内蒙古大学
INNER MONGOLIA UNIVERSITY

学士学位论文

基于 Flask 的可视化 COVID-19 疫情数据监测平台

作者姓名：刘浩鑫

指导教师：王震

学位类别：学年论文

专　　业：信息与计算科学

学院（系）：内蒙古大学 数学科学学院

2021 年 9 月

Visual COVID-19 epidemic data monitoring platform based on Flask and MySQL

A thesis submitted to

University of Chinese Academy of Sciences

in partial fulfillment of the requirement

for the degree of

Bachelor of Natural Science

in Information and Computing Sciences

By

Liu Haoxin

Supervisor: Associate Professor Wang Zhen

Department of mathmatics, Inner Mongolia University

September, 2021

摘 要

自 2019 年末一种新型冠状病毒（COVID-19）在我国武汉被发现以来，人们的日常生活收到了极大的影响。昔日车水马龙的街道上冷冷清清，再也没有以往的欢声笑语。这样的环境给我们带来了恐慌，因此疫情数据的监测与预测也越来越显得重要。及时、准确的疫情数据对于国家政策的制定、社会的经济发展具有十分重要的意义。

此平台以腾讯疫情数据网站所公布各省市、国内外疫情数据作为研究对象，使用 Python 爬取数据并将数据存储于 MySQL 数据库中，然后使用 Flask 技术将数据可视化展现在面板中以供监测，并提供累计确诊地图、新增确诊病例地图。

关键词： Python，Flask，COVID-19，数据可视化，MySQL

Abstract

Since the end of 2019, a New Coronavirus (COVID-19) was discovered in Wuhan, China, and people's daily life has been greatly affected. The busy streets of the past were deserted, without the laughter of the past. Such an environment has brought us panic, so the monitoring and prediction of epidemic data is becoming more and more important. Timely and accurate epidemic data is of great significance for the formulation of national policies and social and economic development.

This platform takes the epidemic data of provinces and cities, domestic and foreign countries published by Tencent epidemic data website as the research object, uses Python to crawl the data and store the data in MySQL database, then uses flask technology to visualize the data in the panel for monitoring, and provides cumulative confirmed map and new confirmed case map.

Keywords: Python, Flask, COVID-19, Data Visualization, MySQL

目 录

第 1 章 绪论 ····· 1
1.1 研究背景与意义 ····· 1
1.1.1 研究背景 ····· 1
1.1.2 研究意义 ····· 1
1.2 国外研究现状 ····· 2
1.3 研究方法 ····· 2
第 2 章 开发环境及其简介 ····· 5
2.1 开发环境 ····· 5
2.2 开发环境简介 ····· 5
第 3 章 疫情数据的获取与存储 ····· 9
3.1 数据的获取 ····· 9
3.2 数据的存储 ····· 11
第 4 章 网页设计 ····· 13
4.1 主页（index.html）的设计 ····· 13
4.2 数据可视化 ····· 13
4.3 对手机设备的适配 ····· 15
附录 A Python 代码 ····· 17
A.1 spider.py—用来爬取网页数据 ····· 17
A.2 utils.py—app.py 中所调用的方法 ····· 20
A.3 app.py—启动 Flask 的主程序 ····· 23
附录 B 网页代码 ····· 27
B.1 index.html ····· 27
B.2 controller.js ····· 29
B.3 ec_center.js ····· 32

图形列表

1.1 谷歌新闻疫情数据监测网站 ····· 2

2.1 Flask 工作流程 ····· 6

3.1 查看腾讯新闻疫情数据网站 network 情况 ····· 9

4.1 全国累计趋势与新增趋势 ····· 14

第 1 章　绪论

1.1　研究背景与意义

1.1.1　研究背景

目前，世界仍在与新冠病毒作斗争，世界正处于大变革、大调整和两极分化的转折点。从目前疫情来看，即使一些国家实现了所谓的“群体免疫”，但除中国、韩国、朝鲜和新西兰等少数国家外，大多数国家的社会开放和经济重启计划仍远未达到预期。一些医学专家甚至断言，COVID-19 将与人类长期共存。在此之前，由于中美之间的结构性竞争和贸易摩擦，全球经济地理空间重组。COVID-19 的爆发无疑加剧了中美之间固有的差异。疫情发生后，世界格局将迎来新一轮深刻调整，世界新秩序正在全球抗击疫情的斗争中悄然孕育。展望未来，以开放、包容、协作和协商为特征的全球安全进程可能会突然结束，取而代之的是全球供应链和产业链的中断、国家间人员交流的限制性孤立以及各种种族主义和仇外心理的抬头。在新的政治形势和新的技术环境下，大数据被广泛应用于新冠疫情防控和社会稳定监测。以新冠疫情防控为契机，以精准安全管理为目标，以大数据为核心动力，全球安全管理人工智能时代正在悄然开启。

2020 年 2 月，世卫组织总干事谭德塞在记者会上宣布，将新型冠状病毒命名为“COVID-19”。据国家卫生委员会统计，截止 2021 年 8 月 7 日，我国 31 个省（区、市）和新疆建设兵团累计确诊病例 121454 例，累计死亡 5654 人；国外累计确诊病例 202337884，累计死亡 4285250 人。我国国内确诊病例数仍呈现上升趋势，可见当前国内疫情形势仍然相当严峻。

1.1.2　研究意义

新型冠状病毒肺炎疫情在 2019 年底爆发，并迅速在全球范围内传播。疫情发展实况与每个人的工作和生活息息相关。同时，网络上也出现了疫情相关的虚假信息和谣言，扰乱了社会防疫抗疫秩序。基于爬虫编程技术实现了对新冠疫情数据的采集，通过疫情数据可视化，将国内外疫情形态以更直观的方式传递给大众，帮助人们更好地了解国内外疫情实况，提高防范意识，从而有效地协调、安排和开展各项工作和生活等。本系统使用 Flask 框架、Python 构建轻量级疫情数据发布平台，将疫情数据转换为可视化的地图与表格，最大程度上满足疫情时代

下人们的普遍需求。本文将从疫情数据的获取及存储、网页设计等几个方面进行阐述。(?)

2020 年初，新型冠状病毒肺炎疫情已成为世界焦点，从疫情初现端倪，政府和企业在此次疫情中的数据可视化，让公众直观地掌握疫情动态。达到防控风险，抑制疫情的再扩散的目的。当今世界已经进入大数据时代，各国已将数据列为新的战略资源。在经济高速运转的今天，数据不再局限于小小的一个圈子，而是更加开放、包容，数据规模也更为客观。数据不再是简单的数字，如何从数据的背后挖掘出数据潜藏的价值成为国家和企业关心的热点问题。(?)

1.2 国外研究现状

针对全球流行的新型冠状病毒，Google 新闻推出基于维基百科以及其他数据来源的疫情数据监测平台。世卫组织（WHO）也在其官网发布疫情数据以及预防感染措施，以减少人们对新冠病毒的恐惧。

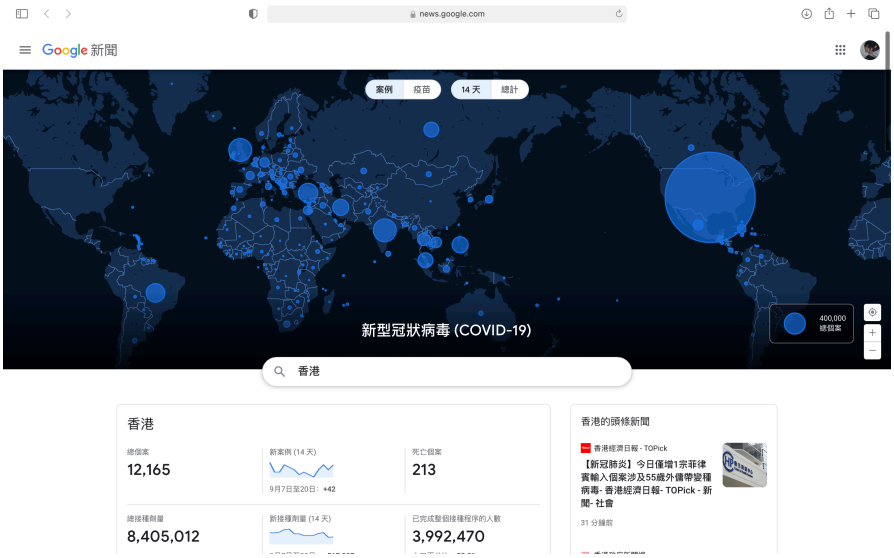

图 1.1 谷歌新闻疫情数据监测网站

Figure 1.1 Google News epidemic data monitoring website

1.3 研究方法

由于时间和精力的限制，本系统采用的疫情原始数据来自于阿里巴巴、腾讯或百度等已有疫情数据平台的数据源，而不再去统计疫情数据。

在现实生活中，单纯的数字不能直观地显示一段时间以来的发展状况，并且

人脑对图像信息的处理能力也要优于对单纯数据的处理, 所以将数据可视化更有利于普通大众更快地获取信息。同时, 数据可能是多维度的, 将数据按照一定规则分类、排序可视化之后更能清楚地看到数据的多维性, 更能表达出数据与数据之间的关系。当前的数据可视化的系统设计方法有多种, 但是主要的设计方法还是设计前端 + 后端 + 数据库的主体思路。(**?**)

基于 Python+Echarts 的大数据可视化系统采用 B/S 架构, 借助于 Python 强大的数据获取和处理技术实现了区域网络餐饮数据的采集、清洗、整理及分析计算工作并推送至 MySQL 数据库中。后台采用基于 Python 的 Flask 框架实现数据接口功能, 前端综合运用了 HTML、CSS、JavaScript 等, 并结合 Echarts 数据可视化组件, 实现了数据到可视化图表的转换。(**?**)

第 2 章　开发环境及其简介

2.1　开发环境

以下为开发环境列表

开发环境	名称	版本
开发系统	macOS Big Sur	11.5.1
IDE	PyCharm	2021.2
前端	Vscode	1.60.1 (Universal)
Python	Python	3.7.1
Python 依赖	flask, requests, pymysql	—
数据库	MySQL	8.0.26 Homebrew

2.2　开发环境简介

Flask 是一个使用 Python 编写的轻量级 Web 应用框架，包括 Werkzeug 和 Jinja2 两个核心函数库，它们分别负责业务处理和安全方面的功能，这些基础函数为 web 项目开发过程提供了丰富的基础组件。。其 WSGI 工具箱采用 Werkzeug ，模板引擎则使用 Jinja2 ，Flask 使用 BSD 授权。Flask 的基本模式为在程序里将一个视图函数分配给一个 URL，每当用户访问这个 URL 时，系统就会执行给该 URL 分配好的视图函数，获取函数的返回值并将其显示到浏览器上，其工作过程如下所示。(**?**)

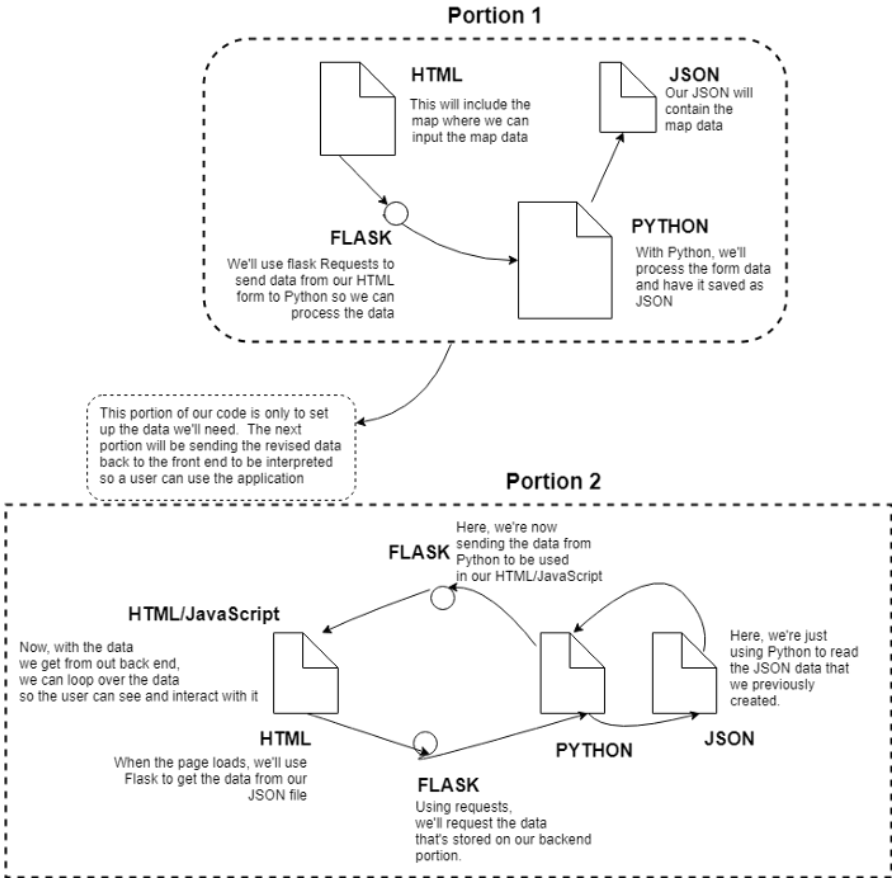

图 2.1 Flask 工作流程

Figure 2.1 Flash workflow

默认情况下，Flask 不包含数据库抽象层、表单验证，或是其它任何已有多种库可以胜任的功能。然而，Flask 支持用扩展来给应用添加这些功能，如同是 Flask 本身实现的一样。众多的扩展提供了数据库集成、表单验证、上传处理、各种各样的开放认证技术等功能。Flask 也许是“微小”的，但它已准备好在需求繁杂的生产环境中投入使用。**(?)** 其特色有如下：

- 内置开发用服务器和调试器
- 集成的单元测试支持
- RESTful 请求分派
- 使用 Jinja2 模板引擎
- 支持安全 cookie（客户端会话）
- WSGI1.0 兼容
- 基于 Unicode
- 详细的文件、教学

- Google App Engine 兼容
- 可用 Extensions 增加其他功能

第 3 章　疫情数据的获取与存储

3.1　数据的获取

通过分析腾讯新冠肺炎最新动态网站 https://news.qq.com/zt2020/page/feiyan.htm，并使用开发者工具对其进行分析，可以得到腾讯疫情数据的接口如下。

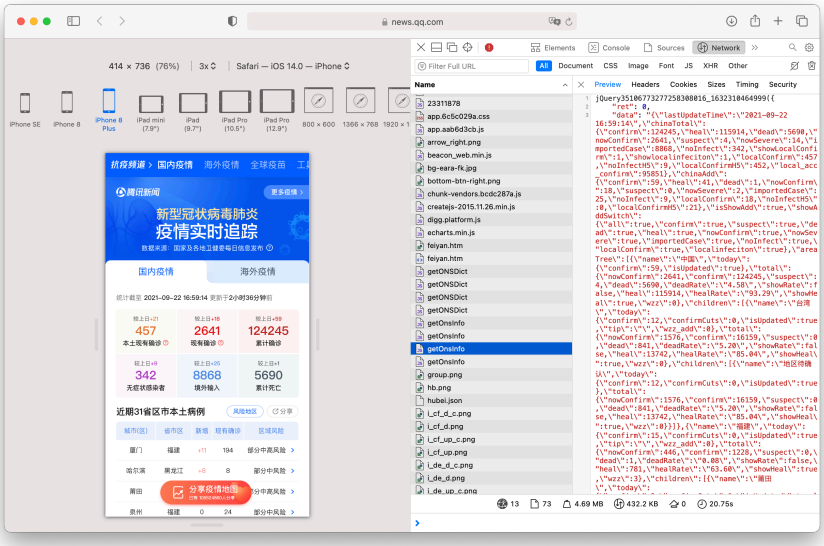

图 3.1 查看腾讯新闻疫情数据网站 network 情况

Figure 3.1 Check Tencent News Epidemic Data Website Network Status

之后，使用 Python 的 requests 库请求数据接口，然后解析 json 数据，将数据转换为字典类型，代码如下。

```
def get_tencent_data():
    url = 'https://view.inews.qq.com/g2/getOnsInfo?name=disease_h5'
    url_his = 'https://view.inews.qq.com/g2/getOnsInfo?name=disease_other'
    headers = {
        'user-agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/65.0.3325.181 Safari/537.36',
    }

    r = requests.get(url, headers)
    res = json.loads(r.text)  # json字符串转字典
    data_all = json.loads(res['data'])

    r_his = requests.get(url_his, headers)
    res_his = json.loads(r_his.text)
```

```
    data_his = json.loads(res_his['data'])

    history = {}  # 历史数据
```

r 的键值为 ret，data，其中 ret 为响应值，0 为请求成功，data 则是我们所需要的数据。使用 json 库将 data 转为字典类型，对数据进行一些处理之后，存入数据库。

```
    for i in data_his["chinaDayList"]:
        ds = "2020." + i["date"]
        tup = time.strptime(ds, "%Y.%m.%d")
        ds = time.strftime("%Y-%m-%d", tup)  # 改变时间格式
        confirm = i["confirm"]
        suspect = i["suspect"]
        heal = i["heal"]
        dead = i["dead"]
        history[ds] = {"confirm": confirm, "suspect": suspect, "heal": heal, "dead": dead}
    for i in data_his["chinaDayAddList"]:
        ds = "2020." + i["date"]
        tup = time.strptime(ds, "%Y.%m.%d")
        ds = time.strftime("%Y-%m-%d", tup)
        confirm = i["confirm"]
        suspect = i["suspect"]
        heal = i["heal"]
        dead = i["dead"]
        history[ds].update({"confirm_add": confirm, "suspect_add": suspect, "heal_add": heal
    , "dead_add": dead})
```

将爬取到的 json 数据转换为字典类型并存入到 histoty 与 details 中。

```
    def update_details():
    """
    更新 details 表
    :return:
    """
    cursor = None
    conn = None
    try:
        li = get_tencent_data()[1]  # 0 是历史数据字典,1 最新详细数据列表
        conn, cursor = get_conn()
        sql = """insert into details(update_time,province,city,confirm,confirm_add,heal,dead
    ) values(%s,%s,%s,%s,%s,%s,%s)"""
        sql_query = """select %s='(select update_time from details order by id desc limit 1)
    '"""  # 对比当前最大时间戳
```

```
        cursor.execute(sql_query, li[0][0])
        if not cursor.fetchone()[0]:
            print(f"{time.asctime()}开始更新最新数据")
            for item in li:
                cursor.execute(sql, item)
            conn.commit()  # 提交事务 update delete insert操作
            print(f"{time.asctime()}更新最新数据完毕")
        else:
            print(f"{time.asctime()}已是最新数据! ")
    except:
        traceback.print_exc()
    finally:
        close_conn(conn, cursor)
```

3.2　数据的存储

本系统所用到 MySQL 数据库为 cov，内含 history 与 details 两个表，history 用于存放过去三十天的疫情数据，detail 用来存放今日国内疫情数据，其各自结构如下所示。

```
mysql> desc history;
+-------------+------+------+-----+---------+-------+
| Field       | Type | Null | Key | Default | Extra |
+-------------+------+------+-----+---------+-------+
| ds          | date | YES  |     | NULL    |       |
| confirm     | int  | YES  |     | NULL    |       |
| suspect     | int  | YES  |     | NULL    |       |
| heal        | int  | YES  |     | NULL    |       |
| dead        | int  | YES  |     | NULL    |       |
| confirm_add | int  | YES  |     | NULL    |       |
| suspect_add | int  | YES  |     | NULL    |       |
| heal_add    | int  | YES  |     | NULL    |       |
| dead_add    | int  | YES  |     | NULL    |       |
+-------------+------+------+-----+---------+-------+
9 rows in set (0.00 sec)

mysql> desc details;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| update_time | date        | YES  |     | NULL    |       |
| province    | varchar(25) | YES  |     | NULL    |       |
| city        | varchar(25) | YES  |     | NULL    |       |
| confirm     | int         | YES  |     | NULL    |       |
```

```
| confirm_add | int         | YES  |     | NULL    |       |
| heal        | int         | YES  |     | NULL    |       |
| dead        | int         | YES  |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
7 rows in set (0.00 sec)
```

创建与关闭数据库连接。

```
    def get_conn():
    """
    :return: 连接, 游标
    """
    # 创建连接
    conn = pymysql.connect(host="127.0.0.1",
                           user="root",
                           password="123456",
                           db="cov",
                           charset="utf8")
    # 创建游标
    cursor = conn.cursor()  # 执行完毕返回的结果集默认以元组显示
    return conn, cursor


def close_conn(conn, cursor):
    if cursor:
        cursor.close()
    if conn:
        conn.close()
```

第 4 章　网页设计

4.1　主页（index.html）的设计

Html 部分较为清晰简单，主页上部设计一列导航栏，主体部分分为左中右三个大部分，每个大部分分为两个小部分，后期将使用 Echarts 绘制图表并进行响应式布局。

```
<div class="row-cols-md-4 row">
    <div id="l1">我是左1</div>
    <div id="l2">我是左2</div>
</div>
<div class="row-cols-md-4 row">
    <div id="c1" class="row clearfix">
        <div class="txt"><h2 class="text-lg-center text-center">累计确诊</h2></div>
        <div class="txt"><h2 class="text-lg-center text-center">剩余疑似</h2></div>
        <div class="txt"><h2 class="text-lg-center text-center">累计治愈</h2></div>
        <div class="txt"><h2 class="text-lg-center text-center">累计死亡</h2></div>
        <div class="num"><h1 class="text-primary text-center"></h1></div>
        <div class="num"><h1 class="text-primary text-center"></h1></div>
        <div class="num"><h1 class="text-primary text-center"></h1></div>
        <div class="num"><h1 class="text-primary text-center"></h1></div>
    </div>
    <div id="c2">我是中2</div>
</div>
<div class="row-cols-md-4 row">
    <div id="r1">我是右1</div>
    <div id="r2">我是右2</div>
</div>
```

4.2　数据可视化

为了使得该系统在不同屏幕尺寸的设备上能够正确显示，按照 Bootstrap 框架对前端网页进行构建。页面顶部为导航栏（navbar navbar-default），作为不同功能分区以及后续更多功能的入口，右端搜索接口接入百度搜索，提供更为丰富的搜索功能。页面主体分为三个大部分，其中包括六个小部分，并按从左到右排列，每个 div 的 id 依次记为 l1，l2，c1，c2，r1，r2。对于每一个列，其 div 的 class 设置为 row-cols-md-4 row，以实现响应式布局。左上角 l1 为全国累计趋势

曲线表，使用 ECharts 库以实现绘制。因为纵轴所表示的两个数值差距过大，l1 与 l2 使用两个 y 轴绘制，并使用不同颜色加以区分。

在 utils.py 中，定义了每个 div 获取数据的方法，例如 getunderline l1underline data，通过调用 utils.py 中的方法，从而连接到数据库并获得每个 div 所需的数据 data。在 controller.js 中定义了数据在图表中的显示形式。

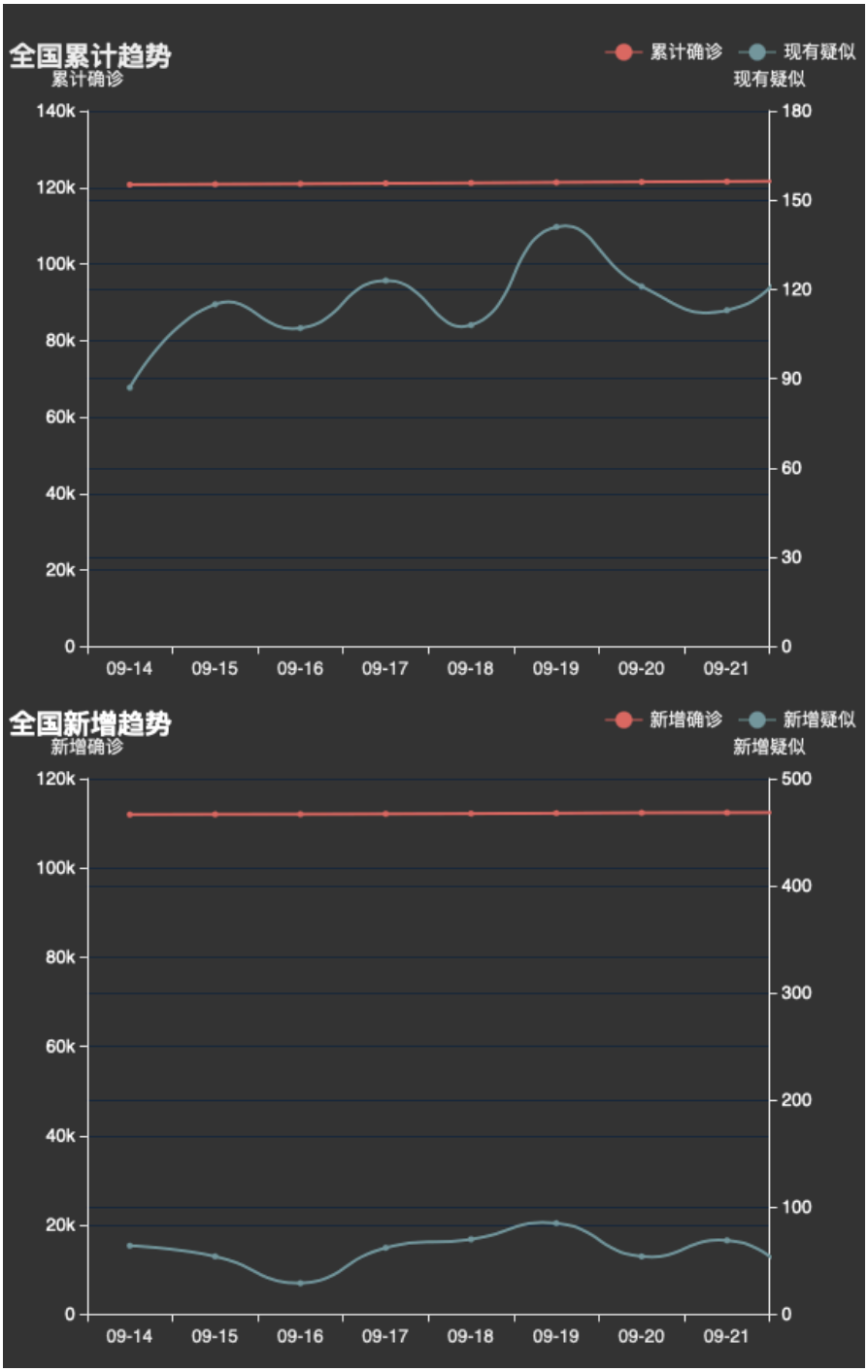

图 4.1 全国累计趋势与新增趋势

Figure 4.1 National cumulative trends and new trends

中上数据统计部分，使用类选择器显示当前累计确诊、剩余疑似、累计治愈以及累计死亡数据。

```
function get_c1_data() {
$.ajax({
    url: "/c1",
    success: function (data) {
        $(".num h1").eq(0).text(data.confirm);
        console.log(data.confirm);
        $(".num h1").eq(1).text(data.suspect);
        $(".num h1").eq(2).text(data.heal);
        $(".num h1").eq(3).text(data.dead);
    },
    error: function (xhr, type, errorThrown) {

    }
})
}
```

4.3　对手机设备的适配

为了使得该系统可以在手机设备上显示，在 main.css 中加入了旋转屏幕的样式。当屏幕尺寸判断为竖屏时，将旋转页面，使其变为横屏。

```
@media screen and (orientation: portrait) {
    html{
        width : 100vmin;
        height : 100vmax;
    }
    body{
        width : 100vmin;
        height : 100vmax;
    }
    #gyroContain{
        width : 100vmax;
        height : 100vmin;
        transform-origin: top left;
        transform: rotate(90deg) translate(0,-100vmin);
    }
  }
@media screen and (orientation: landscape) {
    html{
        width : 100vmax;
        height : 100vmin;
    }
```

```
        body{
23          width : 100vmax;
            height : 100vmin;
25      }
        #gyroContain{
27          width : 100vmax;
            height : 100vmin;
29      }
    }
```

附录 A　Python 代码

A.1　spider.py—用来爬取网页数据

```
import os
import pymysql
import time
import json
import traceback  # 追踪异常
import requests


def get_tencent_data():
    url = 'https://view.inews.qq.com/g2/getOnsInfo?name=disease_h5'
    url_his = 'https://view.inews.qq.com/g2/getOnsInfo?name=disease_other'
    headers = {
        'user-agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/65.0.3325.181 Safari/537.36',
    }

    r = requests.get(url, headers)
    res = json.loads(r.text)  # json字符串转字典
    data_all = json.loads(res['data'])

    r_his = requests.get(url_his, headers)
    res_his = json.loads(r_his.text)
    data_his = json.loads(res_his['data'])

    history = {}  # 历史数据

    for i in data_his["chinaDayList"]:
        ds = "2020." + i["date"]
        tup = time.strptime(ds, "%Y.%m.%d")
        ds = time.strftime("%Y-%m-%d", tup)  # 改变时间格式
        confirm = i["confirm"]
        suspect = i["suspect"]
        heal = i["heal"]
        dead = i["dead"]
        history[ds] = {"confirm": confirm, "suspect": suspect, "heal": heal, "dead": dead}
    for i in data_his["chinaDayAddList"]:
        ds = "2020." + i["date"]
        tup = time.strptime(ds, "%Y.%m.%d")
        ds = time.strftime("%Y-%m-%d", tup)
        confirm = i["confirm"]
        suspect = i["suspect"]
        heal = i["heal"]
```

```
        dead = i["dead"]
        history[ds].update({"confirm_add": confirm, "suspect_add": suspect, "heal_add": heal, "dead_add": dead})

    # 下面就不用动了
    details = []  # 当日详细数据
    update_time = data_all["lastUpdateTime"]
    data_country = data_all["areaTree"]  # list 25个国家
    data_province = data_country[0]["children"]  # 中国各省
    for pro_infos in data_province:
        province = pro_infos["name"]  # 省名
        for city_infos in pro_infos["children"]:
            city = city_infos["name"]
            confirm = city_infos["total"]["confirm"]
            confirm_add = city_infos["today"]["confirm"]
            heal = city_infos["total"]["heal"]
            dead = city_infos["total"]["dead"]
            details.append([update_time, province, city, confirm, confirm_add, heal, dead])
    return history, details


def get_conn():
    """
    :return: 连接, 游标
    """
    # 创建连接
    conn = pymysql.connect(host="127.0.0.1",
                           user="root",
                           password="123456",
                           db="cov",
                           charset="utf8")
    # 创建游标
    cursor = conn.cursor()  # 执行完毕返回的结果集默认以元组显示
    return conn, cursor


def close_conn(conn, cursor):
    if cursor:
        cursor.close()
    if conn:
        conn.close()


def update_details():
    """
    更新 details 表
    :return:
    """
```

```
    cursor = None
90  conn = None
    try:
92      li = get_tencent_data()[1]  # 0 是历史数据字典,1 最新详细数据列表
        conn, cursor = get_conn()
94      sql = """insert into details(update_time,province,city,confirm,confirm_add,heal,dead
    ) values(%s,%s,%s,%s,%s,%s,%s)"""
        sql_query = """select %s=(select update_time from details order by id desc limit 1)
    """  # 对比当前最大时间戳
96      cursor.execute(sql_query, li[0][0])
        if not cursor.fetchone()[0]:
98          print(f"{time.asctime()}开始更新最新数据")
            for item in li:
100             cursor.execute(sql, item)
            conn.commit()  # 提交事务 update delete insert操作
102         print(f"{time.asctime()}更新最新数据完毕")
        else:
104         print(f"{time.asctime()}已是最新数据! ")
    except:
106     traceback.print_exc()
    finally:
108     close_conn(conn, cursor)


110
def insert_history():
112 """
        插入历史数据
114 :return:
    """
116 cursor = None
    conn = None
118 try:
        dic = get_tencent_data()[0]  # 0 是历史数据字典,1 最新详细数据列表
120     print(f"{time.asctime()}开始插入历史数据")
        conn, cursor = get_conn()
122     sql = """insert into history values(%s,%s,%s,%s,%s,%s,%s,%s,%s)"""
        for k, v in dic.items():
124         # item 格式 {'2020-01-13': {'confirm': 41, 'suspect': 0, 'heal': 0, 'dead': 1}
            cursor.execute(sql, [k, v.get("confirm"), v.get("confirm_add"), v.get("suspect")
    ,
126                              v.get("suspect_add"), v.get("heal"), v.get("heal_add"),
                                 v.get("dead"), v.get("dead_add")])
128
        conn.commit()  # 提交事务 update delete insert操作
130     print(f"{time.asctime()}插入历史数据完毕")
    except:
132     traceback.print_exc()
    finally:
```

```
        close_conn(conn, cursor)


def update_history():
    """
    更新历史数据
    :return:
    """
    cursor = None
    conn = None
    try:
        dic = get_tencent_data()[0]  # 0 是历史数据字典,1 最新详细数据列表
        print(f"{time.asctime()}开始更新历史数据")
        conn, cursor = get_conn()
        sql = """insert into history (ds,confirm,suspect,heal,dead,confirm_add,suspect_add,
    heal_add,dead_add) values(%s,%s,%s,%s,%s,%s,%s,%s,%s)"""
        sql_query = """select confirm from history where ds=%s"""
        for k, v in dic.items():
            # item 格式 {'2020-01-13': {'confirm': 41, 'suspect': 0, 'heal': 0, 'dead': 1}}
            if not cursor.execute(sql_query, k):
                cursor.execute(sql, [k, v.get("confirm"), v.get("confirm_add"), v.get("
    suspect"),
                                     v.get("suspect_add"), v.get("heal"), v.get("heal_add"),
                                     v.get("dead"), v.get("dead_add")])
        conn.commit()  # 提交事务 update delete insert操作
        print(f"{time.asctime()}历史数据更新完毕")
    except:
        traceback.print_exc()
    finally:
        close_conn(conn, cursor)


if __name__ == "__main__":
    insert_history()
    update_history()
    update_details()
```

A.2 utils.py—app.py 中所调用的方法

```
    import time
    import pymysql
    from decimal import Decimal
    import json

```

```
7   def get_time():
        time_str = time.strftime("%Y{}%m{}%d{} %X")
9       return time_str.format("年", "月", "日")


11
    def get_conn():
13      """
        :return: 连接, 游标
15      """
        # 创建连接
17      conn = pymysql.connect(host="localhost",
                               user="root",
19                             password="123456",
                               db="cov",
21                             charset="utf8")
        # 创建游标
23      cursor = conn.cursor()  # 执行完毕返回的结果集默认以元组显示
        return conn, cursor
25


27  def close_conn(conn, cursor):
        cursor.close()
29      conn.close()


31
    def query(sql, *args):
33      """
        封装通用查询
35      :param sql:
        :param args:
37      :return: 返回查询到的结果, ((),(),)的形式
        """
39      conn, cursor = get_conn()
        cursor.execute(sql, args)
41      res = cursor.fetchall()
        close_conn(conn, cursor)
43      return res


45
    def get_c1_data():
47      """
        :return: 返回大屏div id=c1 的数据
49      """
        # 因为会更新多次数据, 取时间戳最新的那组数据
51      sql = "select sum(confirm)," \
              "(select suspect from history order by ds desc limit 1)," \
53            "sum(heal)," \
              "sum(dead) " \
```

```
55              "from details " \
                "where update_time=(select update_time from details order by update_time desc
    limit 1) "
57              #"where update_time=(select update_time from details order by update_time desc
     limit 1) "
        res = query(sql)
59
        res_list = [str(i) for i in res[0]]
61      res_tuple = tuple(res_list)
        return res_tuple
63


65  def get_c2_data():
        """
67      :return:  返回各省数据
        """
69      # 因为会更新多次数据, 取时间戳最新的那组数据
        sql = "select province,sum(confirm) from details " \
71            "where update_time=(select update_time from details " \
              "order by update_time desc limit 1) " \
73            "group by province"
        res = query(sql)
75      return res


77
    def get_l1_data():
79      sql = "select ds,confirm,suspect,heal,dead from history"
        res = query(sql)
81      return res


83
    def get_l2_data():
85      sql = "select ds,confirm_add,suspect_add from history"
        res = query(sql)
87      return res


89
    def get_r1_data():
91      """
        :return:  返回非湖北地区城市确诊人数前5名
93      """
        sql = 'SELECT province,confirm_add FROM ' \
95            '(select province,sum(confirm_add) as confirm_add from details  ' \
              'where update_time=(select update_time from details order by update_time desc
    limit 1) ' \
97            'group by province) as a ' \
              'ORDER BY confirm_add DESC LIMIT 5'
99      res = query(sql)
```

```
        return res
101
103 def get_r2_data():
        '''
105         获取世界各国的疫情数据
            :return:
107     '''
        # 因为会更新多次数据，取时间戳最新的那组数据
109     sql = "select province,sum(confirm_add) from details " \
              "where update_time=(select update_time from details " \
111           "order by update_time desc limit 1) " \
              "group by province"
113     res = query(sql)
        return res
115

117 if __name__ == "__main__":
        print(get_c1_data())
```

A.3　app.py—启动 Flask 的主程序

```
    import utils
2   from flask import Flask
    from flask import request
4   from flask import render_template
    from flask import jsonify
6   from flask_bootstrap import Bootstrap
    import os
8   import string

10  os.system(
        'python3 /Users/liuhaoxin/Documents/GitHub/covid-19-china/spider.py && python3 /
    Users/liuhaoxin/Documents/GitHub/covid-19-china/utils.py')
12
    app = Flask(__name__)
14  bootstrap = Bootstrap(app)

16  @app.route('/')
    def hello_world():
18      return render_template("index.html")

20

    @app.route("/c1")
22  def get_c1_data():
```

```
    data = utils.get_c1_data()
    return jsonify({"confirm":data[0],"suspect":data[1],"heal":data[2],"dead":data[3]})


@app.route("/c2")
def get_c2_data():
    res = []
    for tup in utils.get_c2_data():
        # print(tup)
        res.append({"name":tup[0],"value":int(tup[1])})
    return jsonify({"data":res})


@app.route("/l1")
def get_l1_data():
    data = utils.get_l1_data()
    day,confirm,suspect,heal,dead = [],[],[],[],[]
    for a,b,c,d,e in data[7:]:
        day.append(a.strftime("%m-%d"))
        #  a是datatime类型
        confirm.append(b)
        suspect.append(c)
        heal.append(d)
        dead.append(e)
    return jsonify({"day": day, "confirm": confirm, "suspect": suspect, "heal": heal, "dead": dead})


@app.route("/l2")
def get_l2_data():
    data = utils.get_l2_data()
    day, confirm_add, suspect_add = [], [], []
    for a, b, c in data[7:]:
        day.append(a.strftime("%m-%d"))  # a是datatime类型
        confirm_add.append(b)
        suspect_add.append(c)
    return jsonify({"day": day, "confirm_add": confirm_add, "suspect_add": suspect_add})


@app.route("/r1")
def get_r1_data():
    data = utils.get_r1_data()
    city = []
    confirm = []
    for k,v in data:
        city.append(k)
        confirm.append(int(v))
    return jsonify({"city": city, "confirm": confirm})
```

```
70


72  @app.route("/r2")
    def get_r2_data():
74      res = []
        for tup in utils.get_r2_data():
76          # print(tup)
            res.append({"name": tup[0], "value": int(tup[1])})
78      return jsonify({"data": res})


80
    @app.route("/time")
82  def get_time():
        return utils.get_time()
84


86  if __name__ == '__main__':
        # app.run(host='127.0.0.1', port=5000)
88      bootstrap.run()
```

附录 B　网页代码

B.1　index.html

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>疫情监控系统</title>
    <link rel="icon" href="../static/icon.ico">
    <link rel="stylesheet" href="https://cdn.staticfile.org/twitter-bootstrap/3.3.7/css/bootstrap.min.css">
    <script src="https://cdn.staticfile.org/jquery/2.1.1/jquery.min.js"></script>
    <script src="https://cdn.staticfile.org/twitter-bootstrap/3.3.7/js/bootstrap.min.js"></script>
    <script src="../static/js/jquery-1.11.1.min.js"></script>
    <script src="../static/js/echarts.min.js"></script>
    <script src="../static/js/china.js"></script>
    <link href="../static/css/main.css" rel="stylesheet"/>
</head>
<body>

<div id="tim"></div>
<div id="gyroContain" class="col clearfix">
    <div class="row clearfix">
        <div class="col-md-12 column">
            <nav class="navbar navbar-default" role="navigation">
                <div class="navbar-header">
                    <a class="navbar-brand" href="#">疫情数据监控平台</a>
                </div>

                <div class="collapse navbar-collapse" id="bs-example-navbar-collapse-1">
                    <ul class="nav navbar-nav">
                        <li class="active">
                            <a href="#">国内数据</a>
                        </li>
                        <li>
                            <a href="#">国外数据</a>
                        </li>
                        <li class="dropdown">
                            <a href="#" class="dropdown-toggle" data-toggle="dropdown">更多<strong
                                    class="caret"></strong></a>
                            <ul class="dropdown-menu">
```

```
                                <li>
                                    <a href="https://www.baidu.com/#ie=utf8&wd=疫情最新政策">最新政策</a>
                                </li>
                                <li>
                                    <a href="https://www.baidu.com/#ie=utf8&wd=疫情当地政策">当地政策</a>
                                </li>
                                <li>
                                    <a href="https://www.baidu.com/#ie=utf8&wd=疫情国际消息">国际消息</a>
                                </li>
                                <li class="divider">
                                </li>
                                <li>
                                    <a href="#">健康码</a>
                                </li>
                                <li class="divider">
                                </li>
                                <li>
                                    <a href="#">防疫平台</a>
                                </li>
                            </ul>
                        </li>
                    </ul>

                    <ul class="nav navbar-nav navbar-right">
                        <form class="navbar-form navbar-left" role="search">
                            <div class="form-group">
                                <input type="text" class="form-control" />
                            </div>
                            <button type="submit" class="btn btn-default">搜索</button>
                        </form>
                    </ul>
                </div>

            </nav>
        </div>
    </div>
    <div class="row-cols-md-4 row">
        <div id="l1">我是左1</div>
        <div id="l2">我是左2</div>
    </div>
    <div class="row-cols-md-4 row">
        <div id="c1" class="row clearfix">
            <div class="txt"><h2 class="text-lg-center text-center">累计确诊</h2></div>
            <div class="txt"><h2 class="text-lg-center text-center">剩余疑似</h2></div>
            <div class="txt"><h2 class="text-lg-center text-center">累计治愈</h2></div>
```

```
                <div class="txt"><h2 class="text-lg-center text-center">累计死亡</h2></div>
                <div class="num"><h1 class="text-primary text-center"></h1></div>
                <div class="num"><h1 class="text-primary text-center"></h1></div>
                <div class="num"><h1 class="text-primary text-center"></h1></div>
                <div class="num"><h1 class="text-primary text-center"></h1></div>
            </div>
            <div id="c2">我是中2</div>
        </div>
        <div class="row-cols-md-4 row">
            <div id="r1">我是右1</div>
            <div id="r2">我是右2</div>
        </div>

        <script src="../static/js/ec_center.js"></script>
        <script src="../static/js/ec_left1.js"></script>
        <script src="../static/js/ec_left2.js"></script>
        <script src="../static/js/ec_right1.js"></script>
        <script src="../static/js/ec_right2.js"></script>
        <script src="../static/js/controller.js"></script>
    </div>
    </body>
    </html>
```

B.2　controller.js

```
function gettime() {
    $.ajax({
        url: "/time",
        timeout: 10000, //超时时间设置为10秒;
        success: function (data) {
            $("#tim").html(data)
        },
        error: function (xhr, type, errorThrown) {

        }
    });
}

function get_c1_data() {
    $.ajax({
        url: "/c1",
        success: function (data) {
            $(".num h1").eq(0).text(data.confirm);
            console.log(data.confirm);
            $(".num h1").eq(1).text(data.suspect);
            $(".num h1").eq(2).text(data.heal);
```

```
22                $(".num h1").eq(3).text(data.dead);
            },
24          error: function (xhr, type, errorThrown) {

26          }
        })
28  }

30  function get_c2_data() {
        $.ajax({
32          url: "/c2",
            success: function (data) {
34              ec_center_option.series[0].data = data.data
                ec_center.setOption(ec_center_option)
36          },
            error: function (xhr, type, errorThrown) {
38
            }
40      })
    }
42
    function get_l1_data() {
44      $.ajax({
            url: "/l1",
46          success: function (data) {
                var day = []
48              for (let i = 0; i < 8; i++) {
                    day[7 - i] = data.day[data.day.length - 1 - i]
50              }
                // ec_left1_Option.xAxis[0].data=data.day
52              ec_left1_Option.xAxis[0].data = day
                ec_left1_Option.series[0].data = data.confirm
54              console.log(data.confirm);
                ec_left1_Option.series[1].data = data.suspect
56              // ec_left1_Option.series[2].data = data.heal
                // ec_left1_Option.series[3].data = data.dead
58              ec_left1.setOption(ec_left1_Option)
                // console.log(day)
60          },
            error: function (xhr, type, errorThrown) {
62
            }
64      })
    }
66
    function get_l2_data() {
68      $.ajax({
            url: "/l2",
```

```
            success: function (data) {
                var day = []
                for (let i = 0; i < 8; i++) {
                    day[7 - i] = data.day[data.day.length - 1 - i]
                }
                ec_left2_Option.xAxis[0].data = day
                ec_left2_Option.series[0].data = data.confirm_add
                ec_left2_Option.series[1].data = data.suspect_add
                ec_left2.setOption(ec_left2_Option)
            },
            error: function (xhr, type, errorThrown) {

            }
        })
    }

    function get_r1_data() {
        $.ajax({
            url: "/r1",
            success: function (data) {
                ec_right1_option.xAxis.data = data.city;
                ec_right1_option.series[0].data = data.confirm;
                ec_right1.setOption(ec_right1_option);
            }
        })
    }

    function get_r2_data() {
        $.ajax({
            url: "/r2",
            success: function (data) {
                ec_right2_option.series[0].data = data.data
                ec_right2.setOption(ec_right2_option)

            },
            error: function (xhr, type, errorThrown) {

            }
        })
    }

    gettime()
    get_c1_data()
    get_c2_data()
    get_l1_data()
    get_l2_data()
    get_r1_data()
    get_r2_data()
```

```
118
    setInterval(gettime, 1000)
120 setInterval(get_c1_data, 1000 * 10)
    setInterval(get_c2_data, 10000 * 10)
122 setInterval(get_l1_data, 10000 * 10)
    setInterval(get_l2_data, 10000 * 10)
124 setInterval(get_r1_data, 10000 * 10)
    setInterval(get_r2_data, 10000 * 10)
```

B.3 ec_center.js

```
1   var ec_center = echarts.init(document.getElementById('c2'), "dark");

3   var mydata = []

5   var ec_center_option = {
        title: {
7           text: '全国累计确诊地图',
            subtext: '',
9           x: 'center'
        },
11      tooltip: {
            trigger: 'item'
13      },
        //左侧小导航图标
15      visualMap: {
            show: true,
17          x: 'left',
            y: 'bottom',
19          textStyle: {
                fontSize: 8,
21          },
            splitList: [{ start: 1,end: 9 },
23              {start: 10, end: 99 },
                { start: 100, end: 999 },
25              {  start: 1000, end: 9999 },
                { start: 10000 }],
27          color: ['#8A3310', '#C64918', '#E55B25', '#F2AD92', '#F9DCD1']
        },
29      //配置属性
        series: [{
31          name: '累计确诊人数',
            type: 'map',
33          mapType: 'china',
            roam: false, //拖动和缩放
35          itemStyle: {
```

```
                normal: {
                    borderWidth: .5, //区域边框宽度
                    borderColor: '#009fe8', //区域边框颜色
                    areaColor: "#ffefd5", //区域颜色
                },
                emphasis: { //鼠标滑过地图高亮的相关设置
                    borderWidth: .5,
                    borderColor: '#4b0082',
                    areaColor: "#fff",
                }
            },
            label: {
                normal: {
                    show: true, //省份名称
                    fontSize: 8,
                },
                emphasis: {
                    show: true,
                    fontSize: 8,
                }
            },
            data:[] //mydata //数据
        }]
    };
    ec_center.setOption(ec_center_option)
```

参考文献

李宏. 基于民众视阈下的疫情数据可视化设计路径研究 [J]. 包装工程艺术版, 2020, 41(10): 221-227.

赵娟. 浅谈时空大数据与数据可视化在疫情地图中的应用 [J]. 卫星导航定位技术文集 (2020), 2020.

郭宏曦. 新型冠状病毒肺炎疫情数据的可视化探索 [J]. 计算机与网络, 2020, 6.

陈俊生, 彭莉芬. 基于 Python+ Echarts 的大数据可视化系统的设计与实现 [J]. 安徽电子信息职业技术学院学报, 2019, 18(4): 6-9.

韩洪勇, 冉春晴, 陈硕. 基于 Echarts 和 Flask 的数据可视化系统 [J]. 中国新通信, 2020.

Grinberg M. Flask web development: developing web applications with python [M]. ” O’Reilly Media, Inc.”, 2018.