')['confirmed_cases'].max().sort_values(asc
ending=False).head(10)

plt.xlabel("国家")
plt.ylabel("累计确诊（百万人）")
plt.title('2021 年 12 月 5 日~20 日累计确诊前十的国家和数量')
plt.bar(list.index, list.values, width=0.5)
for a, b in zip(list.index, list.values):
    plt.text(a, b+2, b, ha='center', va='top', fontsize=10)

plt.show()

```

绘制出的结果如下：



可以看到，美国、印度、巴西等国的累计确诊数据遥遥领先，英国、俄罗斯、土耳其、法国、德国等累计确诊人数相差不大。

4、累计确诊人数的比例

下面用饼图展示各个国家累计确诊人数的比例。首先删除所有非国家的数据，并把数据统一为百万为单位。这里通过排序获取累计确诊为前 25 的国家，并将其他其他的国家统一合并成“其他”项。这里用 `df1` 存储了前 25 个国家的累计确诊数，然后再用 `othernum` 存储其他所有国家的累和，最后把它加入到 `df1` 中，最终再对 `df1` 中的所有数据进行绘制，代码如下：

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

plt.rcParams['font.sans-serif'] = ['SimHei']

NotCountryList=['World','Africa','North America','South America','Europe','Asia','Oceania','High income','European Union','Upper middle income','Lower middle income','Low income']

def num_transform(data):
    y = str(data)
    y = y.replace(',', '')
    if y.isdigit():
```

```

        return int(y) / 1000000
    elif len(y) > 0:
        return float(y.split(' ')[0])
    else:
        return 0

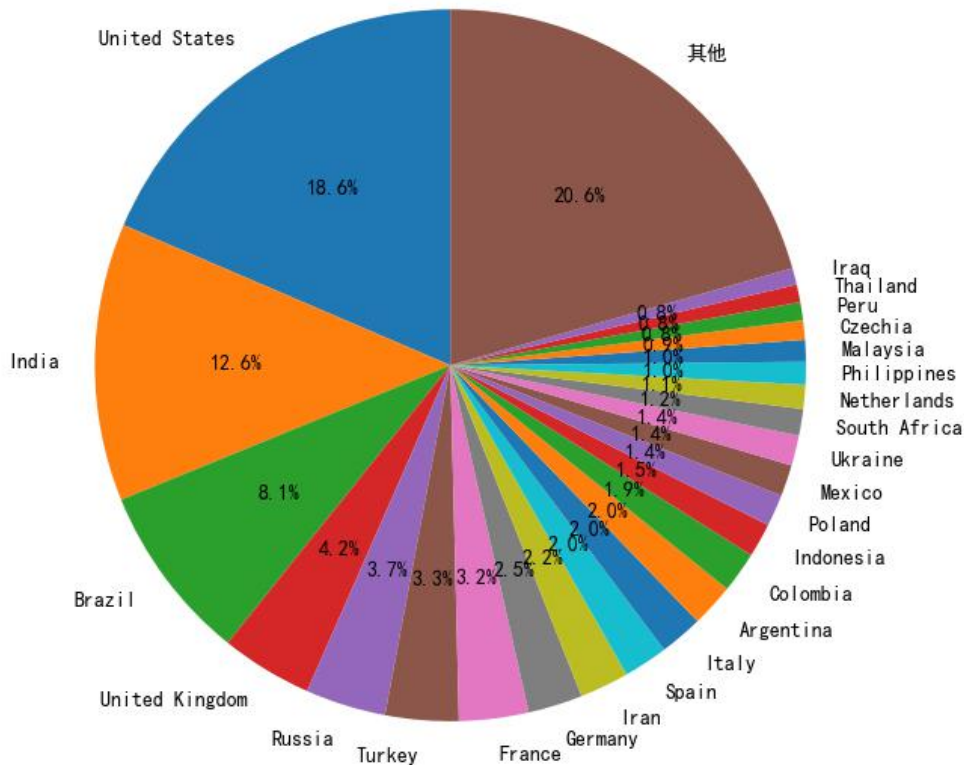
df = pd.read_csv('data.csv')
#筛掉非国家的数据
df = df.loc[df['country'].isin(NotCountryList) == False]
#对数据统一为百万为单位
df['confirmed_cases'] =
df['confirmed_cases'].apply(num_transform)
list =
df.groupby('country')['confirmed_cases'].max().sort_values(asc
ending=False).head(25).index.tolist()
#将非前 25 的所有国家合并为其他
df1 =
df.loc[df['country'].isin(list)].groupby('country')['confirmed
_cases'].max()
othernum = df.loc[df['country'].isin(list) ==
False].groupby('country')['confirmed_cases'].max().sum()
df1.sort_values(ascending=False, inplace=True)
df1 = df1.append(pd.Series(othernum, index=['其他']))
#绘制饼图
plt.pie(df1, labels=df1.index, autopct='%1.1f%%', shadow=False,
startangle=90)
plt.title('2021 年 12 月 5 日-20 日各国累计确诊比例')

plt.show()

```

最终绘制结果如下图：

2021年12月5日-20日各国累计确诊比例



结合上面分析的数据可以看到，前十的国家占了累计确诊病例的一半还多，剩下的所有国家累加起来才也占一半不到，说明疫情主要还是分布在几个大国，而在其他小国分布的是较为稀疏的，但可能也和其他小国的人口数量较少有关，故下面分析占人口比例的数据。

5、总人口比例最高的 10 个国家

下面展示累计确诊人数占国家总人口比例最高的国家，同样采用柱状图 bar 来进行显示，在 CSV 中筛除非国家的数据，并删除逗号统一数据形式。由于在爬取中的数据存在 `confirmed_cases_per_million` 这个变量，可以直接作为总人口比例前十的数据，用 `sort` 进行排并选取前十项，代码如下：

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

```

plt.rcParams['font.sans-serif'] = ['SimHei']

NotCountryList=['World','Africa','North America','South
America','Europe','Asia','Oceania','High income','European
Union','Upper middle income','Lower middle income','Low
income']

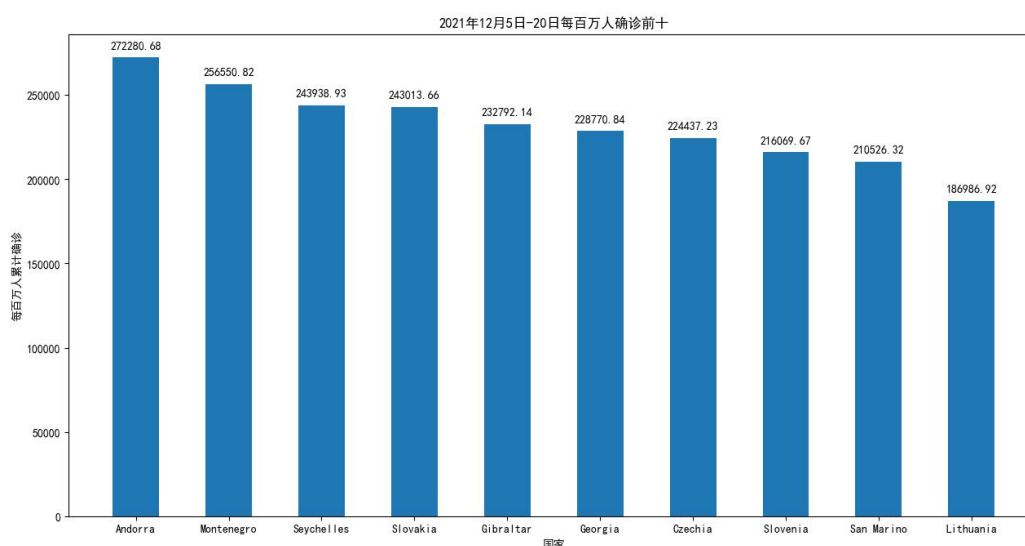
def num_transform(data):
    y = str(data)
    y = y.replace(',','')
    if len(y) > 0:
        return float(y)
    else:
        return 0

df = pd.read_csv('data.csv')
#筛掉非国家的数据
df = df.loc[df['country'].isin(NotCountryList) == False]
#删除"," 统一数据形式
df['confirmed_cases_per_million'] =
df['confirmed_cases_per_million'].apply(num_transform)
#获取占比前 10 的国家
list = df.groupby('country')
['confirmed_cases_per_million'].max().sort_values(ascending=False).head(10)
plt.xlabel("国家")
plt.ylabel("每百万人累计确诊")
plt.title('2021 年 12 月 5 日-20 日每百万人确诊前十')

plt.bar(list.index, list.values, width=0.5)
for a, b in zip(list.index, list.values):
    plt.text(a, b+10000, b, ha='center', va='top', fontsize=10)
plt.show()

```

绘制出柱状图如下：



可以看到，在前面几项分析中，如果是但凡涉及到总数据，美国都遥遥领先，但是人口占比前十并没有出现总和前十的任何一个国家，反而都是以下很小的国家。可以看出，美国虽然病例很多，但是占全国的总人数的占比并不是很高，相反，是一些小国家可能由于人口较少而确诊人数又多，从而导致比例很高。

6、疫苗接种情况

接下来以地图的形式绘制全球的疫苗接种情况。该问题和上述问题不同，原因是要以地图的形式进行展示。这里使用了一个 Python 的库，即 `pyecharts`，用它其中的 `map` 函数来绘制地图。首先根据 `people_vaccinated` 的里的百分比的数，去掉百分数并统一除以 100，这样就获得了一个 0-1 之间的疫苗接种率，再把这些数据去掉非国家的部分之后给传入 `map`，获得了一个 `6.html` 文件，代码如下：

```
from pyecharts import options as opts
from pyecharts.charts import Map
import pandas as pd

def num_transform(data):
    y = str(data)
    y = y.replace('%', '')
    return float(y) / 100

NotCountryList=['World','Africa','North America','South America','Europe','Asia','Oceania','High income','European
```

```

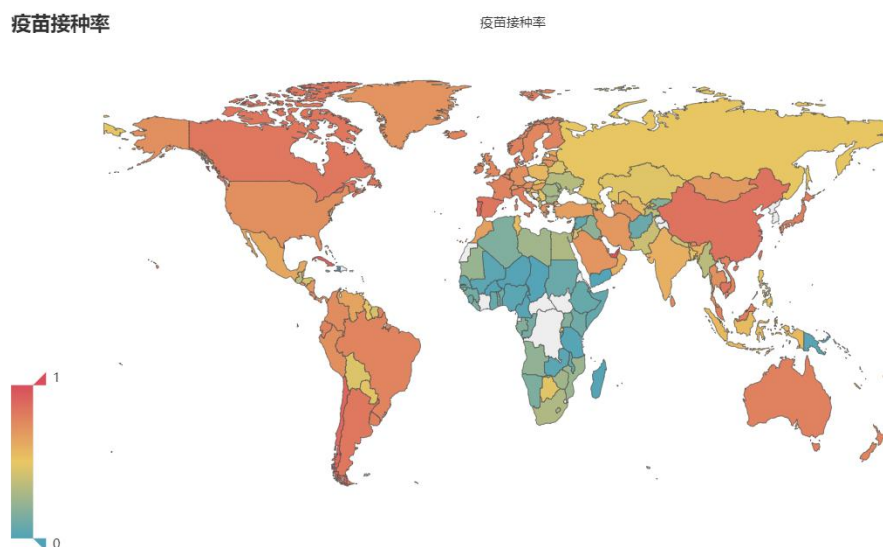
Union','Upper middle income','Lower middle income','Low
income']

df = pd.read_csv('data.csv')
#筛掉非国家的数据
df = df.loc[df['country'].isin(NotCountryList) == False]
df['people_vaccinated'] =
df['people_vaccinated'].apply(num_transform)
df = df.groupby('country')['people_vaccinated'].max()

Map().add(
    "疫苗接种率",
    [list(z) for z in zip(df.index, df.values)],
    is_map_symbol_show=False,
    maptype="world",
    label_opts=opts.LabelOpts(is_show=False),
    itemstyle_opts=opts.ItemStyleOpts(color="rgba(255,255,255,
0.5)")),
).set_series_opts(label_opts=opts.LabelOpts(is_show=False)).se
t_global_opts(
    title_opts=opts.TitleOpts(title="疫苗接种率"),
    visualmap_opts=opts.VisualMapOpts(max_=1.0),
).render("6.html")

```

打开 6.html，绘制出的地图如下：



在上述地图中，颜色深的表示疫苗接种率较高的，浅的则表示为较低。从图中可以看到，亚洲、美洲、拉丁美洲基本接种率都在一半以上，而非洲等国家则普遍接种率较少。可知疫苗接种率和国家经济实力息息相关的，经济不太发达的国家接种较少。

7、疫苗接种率

下面用柱状图展示疫苗接种率最低的十个国家，在爬取的数据中存在一项 `people_fully_vaccinated`，来展示出该国家疫苗接种的比例。统一处理掉百分号统一数据，然后再除去了非国家的项目后，在排通过排序筛选出前十，并用柱状图展示，和前面的基本过程是类似的。

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

plt.rcParams['font.sans-serif'] = ['SimHei']

NotCountryList=['World','Africa','North America','South America','Europe','Asia','Oceania','High income','European Union','Upper middle income','Lower middle income','Low income']

def num_transform(data):
    y = str(data)
    y = y.replace('%', '')
    if len(y) > 0:
        return float(y) / 100
    else:
        return 0

#筛掉非国家的数据
df = pd.read_csv('data.csv')
df = df.loc[df['country'].isin(NotCountryList) == False]
#统一百分比数据形式
df['people_fully_vaccinated'] =
df['people_fully_vaccinated'].apply(num_transform)
#取最低的五个国家
list =
df.groupby('country')['people_fully_vaccinated'].max().sort_values(ascending=True).head(10)
```

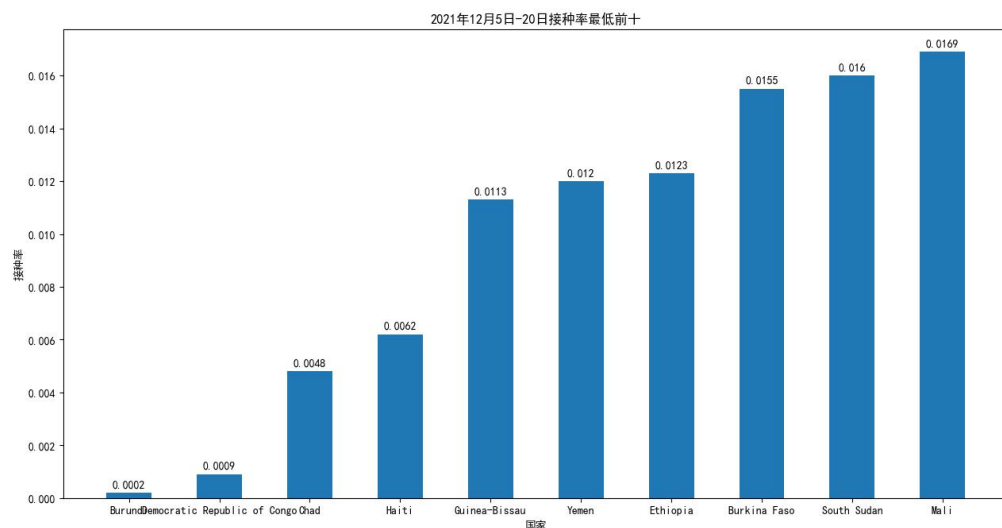


```
plt.xlabel("国家")
plt.ylabel("接种率")
plt.title('2021 年 12 月 5 日-20 日接种率最低前十')

plt.bar(list.index, list.values, width=0.5)
for a, b in zip(list.index, list.values):
    plt.text(a, b+0.0005, b, ha='center', va='top',
             fontsize=10)

plt.show()
```

绘制的柱状图如下：



可以看到，非洲的效果普遍接种率较低，都不到 0.02，可见疫苗接种率和经济实力是息息相关的。

8、累计确诊人数箱型图

下面绘制全球 GDP 前十名确诊的箱形图。首先，根据该网站可以直接查找获取 GDP 排名前十的国家，这里将 GDP 前十的国家提取出来，并存储在一个 gdpList 的变量中，从表格中读取读取信息并筛选出在 gdpList 中的国家的确诊人数，并传给 box 函数去绘制箱型图，并输出整体的平均数。

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

```

plt.rcParams['font.sans-serif'] = ['SimHei']

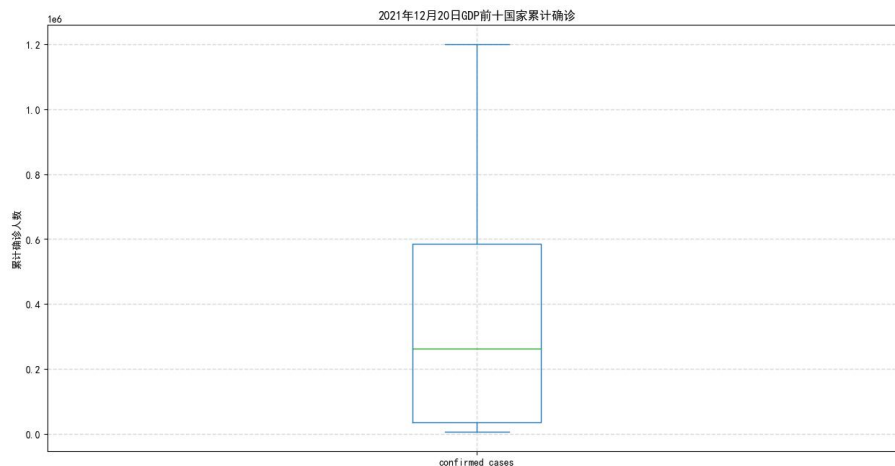
NotCountryList=['World','Africa','North America','South
America','Europe','Asia','Oceania','High income','European
Union','Upper middle income','Lower middle income','Low
income']
gdpList=['Luxembourg','Singapore','Ireland','Qatar','Bermuda',
'Cayman Islands','Switzerland','United Arab
Emirates','Norway','Brunei']

def num_transform(data):
    y = str(data)
    y = y.replace(',',' ')
    if y.isdigit():
        return int(y)
    elif len(y) > 0:
        return float(y.split(' ')[0]) * 1000000
    else:
        return 0

df = pd.read_csv('data.csv')
df = df.loc[(df['country'].isin(NotCountryList) == False) &
(df['country'].isin(gdpList))]
#统一数据形式
df['confirmed_cases'] =
df['confirmed_cases'].apply(num_transform)
#绘制箱型图
list =
df.groupby('country')['confirmed_cases'].max().sort_values(asc
ending=False)
list.plot.box(title='2021 年 12 月 20 日 GDP 前十国家累计确诊')
plt.ylabel("累计确诊人数")
plt.grid(linestyle='--', linewidth=1, color='#d8d8d8')
plt.show()
print ("平均数: ", list.mean())

```

最终绘制的箱形图如下，平均数在终端输出结果如下：



```
PS D:\大三\python程序设计\江妹潼-数据可视化\code> python -u "d:\大三\python程序设计\江妹潼-数据可视化\code\covid8.py"
平均数: 360956.5
```

9、死亡率

死亡率最高的十个国家如下所示，数据处理的流程跟上述基本相同。唯一的不同是需要读取两部分的数据，分别是确诊数和确诊死亡数，并给他们相除来计算出死亡率，再对死亡率进行排名和绘图，区别是比上面多了一步计算死亡率的过程。

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

plt.rcParams['font.sans-serif'] = ['SimHei']
NotCountryList=['World','Africa','North America','South
America','Europe','Asia','Oceania','High income','European
Union','Upper middle income','Lower middle income','Low
income']

def num_transform(data):
    y = str(data)
    y = y.replace(',', '')
    if y.isdigit():
        return int(y)
    elif not np.isnan(float(y.split(' ')[0])):
        return int(float(y.split(' ')[0]) * 1000000)
    else:
        return np.nan
```

```

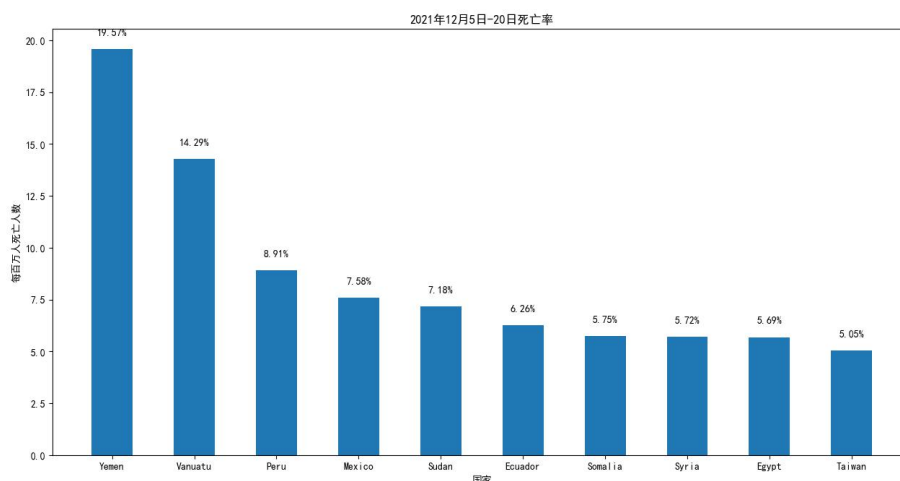
df = pd.read_csv('data.csv')
#筛掉非国家的数据
df = df.loc[df['country'].isin(NotCountryList) == False]
df['confirmed_deaths'] =
df['confirmed_deaths'].apply(num_transform)
df['confirmed_cases'] =
df['confirmed_cases'].apply(num_transform)
maxdate = df['time'].max()
df = df.loc[df['time']==maxdate]
#计算死亡率
df['confirmed_deaths_rate'] = df['confirmed_deaths'] /
df['confirmed_cases'] * 100
#取前 10 的国家
list =
df.groupby('country')['confirmed_deaths_rate'].max().sort_valu
es(ascending=False).head(10)

plt.xlabel("国家")
plt.ylabel("每百万人死亡人数")
plt.title('2021 年 12 月 5 日-20 日死亡率')
plt.bar(list.index, list.values, width=0.5)
for a, b in zip(list.index, list.values):
    plt.text(a, b+1, '{:.2f}%'.format(b), ha='center',
va='top', fontsize=10)

plt.show()

```

绘制出的死亡率最高的 10 个国家如下图：



从死亡率的表中可以看到，并没有包括在累计确诊数前十国家中的任何一个，相反是一些其他的中小国家，死亡率较高。死亡率一定程度上反映了国家对已确诊病例的处理措施，可以看到大国的处理还是相对比较好的，死亡率不是很高，相反是一些其他国家可能医疗体系已经崩溃，所以导致病人得不到救助而死亡率攀升的情况。

四、 数据分析与预测

1、全世界应对新冠疫情最好的 10 个国家

结合上述分析数据，要判断一个国家的抗疫效果如何，要从确诊率、疫苗接种率和死亡率三个方面进行衡量。首先，确诊率可以通过每百万的确诊数来获得，疫苗接种率在爬取的数据中可以直接获得，而在爬取的数据中，用死亡数除以确诊数可以获得死亡率，这样就获得了三个 0-1 之间，以比例的形式展现的参数。再人为规定一个综合参数，给予确诊率、接种率、死亡率不同的比例，这里我给他们的比例分别是 3 : 4 : 3，其中确诊率越低，接种率越高，死亡率越低，体现为抗疫效果越好。但是此时运行发现是为负数，为了方便展示，统一乘上-1 变为正数，此时综合参数越小，则证明抗疫效果越好。再排序取前十个，然后画出柱状图，代码展示如下：

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

plt.rcParams['font.sans-serif'] = ['SimHei']
```

```

NotCountryList=['World','Africa','North America','South
America','Europe','Asia','Oceania','High income','European
Union','Upper middle income','Lower middle income','Low
income']

def num_transform(data):
    y = str(data)
    y = y.replace(',',' ')
    if y.isdigit():
        return int(y)
    elif y.find('million') != -1:
        return int(float(y.split(' ')[0]) * 1000000)
    elif y.find('billion') != -1:
        return int(float(y.split(' ')[0]) * 1000000000)
    else:
        return 0

def num_transform_float(data):
    y = str(data)
    y = y.replace('%',' ')
    y = y.replace(',',' ')
    if len(y) > 0:
        return float(y)
    else:
        return 0

df = pd.read_csv('data.csv')
#筛掉非国家的数据
df = df.loc[df['country'].isin(NotCountryList) == False]
#获取所有的数据
df['confirmed_cases_per_million'] =
df['confirmed_cases_per_million'].apply(num_transform_float)
df['people_vaccinated'] =
df['people_vaccinated'].apply(num_transform_float)
df['confirmed_deaths'] =
df['confirmed_deaths'].apply(num_transform_float)
df['confirmed_cases'] =
df['confirmed_cases'].apply(num_transform)
df['new_cases'] = df['new_cases'].apply(num_transform)
#确诊率

```

```

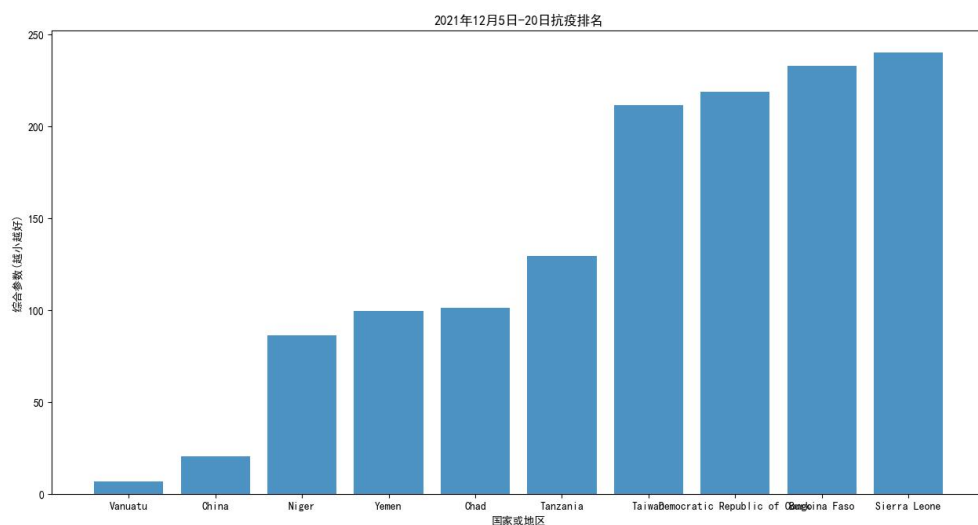
df1 =
df.groupby('country')['confirmed_cases_per_million'].max().sort_index()
#疫苗接种率
df2 =
df.groupby('country')['people_vaccinated'].max().sort_index()
/ 100
#死亡数/确诊数=死亡率
df3 =
df.groupby('country')['confirmed_deaths'].max().sort_index()
df4 =
df.groupby('country')['confirmed_cases'].max().sort_index()
df3 = df3 / df4
#确诊率越低，接种率越高，死亡率越低，抗疫效果越好,体现为数值越大越好
valuedf = (-1*df1) * 0.3 + df2 * 0.4 + (-1*df3) * 0.3
#乘以-1 变为正数，此时越小越好
valuedf *= -1

valuedf = valuedf.sort_values(ascending=True).head(10)

plt.bar(valuedf.index, valuedf.values, alpha=0.8)
plt.ylabel("综合参数(越小越好)")
plt.title("2021年12月5日-20日抗疫排名")
plt.xlabel("国家或地区")
plt.show()

```

绘制出结果如下：



从结果中可以看到，位于南太平洋西部的瓦努阿图共和国抗疫效果位列第一，其次是中国，尼泊尔，也门等国家。可以看到，中国的抗疫效果算是非常突出的，综合参数是很小的，可以体现我国的疫情防控确实非常的有效，同时亚洲地区的普疫情防控也普遍较好。

2、后 5 天的预测

在前面的分析中可以看到，在全球新确诊的病例数和排名前十各国的病例数的变化趋势中可以发现，是以一周为一个周期进行周期性变化的，所以这里预测也是使用了周期的性质。这里用 base 来存储在七天基本的平均波动，并把平均波动加在前面七天的数据上来绘制出整体的趋势图，绘制出的图片如下。

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

plt.rcParams['font.sans-serif'] = ['SimHei']

def num_transform(data):
    y = str(data)
    y = y.replace(',', '')
    if y.isdigit():
        return int(y)
    else:
        return 0

df = pd.read_csv('data.csv')
#X Y 表示原始数据
Y =
df.loc[df['country']=='World']['new_cases'].apply(num_transform).values
X = df.loc[df['country']=='World']['time']

#NewY 表示预测的数据，前 10 保持不变
NewY = []
for i in range(0, 10):
    NewY.append(Y[i])
#用 base 计算 7 天周期的波动平均值
base = 0
for i in range(0, 3):
    base += Y[i + 7] - Y[i]
```



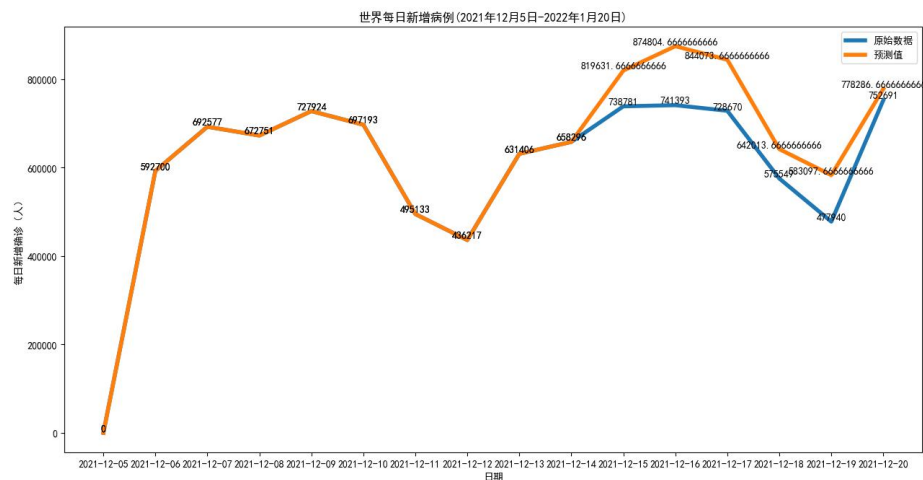
```

base = (base / 3)
#预测后 5 天的数据
for i in range(10, 16):
    NewY.append(Y[i - 7] + base)

#绘制趋势图
plt.plot(X, Y, linewidth=4)
for (a, b) in zip(X, Y):
    plt.text(a, b, b, ha='center', va='bottom', fontsize=10)
plt.plot(X, NewY, linewidth=4)
for (a, b) in zip(X, NewY):
    plt.text(a, b, b, ha='center', va='bottom', fontsize=10)
plt.legend(['原始数据', '预测值'])
plt.xlabel("日期")
plt.ylabel("每日新增确诊（人）")
plt.title('世界每日新增病例(2021 年 12 月 5 日-2022 年 1 月 20 日)')
plt.show()

```

可以看到和原数据还是有一定的差距的，原因是这里面的数据过少，没有找到更有效地的规律。



五、 实验感想与总结

本次 python 大作业是一次较为完整的，从数据爬取到处理分析的整个过程。整个实验耗费的时间和精力也是非常多的。在实验的一开始，需找到能够爬取的网站时，就颇费功夫，有的网站需要挂 vpn，而有的网站则设置了一定的反爬措施，要爬取到需要的数据并能处理的数据并不容易。爬取下来之后，要对数据来进行分析，在数据可视化和数据处理两节课上，我们学习了基本的数据

处理的技能，而在此次大作业中就已经得到了整体的应用。只有从数据中提取出有效的信息，并将其绘制成合适的图形，才能更好的展现出数据所反映的背后的趋势和信息，并帮助做未来的预测。

在整个在本次实验中，不仅锻炼了我爬虫的能力和数据处理的能力，同时也帮助我对全球新冠疫情的发展态势和当前的一些数据有了进一步的了解，让我学习到了更多的 python 编码技巧，python 编写更加熟练。本次实验也表明世界的新冠疫情仍在延续，我们将长期与新冠共存，国际抗疫形势依旧严峻，不容我们小觑。