0、写在前面

作者：魏博 个人邮箱：Webeau24@outlook.com 欢迎学习交流

本篇数据分析项目来自于github上的公共数据分析开源项目bigdata_analyse的COVID-19新冠疫情的数据分析项目，个人按照一般数据分析步骤对代码进行复现、补充了数据探索步骤、优化了部分可视化与数据清洗部分代码。

```
#先看一下现在的路径
import os
# 获取当前工作路径
print("当前工作路径:", os.getcwd())
```

```
当前工作路径: d:\乱七八糟的文件\github开源项目\csse_covid_19_time_series
```

1、数据集说明

这是一份来自 Johns Hopkins University 在github 开源的全球新冠肺炎 COVID-19 数据集，每日时间序列汇总，包括确诊、死亡和治愈。所有数据来自每日病例报告。

> 由于数据集中没有美国的治愈数据，所以在统计全球的现有确诊人员和治愈率的时候会有很大误差，代码里面先不做这个处理，期待数据集的完善。

2、数据探索与数据清洗

这里我们先通过pandas库读取csv文件为dataframe格式

```
import pandas as pd
confirmed_data = pd.read_csv('time_series_covid19_confirmed_global.csv')
deaths_data = pd.read_csv('time_series_covid19_deaths_global.csv')
recovered_data = pd.read_csv('time_series_covid19_recovered_global.csv')
```

- 数据探索

```
# 1. 查看数据结构
print("确诊数据集结构:")
print(confirmed_data.shape)
print("\n死亡数据集结构:")
print(deaths_data.shape)
print("\n康复数据集结构:")
print(recovered_data.shape)
```

```
确诊数据集结构:
(274, 428)

死亡数据集结构:
(274, 428)

康复数据集结构:
(259, 428)
```

```
#输出其中一个数据表的前五行和前五列，大致了解数据的结构
print(confirmed_data.iloc[:5, :5])
274-259
```

```
  Province/State Country/Region       Lat       Long  1/22/20
0            NaN    Afghanistan  33.93911  67.709953        0
1            NaN        Albania  41.15330  20.168300        0
2            NaN        Algeria  28.03390   1.659600        0
3            NaN        Andorra  42.50630   1.521800        0
4            NaN         Angola -11.20270  17.873900        0






15
```

我们通过把数据集的列名从Pandas Index对象转为Python列表，打印输出这个列表来查看数据集的列名。检查到这三个数据集列名一致。

```
# 2. 检查列名是否一致
print("\n列名对比:")
print("确诊:", confirmed_data.columns.tolist())
print("死亡:", deaths_data.columns.tolist()[:5])
print("康复:", recovered_data.columns.tolist()[:5])
```

```
列名对比:
确诊: ['Province/State', 'Country/Region', 'Lat', 'Long', '1/22/20', '1/23/20',
'1/24/20', '1/25/20', '1/26/20', '1/27/20', '1/28/20', '1/29/20', '1/30/20',
'1/31/20', '2/1/20', '2/2/20', '2/3/20', '2/4/20', '2/5/20', '2/6/20', '2/7/20',
'2/8/20', '2/9/20', '2/10/20', '2/11/20', '2/12/20', '2/13/20', '2/14/20',
'2/15/20', '2/16/20', '2/17/20', '2/18/20', '2/19/20', '2/20/20', '2/21/20',
'2/22/20', '2/23/20', '2/24/20', '2/25/20', '2/26/20', '2/27/20', '2/28/20',
'2/29/20', '3/1/20', '3/2/20', '3/3/20', '3/4/20', '3/5/20', '3/6/20', '3/7/20',
'3/8/20', '3/9/20', '3/10/20', '3/11/20', '3/12/20', '3/13/20', '3/14/20',
'3/15/20', '3/16/20', '3/17/20', '3/18/20', '3/19/20', '3/20/20', '3/21/20',
'3/22/20', '3/23/20', '3/24/20', '3/25/20', '3/26/20', '3/27/20', '3/28/20',
'3/29/20', '3/30/20', '3/31/20', '4/1/20', '4/2/20', '4/3/20', '4/4/20', '4/5/20',
'4/6/20', '4/7/20', '4/8/20', '4/9/20', '4/10/20', '4/11/20', '4/12/20',
'4/13/20', '4/14/20', '4/15/20', '4/16/20', '4/17/20', '4/18/20', '4/19/20',
'4/20/20', '4/21/20', '4/22/20', '4/23/20', '4/24/20', '4/25/20', '4/26/20',
'4/27/20', '4/28/20', '4/29/20', '4/30/20', '5/1/20', '5/2/20', '5/3/20',
'5/4/20', '5/5/20', '5/6/20', '5/7/20', '5/8/20', '5/9/20', '5/10/20', '5/11/20',
'5/12/20', '5/13/20', '5/14/20', '5/15/20', '5/16/20', '5/17/20', '5/18/20',
'5/19/20', '5/20/20', '5/21/20', '5/22/20', '5/23/20', '5/24/20', '5/25/20',
'5/26/20', '5/27/20', '5/28/20', '5/29/20', '5/30/20', '5/31/20', '6/1/20',
'6/2/20', '6/3/20', '6/4/20', '6/5/20', '6/6/20', '6/7/20', '6/8/20', '6/9/20',
'6/10/20', '6/11/20', '6/12/20', '6/13/20', '6/14/20', '6/15/20', '6/16/20',
'6/17/20', '6/18/20', '6/19/20', '6/20/20', '6/21/20', '6/22/20', '6/23/20',
'6/24/20', '6/25/20', '6/26/20', '6/27/20', '6/28/20', '6/29/20', '6/30/20',
'7/1/20', '7/2/20', '7/3/20', '7/4/20', '7/5/20', '7/6/20', '7/7/20', '7/8/20',
'7/9/20', '7/10/20', '7/11/20', '7/12/20', '7/13/20', '7/14/20', '7/15/20',
'7/16/20', '7/17/20', '7/18/20', '7/19/20', '7/20/20', '7/21/20', '7/22/20',
'7/23/20', '7/24/20', '7/25/20', '7/26/20', '7/27/20', '7/28/20', '7/29/20',
'7/30/20', '7/31/20', '8/1/20', '8/2/20', '8/3/20', '8/4/20', '8/5/20', '8/6/20',
'8/7/20', '8/8/20', '8/9/20', '8/10/20', '8/11/20', '8/12/20', '8/13/20',
'8/14/20', '8/15/20', '8/16/20', '8/17/20', '8/18/20', '8/19/20', '8/20/20',
'8/21/20', '8/22/20', '8/23/20', '8/24/20', '8/25/20', '8/26/20', '8/27/20',
'8/28/20', '8/29/20', '8/30/20', '8/31/20', '9/1/20', '9/2/20', '9/3/20',
'9/4/20', '9/5/20', '9/6/20', '9/7/20', '9/8/20', '9/9/20', '9/10/20', '9/11/20',
'9/12/20', '9/13/20', '9/14/20', '9/15/20', '9/16/20', '9/17/20', '9/18/20',
'9/19/20', '9/20/20', '9/21/20', '9/22/20', '9/23/20', '9/24/20', '9/25/20',
'9/26/20', '9/27/20', '9/28/20', '9/29/20', '9/30/20', '10/1/20', '10/2/20',
'10/3/20', '10/4/20', '10/5/20', '10/6/20', '10/7/20', '10/8/20', '10/9/20',
'10/10/20', '10/11/20', '10/12/20', '10/13/20', '10/14/20', '10/15/20',
'10/16/20', '10/17/20', '10/18/20', '10/19/20', '10/20/20', '10/21/20',
'10/22/20', '10/23/20', '10/24/20', '10/25/20', '10/26/20', '10/27/20',
'10/28/20', '10/29/20', '10/30/20', '10/31/20', '11/1/20', '11/2/20', '11/3/20',
'11/4/20', '11/5/20', '11/6/20', '11/7/20', '11/8/20', '11/9/20', '11/10/20',
'11/11/20', '11/12/20', '11/13/20', '11/14/20', '11/15/20', '11/16/20',
'11/17/20', '11/18/20', '11/19/20', '11/20/20', '11/21/20', '11/22/20',
'11/23/20', '11/24/20', '11/25/20', '11/26/20', '11/27/20', '11/28/20',
'11/29/20', '11/30/20', '12/1/20', '12/2/20', '12/3/20', '12/4/20', '12/5/20',
'12/6/20', '12/7/20', '12/8/20', '12/9/20', '12/10/20', '12/11/20', '12/12/20',
'12/13/20', '12/14/20', '12/15/20', '12/16/20', '12/17/20', '12/18/20',
'12/19/20', '12/20/20', '12/21/20', '12/22/20', '12/23/20', '12/24/20',
'12/25/20', '12/26/20', '12/27/20', '12/28/20', '12/29/20', '12/30/20',
'12/31/20', '1/1/21', '1/2/21', '1/3/21', '1/4/21', '1/5/21', '1/6/21', '1/7/21',
'1/8/21', '1/9/21', '1/10/21', '1/11/21', '1/12/21', '1/13/21', '1/14/21',
'1/15/21', '1/16/21', '1/17/21', '1/18/21', '1/19/21', '1/20/21', '1/21/21',
'1/22/21', '1/23/21', '1/24/21', '1/25/21', '1/26/21', '1/27/21', '1/28/21',
'1/29/21', '1/30/21', '1/31/21', '2/1/21', '2/2/21', '2/3/21', '2/4/21', '2/5/21',
'2/6/21', '2/7/21', '2/8/21', '2/9/21', '2/10/21', '2/11/21', '2/12/21',
'2/13/21', '2/14/21', '2/15/21', '2/16/21', '2/17/21', '2/18/21', '2/19/21',
'2/20/21', '2/21/21', '2/22/21', '2/23/21', '2/24/21', '2/25/21', '2/26/21',
'2/27/21', '2/28/21', '3/1/21', '3/2/21', '3/3/21', '3/4/21', '3/5/21', '3/6/21',
'3/7/21', '3/8/21', '3/9/21', '3/10/21', '3/11/21', '3/12/21', '3/13/21',
'3/14/21', '3/15/21', '3/16/21', '3/17/21', '3/18/21', '3/19/21', '3/20/21']
死亡: ['Province/State', 'Country/Region', 'Lat', 'Long', '1/22/20']
康复: ['Province/State', 'Country/Region', 'Lat', 'Long', '1/22/20']
```

我们已经得知康复数据相比确诊和死亡数据要缺失样本，我们需要找出有哪些缺失

```
print(confirmed_data[['Country/Region','Province/State']])
print(recovered_data[['Country/Region','Province/State']])
```

```
         Country/Region Province/State
0           Afghanistan            NaN
1               Albania            NaN
2               Algeria            NaN
3               Andorra            NaN
4                Angola            NaN
..                  ...            ...
269             Vietnam            NaN
270  West Bank and Gaza            NaN
271               Yemen            NaN
272              Zambia            NaN
273            Zimbabwe            NaN

[274 rows x 2 columns]
         Country/Region Province/State
0           Afghanistan            NaN
1               Albania            NaN
2               Algeria            NaN
3               Andorra            NaN
4                Angola            NaN
..                  ...            ...
254             Vietnam            NaN
255  West Bank and Gaza            NaN
256               Yemen            NaN
257              Zambia            NaN
258            Zimbabwe            NaN

[259 rows x 2 columns]
```

```
# 提取关键地理信息列
geo_cols = ['Country/Region', 'Province/State']
geo_confirmed = confirmed_data[geo_cols]
geo_recovered = recovered_data[geo_cols]

# 定义通用差异查找函数
def find_diff(left, right):
    return left.merge(right, on=geo_cols, how='left', indicator=True)\
              .query('_merge == "left_only"')\
              .drop(columns='_merge')

# 查找双向差异
print("确诊数据有但康复数据缺失的行:\n", find_diff(geo_confirmed, geo_recovered))
print("\n康复数据有但确诊数据缺失的行:\n", find_diff(geo_recovered, geo_confirmed))

# 其他检查（保持原功能）
checks = [
    ("国家级别缺失", set(confirmed_data['Country/Region']) -
set(recovered_data['Country/Region'])),
    ("省份缺失示例",
confirmed_data[~confirmed_data['Province/State'].isin(recovered_data['Province/Sta
te'])][geo_cols].head()),
    ("缺失日期", sorted(set(confirmed_data.columns[4:]) -
set(recovered_data.columns[4:]))[:5]),
    ("加拿大记录数", len(recovered_data[recovered_data['Country/Region'] ==
'Canada']))
]

for desc, result in checks:
    print(f"\n{desc}: {result}")
```

```
确诊数据有但康复数据缺失的行:
    Country/Region              Province/State
39          Canada                     Alberta
40          Canada            British Columbia
41          Canada            Diamond Princess
42          Canada              Grand Princess
43          Canada                    Manitoba
44          Canada               New Brunswick
45          Canada   Newfoundland and Labrador
46          Canada       Northwest Territories
47          Canada                 Nova Scotia
48          Canada                     Nunavut
49          Canada                     Ontario
50          Canada        Prince Edward Island
51          Canada                      Quebec
52          Canada      Repatriated Travellers
53          Canada                Saskatchewan
54          Canada                       Yukon
```

```
康复数据有但确诊数据缺失的行：
   Country/Region Province/State
39         Canada            NaN

国家级别缺失: set()

省份缺失示例:     Country/Region    Province/State
39         Canada           Alberta
40         Canada  British Columbia
41         Canada  Diamond Princess
42         Canada    Grand Princess
43         Canada          Manitoba

缺失日期: []

加拿大记录数: 1
```

从结果上看，康复数据与死亡数据和确诊数据存在样本数量差距的最主要原因是康复数据统计中只统计了加拿大全国的数据，因此缺少省份和地区的详细数据，这一点我们在后面的分析中要根据这个情况对其余两个数据进行处理，只考虑加拿大全国的情况以保证数据的准确性。

接下来我们对美国的名称格式化，把地区和国家的错误地方进行修改，方便后续分析。

```python
# 美国的名称格式化
confirmed_data['Country/Region']=confirmed_data['Country/Region'].apply(lambda x: 'United States' if x == 'US' else x)
deaths_data['Country/Region']=deaths_data['Country/Region'].apply(lambda x: 'United States' if x == 'US' else x)
recovered_data['Country/Region']=recovered_data['Country/Region'].apply(lambda x: 'United States' if x == 'US' else x)
```

```python
# 将台湾的数据归到中国（这一块优化了代码的执行效率，定义函数，对每个数据集执行了数据替换操作）
from pandas.plotting import table


def correct_taiwan_data(df):
    taiwan_mask = df['Country/Region'].str.contains('Taiwan', na=False)
    if taiwan_mask.any():
        df.loc[taiwan_mask, 'Country/Region'] = 'China'
        df.loc[taiwan_mask, 'Province/State'] = 'Taiwan'
    return df

confirmed_data = correct_taiwan_data(confirmed_data)
deaths_data = correct_taiwan_data(deaths_data)
recovered_data = correct_taiwan_data(recovered_data)
print(confirmed_data['Country/Region'].str.contains('Taiwan', na=False))
```

```
0      False
1      False
2      False
3      False
4      False
       ...
269    False
270    False
271    False
272    False
273    False
Name: Country/Region, Length: 274, dtype: bool
```

```
# 验证修改结果
for df, name in [(confirmed_data, '确诊'), (deaths_data, '死亡'), (recovered_data, '康复')]:
    taiwan_in_china = df[(df['Country/Region'] == 'China') & (df['Province/State'] == 'Taiwan')]
    print(f"{name}数据集中包含{taiwan_in_china.shape[0]}条“国家：中国、地区：台湾”数据")
```

```
确诊数据集中包含1条“国家：中国、地区：台湾”数据
死亡数据集中包含1条“国家：中国、地区：台湾”数据
康复数据集中包含1条“国家：中国、地区：台湾”数据
```

接下来对所有的地区和国家英文名称替换为中文名称，采用字典的方式，apply函数，lambda函数，字典的get方法，来实现。

```
# 增加 Country/Region 和 Province/State 的中文冗余列 Country/Region_zh 、Province/State_zh
country_map = {
    'Singapore Rep.': '新加坡', 'Dominican Rep.': '多米尼加', 'Palestine': '巴勒斯坦', 'Bahamas': '巴哈马', 'Timor-Leste': '东帝汶',
    'Afghanistan': '阿富汗', 'Guinea-Bissau': '几内亚比绍', "Côte d'Ivoire": '科特迪瓦', 'Siachen Glacier': '锡亚琴冰川',
    "Br. Indian Ocean Ter.": '英属印度洋领土', 'Angola': '安哥拉', 'Albania': '阿尔巴尼亚', 'United Arab Emirates': '阿联酋',
    'Argentina': '阿根廷', 'Armenia': '亚美尼亚', 'French Southern and Antarctic Lands': '法属南半球和南极领地', 'Australia': '澳大利亚',
    'Austria': '奥地利', 'Azerbaijan': '阿塞拜疆', 'Burundi': '布隆迪', 'Belgium': '比利时', 'Benin': '贝宁', 'Burkina Faso': '布基纳法索',
    'Bangladesh': '孟加拉国', 'Bulgaria': '保加利亚', 'The Bahamas': '巴哈马', 'Bosnia and Herz.': '波斯尼亚和黑塞哥维那', 'Belarus': '白俄罗斯',
    'Belize': '伯利兹', 'Bermuda': '百慕大', 'Bolivia': '玻利维亚', 'Brazil': '巴西', 'Brunei': '文莱', 'Bhutan': '不丹',
    'Botswana': '博茨瓦纳', 'Central African Rep.': '中非', 'Canada': '加拿大', 'Switzerland': '瑞士', 'Chile': '智利',
    'China': '中国', 'Ivory Coast': '象牙海岸', 'Cameroon': '喀麦隆', 'Dem. Rep. Congo': '刚果民主共和国', 'Congo': '刚果',
    'Colombia': '哥伦比亚', 'Costa Rica': '哥斯达黎加', 'Cuba': '古巴', 'N. Cyprus': '北塞浦路斯', 'Cyprus': '塞浦路斯', 'Czech Rep.': '捷克',
    'Germany': '德国', 'Djibouti': '吉布提', 'Denmark': '丹麦', 'Algeria': '阿尔及利亚', 'Ecuador': '厄瓜多尔', 'Egypt': '埃及',
    'Eritrea': '厄立特里亚', 'Spain': '西班牙', 'Estonia': '爱沙尼亚', 'Ethiopia': '埃塞俄比亚', 'Finland': '芬兰', 'Fiji': '斐',
    'Falkland Islands': '福克兰群岛', 'France': '法国', 'Gabon': '加蓬', 'United Kingdom': '英国', 'Georgia': '格鲁吉亚',
    'Ghana': '加纳', 'Guinea': '几内亚', 'Gambia': '冈比亚', 'Guinea Bissau': '几内亚比绍', 'Eq. Guinea': '赤道几内亚', 'Greece': '希腊',
    'Greenland': '格陵兰', 'Guatemala': '危地马拉', 'French Guiana': '法属圭亚那', 'Guyana': '圭亚那', 'Honduras': '洪都拉斯',
    'Croatia': '克罗地亚', 'Haiti': '海地', 'Hungary': '匈牙利', 'Indonesia': '印度尼西亚', 'India': '印度', 'Ireland': '爱尔兰',
    'Iran': '伊朗', 'Iraq': '伊拉克', 'Iceland': '冰岛', 'Israel': '以色列', 'Italy': '意大利', 'Jamaica': '牙买加', 'Jordan': '约旦',
    'Japan': '日本', 'Kazakhstan': '哈萨克斯坦', 'Kenya': '肯尼亚', 'Kyrgyzstan': '吉尔吉斯斯坦', 'Cambodia': '柬埔寨', 'Korea': '韩国',
    'Kosovo': '科索沃', 'Kuwait': '科威特', 'Lao PDR': '老挝', 'Lebanon': '黎巴嫩', 'Liberia': '利比里亚', 'Libya': '利比亚',
    'Sri Lanka': '斯里兰卡', 'Lesotho': '莱索托', 'Lithuania': '立陶宛', 'Luxembourg': '卢森堡', 'Latvia': '拉脱维亚', 'Morocco': '摩洛哥',
    'Moldova': '摩尔多瓦', 'Madagascar': '马达加斯加', 'Mexico': '墨西哥', 'Macedonia': '马其顿', 'Mali': '马里', 'Myanmar': '缅甸',
    'Montenegro': '黑山', 'Mongolia': '蒙古', 'Mozambique': '莫桑比克', 'Mauritania': '毛里塔尼亚', 'Malawi': '马拉维',
    'Malaysia': '马来西亚', 'Namibia': '纳米比亚', 'New Caledonia': '新喀里多尼亚', 'Niger': '尼日尔', 'Nigeria': '尼日利亚',
    'Nicaragua': '尼加拉瓜', 'Netherlands': '荷兰', 'Norway': '挪威', 'Nepal': '尼泊尔', 'New Zealand': '新西兰', 'Oman': '阿曼',
    'Pakistan': '巴基斯坦', 'Panama': '巴拿马', 'Peru': '秘鲁', 'Philippines': '菲律宾', 'Papua New Guinea': '巴布亚新几内亚',
    'Poland': '波兰', 'Puerto Rico': '波多黎各', 'Dem. Rep. Korea': '朝鲜', 'Portugal': '葡萄牙', 'Paraguay': '巴拉圭',
    'Qatar': '卡塔尔', 'Romania': '罗马尼亚', 'Russia': '俄罗斯', 'Rwanda': '卢旺达', 'W. Sahara': '西撒哈拉', 'Saudi Arabia': '沙特阿拉伯',
    'Sudan': '苏丹', 'S. Sudan': '南苏丹', 'Senegal': '塞内加尔', 'Solomon Is.': '所罗门群岛', 'Sierra Leone': '塞拉利昂',
    'El Salvador': '萨尔瓦多', 'Somaliland': '索马里兰', 'Somalia': '索马里', 'Serbia': '塞尔维亚', 'Suriname': '苏里南',
    'Slovakia': '斯洛伐克', 'Slovenia': '斯洛文尼亚', 'Sweden': '瑞典', 'Swaziland': '斯威士兰', 'Syria': '叙利亚', 'Chad': '乍得',
    'Togo': '多哥', 'Thailand': '泰国', 'Tajikistan': '塔吉克斯坦', 'Turkmenistan': '土库曼斯坦', 'East Timor': '东帝汶',
    'Trinidad and Tobago': '特里尼达和多巴哥', 'Tunisia': '突尼斯', 'Turkey': '土耳其', 'Tanzania': '坦桑尼亚', 'Uganda': '乌干达',
    'Ukraine': '乌克兰', 'Uruguay': '乌拉圭', 'United States': '美国', 'Uzbekistan': '乌兹别克斯坦', 'Venezuela': '委内瑞拉',
    'Vietnam': '越南', 'Vanuatu': '瓦努阿图', 'West Bank': '西岸', 'Yemen': '也门',
'South Africa': '南非', 'Zambia': '赞比亚',
    'Zimbabwe': '津巴布韦', 'Comoros': '科摩罗'
}

province_map = {
    'Anhui': '安徽', 'Beijing': '北京', 'Chongqing': '重庆', 'Gansu': '甘肃',
'Guangdong': '广东',
    'Guangxi': '广西', 'Guizhou': '贵州', 'Hainan': '海南', 'Hebei': '河北',
'Heilongjiang': '黑龙江', 'Henan': '河南',
    'Hong Kong': '香港', 'Hubei': '湖北', 'Hunan': '湖南', 'Inner Mongolia': '内蒙
古', 'Jiangsu': '江苏',
    'Jiangxi': '江西', 'Jilin': '吉林', 'Liaoning': '辽宁', 'Macau': '澳门',
'Ningxia': '宁夏', 'Qinghai': '青海',
    'Shaanxi': '陕西', 'Shandong': '山东', 'Shanghai': '上海', 'Shanxi': '山西',
'Sichuan': '四川', 'Tianjin': '天津',
    'Tibet': '西藏', 'Xinjiang': '新疆', 'Yunnan': '云南', 'Zhejiang': '浙江',
'Fujian':'福建', 'Taiwan': '台湾'
}
#这里get函数的作用是，如果key在字典中存在，则返回对应的值，否则原本的键。
confirmed_data['Country/Region_zh'] =
confirmed_data['Country/Region'].apply(lambda x: country_map.get(x, x))
deaths_data['Country/Region_zh'] = deaths_data['Country/Region'].apply(lambda x:
country_map.get(x, x))
recovered_data['Country/Region_zh'] =
recovered_data['Country/Region'].apply(lambda x: country_map.get(x, x))

confirmed_data['Province/State_zh'] =
confirmed_data['Province/State'].apply(lambda x: province_map.get(x, x))
deaths_data['Province/State_zh'] = deaths_data['Province/State'].apply(lambda x:
province_map.get(x, x))
recovered_data['Province/State_zh'] =
recovered_data['Province/State'].apply(lambda x: province_map.get(x, x))
```

增加代码：检测是否修改列名顺序成功

```
# 打印调整前的列名
print("【调整前列结构】")
print(f"确诊数据列名({len(confirmed_data.columns)}列):
{confirmed_data.columns.tolist()}")
print(f"死亡数据列名({len(deaths_data.columns)}列): {deaths_data.columns.tolist()
[:5]}...")
print(f"康复数据列名({len(recovered_data.columns)}列):
{recovered_data.columns.tolist()[:5]}...\n")
```

```
【调整前列结构】
确诊数据列名(430列): ['Province/State', 'Country/Region', 'Lat', 'Long', '1/22/20',
    '1/23/20', '1/24/20', '1/25/20', '1/26/20', '1/27/20', '1/28/20', '1/29/20',
    '1/30/20', '1/31/20', '2/1/20', '2/2/20', '2/3/20', '2/4/20', '2/5/20', '2/6/20',
    '2/7/20', '2/8/20', '2/9/20', '2/10/20', '2/11/20', '2/12/20', '2/13/20',
    '2/14/20', '2/15/20', '2/16/20', '2/17/20', '2/18/20', '2/19/20', '2/20/20',
    '2/21/20', '2/22/20', '2/23/20', '2/24/20', '2/25/20', '2/26/20', '2/27/20',
    '2/28/20', '2/29/20', '3/1/20', '3/2/20', '3/3/20', '3/4/20', '3/5/20', '3/6/20',
    '3/7/20', '3/8/20', '3/9/20', '3/10/20', '3/11/20', '3/12/20', '3/13/20',
    '3/14/20', '3/15/20', '3/16/20', '3/17/20', '3/18/20', '3/19/20', '3/20/20',
    '3/21/20', '3/22/20', '3/23/20', '3/24/20', '3/25/20', '3/26/20', '3/27/20',
    '3/28/20', '3/29/20', '3/30/20', '3/31/20', '4/1/20', '4/2/20', '4/3/20',
    '4/4/20', '4/5/20', '4/6/20', '4/7/20', '4/8/20', '4/9/20', '4/10/20', '4/11/20',
    '4/12/20', '4/13/20', '4/14/20', '4/15/20', '4/16/20', '4/17/20', '4/18/20',
    '4/19/20', '4/20/20', '4/21/20', '4/22/20', '4/23/20', '4/24/20', '4/25/20',
    '4/26/20', '4/27/20', '4/28/20', '4/29/20', '4/30/20', '5/1/20', '5/2/20',
    '5/3/20', '5/4/20', '5/5/20', '5/6/20', '5/7/20', '5/8/20', '5/9/20', '5/10/20',
    '5/11/20', '5/12/20', '5/13/20', '5/14/20', '5/15/20', '5/16/20', '5/17/20',
    '5/18/20', '5/19/20', '5/20/20', '5/21/20', '5/22/20', '5/23/20', '5/24/20',
    '5/25/20', '5/26/20', '5/27/20', '5/28/20', '5/29/20', '5/30/20', '5/31/20',
    '6/1/20', '6/2/20', '6/3/20', '6/4/20', '6/5/20', '6/6/20', '6/7/20', '6/8/20',
    '6/9/20', '6/10/20', '6/11/20', '6/12/20', '6/13/20', '6/14/20', '6/15/20',
    '6/16/20', '6/17/20', '6/18/20', '6/19/20', '6/20/20', '6/21/20', '6/22/20',
    '6/23/20', '6/24/20', '6/25/20', '6/26/20', '6/27/20', '6/28/20', '6/29/20',
    '6/30/20', '7/1/20', '7/2/20', '7/3/20', '7/4/20', '7/5/20', '7/6/20', '7/7/20',
    '7/8/20', '7/9/20', '7/10/20', '7/11/20', '7/12/20', '7/13/20', '7/14/20',
    '7/15/20', '7/16/20', '7/17/20', '7/18/20', '7/19/20', '7/20/20', '7/21/20',
    '7/22/20', '7/23/20', '7/24/20', '7/25/20', '7/26/20', '7/27/20', '7/28/20',
    '7/29/20', '7/30/20', '7/31/20', '8/1/20', '8/2/20', '8/3/20', '8/4/20', '8/5/20',
    '8/6/20', '8/7/20', '8/8/20', '8/9/20', '8/10/20', '8/11/20', '8/12/20',
    '8/13/20', '8/14/20', '8/15/20', '8/16/20', '8/17/20', '8/18/20', '8/19/20',
    '8/20/20', '8/21/20', '8/22/20', '8/23/20', '8/24/20', '8/25/20', '8/26/20',
    '8/27/20', '8/28/20', '8/29/20', '8/30/20', '8/31/20', '9/1/20', '9/2/20',
    '9/3/20', '9/4/20', '9/5/20', '9/6/20', '9/7/20', '9/8/20', '9/9/20', '9/10/20',
    '9/11/20', '9/12/20', '9/13/20', '9/14/20', '9/15/20', '9/16/20', '9/17/20',
    '9/18/20', '9/19/20', '9/20/20', '9/21/20', '9/22/20', '9/23/20', '9/24/20',
    '9/25/20', '9/26/20', '9/27/20', '9/28/20', '9/29/20', '9/30/20', '10/1/20',
    '10/2/20', '10/3/20', '10/4/20', '10/5/20', '10/6/20', '10/7/20', '10/8/20',
    '10/9/20', '10/10/20', '10/11/20', '10/12/20', '10/13/20', '10/14/20', '10/15/20',
    '10/16/20', '10/17/20', '10/18/20', '10/19/20', '10/20/20', '10/21/20',
    '10/22/20', '10/23/20', '10/24/20', '10/25/20', '10/26/20', '10/27/20',
    '10/28/20', '10/29/20', '10/30/20', '10/31/20', '11/1/20', '11/2/20', '11/3/20',
    '11/4/20', '11/5/20', '11/6/20', '11/7/20', '11/8/20', '11/9/20', '11/10/20',
    '11/11/20', '11/12/20', '11/13/20', '11/14/20', '11/15/20', '11/16/20',
    '11/17/20', '11/18/20', '11/19/20', '11/20/20', '11/21/20', '11/22/20',
    '11/23/20', '11/24/20', '11/25/20', '11/26/20', '11/27/20', '11/28/20',
    '11/29/20', '11/30/20', '12/1/20', '12/2/20', '12/3/20', '12/4/20', '12/5/20',
    '12/6/20', '12/7/20', '12/8/20', '12/9/20', '12/10/20', '12/11/20', '12/12/20',
    '12/13/20', '12/14/20', '12/15/20', '12/16/20', '12/17/20', '12/18/20',
    '12/19/20', '12/20/20', '12/21/20', '12/22/20', '12/23/20', '12/24/20',
    '12/25/20', '12/26/20', '12/27/20', '12/28/20', '12/29/20', '12/30/20',
    '12/31/20', '1/1/21', '1/2/21', '1/3/21', '1/4/21', '1/5/21', '1/6/21', '1/7/21',
    '1/8/21', '1/9/21', '1/10/21', '1/11/21', '1/12/21', '1/13/21', '1/14/21',
    '1/15/21', '1/16/21', '1/17/21', '1/18/21', '1/19/21', '1/20/21', '1/21/21',
    '1/22/21', '1/23/21', '1/24/21', '1/25/21', '1/26/21', '1/27/21', '1/28/21',
    '1/29/21', '1/30/21', '1/31/21', '2/1/21', '2/2/21', '2/3/21', '2/4/21', '2/5/21',
    '2/6/21', '2/7/21', '2/8/21', '2/9/21', '2/10/21', '2/11/21', '2/12/21',
    '2/13/21', '2/14/21', '2/15/21', '2/16/21', '2/17/21', '2/18/21', '2/19/21',
    '2/20/21', '2/21/21', '2/22/21', '2/23/21', '2/24/21', '2/25/21', '2/26/21',
    '2/27/21', '2/28/21', '3/1/21', '3/2/21', '3/3/21', '3/4/21', '3/5/21', '3/6/21',
    '3/7/21', '3/8/21', '3/9/21', '3/10/21', '3/11/21', '3/12/21', '3/13/21',
    '3/14/21', '3/15/21', '3/16/21', '3/17/21', '3/18/21', '3/19/21', '3/20/21',
    'Country/Region_zh', 'Province/State_zh']
    死亡数据列名(430列): ['Province/State', 'Country/Region', 'Lat', 'Long',
    '1/22/20']...
    康复数据列名(430列): ['Province/State', 'Country/Region', 'Lat', 'Long',
    '1/22/20']...
```

```
# 调整字段顺序，目的是让中文名称列显示在最前面
confirmed_data = confirmed_data[['Province/State_zh', 'Country/Region_zh'] +
confirmed_data.columns[:-2].to_list()]
deaths_data = deaths_data[['Province/State_zh', 'Country/Region_zh'] +
deaths_data.columns[:-2].to_list()]
recovered_data = recovered_data[['Province/State_zh', 'Country/Region_zh'] +
recovered_data.columns[:-2].to_list()]
```

```
# 打印验证结果
print("【调整后验证】")
print(f"确诊数据新列顺序({len(confirmed_data.columns)}列):
{confirmed_data.columns.tolist()[:10]}...")
print(f"死亡数据新列顺序({len(deaths_data.columns)}列):
{deaths_data.columns.tolist()[:5]}...")
print(f"康复数据新列顺序({len(recovered_data.columns)}列):
{recovered_data.columns.tolist()[:5]}...")
print("\n验证结果：中文列应位于最前，且总列数保持不变")
```

```
【调整后验证】
确诊数据新列顺序(430列): ['Province/State_zh', 'Country/Region_zh',
'Province/State', 'Country/Region', 'Lat', 'Long', '1/22/20', '1/23/20',
'1/24/20', '1/25/20']...
死亡数据新列顺序(430列): ['Province/State_zh', 'Country/Region_zh',
'Province/State', 'Country/Region', 'Lat']...
康复数据新列顺序(430列): ['Province/State_zh', 'Country/Region_zh',
'Province/State', 'Country/Region', 'Lat']...

验证结果：中文列应位于最前，且总列数保持不变
```

```
print("【调整后验证】")
print(f"确诊数据新列顺序({len(confirmed_data.columns)}列):
{confirmed_data.columns.tolist()[:5]}...")
print(f"死亡数据新列顺序({len(deaths_data.columns)}列):
{deaths_data.columns.tolist()[:5]}...")
print(f"康复数据新列顺序({len(recovered_data.columns)}列):
{recovered_data.columns.tolist()[:5]}...")
print("\n验证结果：中文列应位于最前，且总列数保持不变")
```

```
【调整后验证】
确诊数据新列顺序(430列): ['Province/State_zh', 'Country/Region_zh',
'Province/State', 'Country/Region', 'Lat']...
死亡数据新列顺序(430列): ['Province/State_zh', 'Country/Region_zh',
'Province/State', 'Country/Region', 'Lat']...
康复数据新列顺序(430列): ['Province/State_zh', 'Country/Region_zh',
'Province/State', 'Country/Region', 'Lat']...

验证结果：中文列应位于最前，且总列数保持不变
```

3、数据分析可视化

3.1 全球新冠疫情情况

3.1.1 全球疫情现状

```
from pyecharts.charts import Map, Timeline, Bar, Line
from pyecharts.components import Table
from pyecharts.options import ComponentTitleOpts
from pyecharts import options as opts  # <-- 关键导入

lastdate = confirmed_data.columns[-1]
confirmed_total = confirmed_data[lastdate].sum()
deaths_total = deaths_data[lastdate].sum()
recovered_total = recovered_data[lastdate].sum()
deaths_rate = deaths_total / confirmed_total
recovered_rate = recovered_total / confirmed_total

table = Table()

headers = ['确诊人数', '死亡人数', '治愈人数', '死亡率', '治愈率']
rows = [
    [confirmed_total, deaths_total, recovered_total, f'{deaths_rate:.2%}',
f'{recovered_rate:.2%}'],
]
```

```
table.add(headers, rows)
table.set_global_opts(
    title_opts=ComponentTitleOpts(title=f'({lastdate})全球疫情情况', subtitle='由于数据集没有美国的治愈数据，所以治愈人数和治愈率都远低于实际，等待数据集完善')
)
table.render_notebook()
```

```
        <style>
        .fl-table {
            margin: 20px;
            border-radius: 5px;
            font-size: 12px;
            border: none;
            border-collapse: collapse;
            max-width: 100%;
            white-space: nowrap;
            word-break: keep-all;
        }

        .fl-table th {
            text-align: left;
            font-size: 20px;
        }

        .fl-table tr {
            display: table-row;
            vertical-align: inherit;
            border-color: inherit;
        }

        .fl-table tr:hover td {
            background: #00d1b2;
            color: #F8F8F8;
        }

        .fl-table td, .fl-table th {
            border-style: none;
            border-top: 1px solid #dbdbdb;
            border-left: 1px solid #dbdbdb;
            border-bottom: 3px solid #dbdbdb;
            border-right: 1px solid #dbdbdb;
            padding: .5em .55em;
            font-size: 15px;
        }

        .fl-table td {
            border-style: none;
            font-size: 15px;
            vertical-align: center;
            border-bottom: 1px solid #dbdbdb;
            border-left: 1px solid #dbdbdb;
            border-right: 1px solid #dbdbdb;
            height: 30px;
        }

        .fl-table tr:nth-child(even) {
            background: #F8F8F8;
        }
    </style>
    <div id="55a3f6a01d984493a50b23c9bc3784eb" class="chart-container" style="">
        <p class="title" style="font-size: 18px; font-weight:bold;" > (3/20/21)全
球疫情情况</p>
        <p class="subtitle" style="font-size: 12px;" > 由于数据集没有美国的治愈数
据，所以治愈人数和治愈率都远低于实际，等待数据集完善</p>
        <table class="fl-table">
<thead>
    <tr>
        <th>确诊人数</th>
        <th>死亡人数</th>
        <th>治愈人数</th>
        <th>死亡率</th>
        <th>治愈率</th>
    </tr>
</thead>
<tbody>
    <tr>
        <td>122813796</td>
        <td>2709639</td>
        <td>69523087</td>
        <td>2.21%</td>
        <td>56.61%</td>
    </tr>
</tbody>
```

```
# 在confirmed定义前添加
lastdate = confirmed_data.columns[-1]  # 自动获取最新日期


confirmed = confirmed_data.groupby('Country/Region').agg({lastdate:
'sum'}).to_dict()[lastdate]
deaths = deaths_data.groupby('Country/Region').agg({lastdate: 'sum'}).to_dict()
[lastdate]
recovered = recovered_data.groupby('Country/Region').agg({lastdate:
'sum'}).to_dict()[lastdate]
exists_confirmed = {key: value - deaths[key] - recovered[key]  for key, value in
confirmed.items()}

# 添加数据有效性校验
exists_confirmed = {k: max(v, 0) for k, v in exists_confirmed.items()}  # 确保现有
```

```
确诊不为负数

c = (
    Map()
    .add("确诊人数", [*confirmed.items()], "world", is_map_symbol_show=False)
    .add("治愈人数", [*recovered.items()], "world", is_map_symbol_show=False)
    .add("死亡人数", [*deaths.items()], "world", is_map_symbol_show=False)
    .add("现有确诊人数", [*exists_confirmed.items()], "world",
is_map_symbol_show=False)
    .set_series_opts(label_opts=opts.LabelOpts(is_show=False))
    .set_global_opts(
        title_opts=opts.TitleOpts(
            title=f'({lastdate})全球疫情现状',
            subtitle='由于数据源没有美国的治愈数据,\n所以美国的现有确诊人数并不准确'
        ),
        visualmap_opts=opts.VisualMapOpts(
            max_=max(confirmed.values()),
            is_piecewise=True,
            range_text=['高', '低'],
            pieces=[  # 添加明确的分段区间
                {"min": 100000, "label": ">10万"},
                {"min": 50000, "max": 99999, "label": "5-10万"},
                {"min": 10000, "max": 49999, "label": "1-5万"},
                {"min": 1000, "max": 9999, "label": "1千-1万"},
                {"min": 0, "max": 999, "label": "<1千"}
            ]
        )
    )
)
c.render_notebook()
```

```
    <div id="378d2c372d074f0490a0f8e7bf108926" style="width:900px; height:500px;">
</div>
```

```
#备份单元格
confirmed = confirmed_data.groupby('Country/Region').agg({lastdate:
'sum'}).to_dict()[lastdate]
deaths = deaths_data.groupby('Country/Region').agg({lastdate: 'sum'}).to_dict()
[lastdate]
recovered = recovered_data.groupby('Country/Region').agg({lastdate:
'sum'}).to_dict()[lastdate]
exists_confirmed = {key: value - deaths[key] - recovered[key]  for key, value in
confirmed.items()}
c = (
    Map()
    .add("确诊人数", [*confirmed.items()], "world", is_map_symbol_show=False)
    .add("治愈人数", [*recovered.items()], "world", is_map_symbol_show=False)
    .add("死亡人数", [*deaths.items()], "world", is_map_symbol_show=False)
    .add("现有确诊人数", [*exists_confirmed.items()], "world",
is_map_symbol_show=False)
    .set_series_opts(label_opts=opts.LabelOpts(is_show=False))
    .set_global_opts(
        title_opts=opts.TitleOpts(title=f'({lastdate})全球疫情现状', subtitle='由于
数据源没有美国的治愈数据,\n所以美国的现有确诊人数并不准确'),
        visualmap_opts=opts.VisualMapOpts(max_=200000),

    )
    .set_global_opts(
        title_opts=opts.TitleOpts(title=f'({lastdate})全球疫情现状', subtitle='由于
数据源没有美国的治愈数据,\n所以美国的现有确诊人数并不准确'),
        visualmap_opts=opts.VisualMapOpts(
            max_=max(confirmed.values()),  # 动态计算最大值
            is_piecewise=True,  # 启用分段式视觉映射
            range_text=['高', '低'],
        ),
    )
)
c.render_notebook()
```

```
    <div id="d71186b23c384c91bde4e92c08407e85" style="width:900px; height:500px;">
</div>
```

3.1.2 全球疫情历史发展情况

```
tl = Timeline()
tl.add_schema(
#         is_auto_play=True,
        is_loop_play=False,
        play_interval=200,
    )
target = confirmed_data.columns[6:].to_list()
target.reverse()
target = target[::7]
target.reverse()
for dt in target:
    confirmed = confirmed_data.groupby('Country/Region').agg({dt:
'sum'}).to_dict()[dt]
    c = (
        Map()
        .add("确诊人数", [*confirmed.items()], "world", is_map_symbol_show=False)
        .set_series_opts(label_opts=opts.LabelOpts(is_show=False))
        .set_global_opts(
            title_opts=opts.TitleOpts(title="全球疫情历史发展情况"),
            visualmap_opts=opts.VisualMapOpts(max_=200000),

        )
    )
    tl.add(c, dt)
tl.render_notebook()
```

3.1.3 各国确诊人数 TOP20 排行

```
tl = Timeline()
tl.add_schema(
#         is_auto_play=True,
        is_loop_play=False,
        play_interval=100,
    )

for dt in confirmed_data.columns[6:]:
    confirmed = confirmed_data.groupby('Country/Region_zh').agg({dt:
'sum'}).sort_values(by=dt, ascending=False)[:20].sort_values(by=dt).to_dict()[dt]
    bar = (
        Bar()
        .add_xaxis([*confirmed.keys()])
        .add_yaxis("确诊人数", [*confirmed.values()],
label_opts=opts.LabelOpts(position="right"))
        .reversal_axis()
        .set_global_opts(
            title_opts=opts.TitleOpts("各国确诊人数排行 TOP20")
        )
    )
    tl.add(bar, dt)
tl.render_notebook()
```

3.1.4 全球疫情趋势

```
targets = confirmed_data.columns[6:].to_list()
new_confirmed_list = []
new_deaths_list = []
new_recovered_list = []
exists_confirmed_list = []
for idx, today in enumerate(targets[1:], 1):
    yesterday = targets[idx-1]
    new_confirmed = confirmed_data[today].sum() - confirmed_data[yesterday].sum()
    new_deaths = deaths_data[today].sum() - deaths_data[yesterday].sum()
    new_recovered = recovered_data[today].sum() - recovered_data[yesterday].sum()
    exists_confirmed = confirmed_data[today].sum() - deaths_data[today].sum() -
recovered_data[today].sum()
    new_confirmed_list.append(int(new_confirmed))
    new_deaths_list.append(int(new_deaths))
    new_recovered_list.append(int(new_recovered))
    # 由于数据集中没有美国的治愈数据，所以在统计全球的现有确诊人员和治愈率的时候会有很大
误差，代码里面先不做这个处理，期待数据集的完善
    exists_confirmed_list.append(int(exists_confirmed))
```

```
c = (
    Line()
    .add_xaxis(targets[1:])
    .add_yaxis('新增确诊人数', new_confirmed_list,
label_opts=opts.LabelOpts(is_show=False), is_symbol_show=False)
    .add_yaxis('新增治愈人数', new_recovered_list,
label_opts=opts.LabelOpts(is_show=False), is_symbol_show=False)
    .add_yaxis('新增死亡人数', new_deaths_list,
label_opts=opts.LabelOpts(is_show=False), is_symbol_show=False)
    .add_yaxis('现有确诊人数', exists_confirmed_list,
label_opts=opts.LabelOpts(is_show=False), is_symbol_show=False)
    .set_global_opts(title_opts=opts.TitleOpts(title="全球疫情趋势"))
)
c.render_notebook()
```

3.2 中国新冠疫情情况

3.2.1 中国疫情现状

```
from pyecharts import options as opts
from pyecharts.charts import Map, Timeline, Bar, Line
from pyecharts.components import Table
from pyecharts.options import ComponentTitleOpts

lastdate = confirmed_data.columns[-1]
confirmed_data_china = confirmed_data[confirmed_data['Country/Region'] == 'China']
deaths_data_china = deaths_data[deaths_data['Country/Region'] == 'China']
recovered_data_china = recovered_data[recovered_data['Country/Region'] == 'China']

confirmed_total_china = confirmed_data_china[lastdate].sum()
deaths_total_china = deaths_data_china[lastdate].sum()
recovered_total_china = recovered_data_china[lastdate].sum()
exists_confirmed_china = confirmed_total_china - deaths_total_china -
recovered_total_china

deaths_rate_china = deaths_total_china / confirmed_total_china
recovered_rate_china = recovered_total_china / confirmed_total_china

table = Table()

headers = ['确诊人数', '死亡人数', '治愈人数', '死亡率', '治愈率', '现有确诊人数']
rows = [
    [confirmed_total_china, deaths_total_china, recovered_total_china,
f'{deaths_rate_china:.2%}', f'{recovered_rate_china:.2%}',
exists_confirmed_china],
]
table.add(headers, rows)
table.set_global_opts(
    title_opts=ComponentTitleOpts(title=f'({lastdate})中国疫情情况')
)
table.render_notebook()
```

```
confirmed = confirmed_data_china.groupby('Province/State_zh').agg({lastdate:
'sum'}).to_dict()[lastdate]
deaths = deaths_data_china.groupby('Province/State_zh').agg({lastdate:
'sum'}).to_dict()[lastdate]
recovered = recovered_data_china.groupby('Province/State_zh').agg({lastdate:
'sum'}).to_dict()[lastdate]
exists_confirmed = {key: value - deaths[key] - recovered[key]  for key, value in
confirmed.items()}
c = (
    Map()
    .add("确诊人数", [*confirmed.items()], "china", is_map_symbol_show=False)
    .add("治愈人数", [*recovered.items()], "china", is_map_symbol_show=False)
    .add("死亡人数", [*deaths.items()], "china", is_map_symbol_show=False)
    .add("现有确诊人数", [*exists_confirmed.items()], "china",
is_map_symbol_show=False)
    .set_series_opts(label_opts=opts.LabelOpts(is_show=True))
    .set_global_opts(
        title_opts=opts.TitleOpts(title=f'({lastdate})中国疫情现状'),
        visualmap_opts=opts.VisualMapOpts(max_=1000),
    )
)
c.render_notebook()
```

3.2.2 中国疫情历史发展情况

```
tl = Timeline()
tl.add_schema(
#         is_auto_play=True,
        is_loop_play=False,
        play_interval=200,
    )
target = confirmed_data_china.columns[6:].to_list()
target.reverse()
target = target[::7]
target.reverse()
for dt in target:
    confirmed = confirmed_data_china.groupby('Province/State_zh').agg({dt:
'sum'}).to_dict()[dt]
    c = (
        Map()
        .add("确诊人数", [*confirmed.items()], "china", is_map_symbol_show=False)
        .set_series_opts(label_opts=opts.LabelOpts(is_show=True))
        .set_global_opts(
            title_opts=opts.TitleOpts(title='中国疫情历史发展情况'),
            visualmap_opts=opts.VisualMapOpts(max_=1000),
        )
    )
    tl.add(c, dt)
tl.render_notebook()
```

3.2.3 各省确诊人数排行 TOP20

```
tl = Timeline()
tl.add_schema(
#         is_auto_play=True,
        is_loop_play=False,
        play_interval=100,
    )

for dt in confirmed_data.columns[6:]:
    confirmed = confirmed_data_china.groupby('Province/State_zh').agg({dt:
'sum'}).sort_values(by=dt, ascending=False)[:20].sort_values(by=dt).to_dict()[dt]
    bar = (
        Bar()
        .add_xaxis([*confirmed.keys()])
        .add_yaxis("确诊人数", [*confirmed.values()],
label_opts=opts.LabelOpts(position="right"))
        .reversal_axis()
        .set_global_opts(
            title_opts=opts.TitleOpts("各省确诊人数排行 TOP20")
        )
    )
    tl.add(bar, dt)
tl.render_notebook()
```

3.2.4 中国疫情趋势

```
targets = confirmed_data_china.columns[6:].to_list()
new_confirmed_list = []
new_deaths_list = []
new_recovered_list = []
exists_confirmed_list = []
for idx, today in enumerate(targets[1:], 1):
    yesterday = targets[idx-1]
    new_confirmed = confirmed_data_china[today].sum() -
confirmed_data_china[yesterday].sum()
    new_deaths = deaths_data_china[today].sum() -
deaths_data_china[yesterday].sum()
    new_recovered = recovered_data_china[today].sum() -
recovered_data_china[yesterday].sum()
    exists_confirmed = confirmed_data_china[today].sum() -
deaths_data_china[today].sum() - recovered_data_china[today].sum()
    new_confirmed_list.append(int(new_confirmed))
    new_deaths_list.append(int(new_deaths))
    new_recovered_list.append(int(new_recovered))
    exists_confirmed_list.append(int(exists_confirmed))
c = (
    Line()
    .add_xaxis(targets[1:])
    .add_yaxis('新增确诊人数', new_confirmed_list,
label_opts=opts.LabelOpts(is_show=False), is_symbol_show=False)
    .add_yaxis('新增治愈人数', new_recovered_list,
label_opts=opts.LabelOpts(is_show=False), is_symbol_show=False)
    .add_yaxis('新增死亡人数', new_deaths_list,
label_opts=opts.LabelOpts(is_show=False), is_symbol_show=False)
    .add_yaxis('现有确诊人数', exists_confirmed_list,
label_opts=opts.LabelOpts(is_show=False), is_symbol_show=False)
    .set_global_opts(title_opts=opts.TitleOpts(title="中国疫情趋势"))
)
c.render_notebook()
```