In [1]:
```
import requests
import pandas as pd
from lxml import etree

html = 'https://ncov.dxy.cn/ncovh5/view/pneumonia'
html_data = requests.get(html)
html_data.encoding = 'utf-8'
html_data = etree.HTML(html_data.text, etree.HTMLParser())
html_data = html_data.xpath(
    '//*[@id="getListByCountryTypeService2true"]/text()')  # xpath方法选择疫情的数据集合
ncov_world = html_data[0][49:-12]
ncov_world = ncov_world.replace('true', 'True')
ncov_world = ncov_world.replace('false', 'False')
ncov_world = eval(ncov_world)

country = []
confirmed = []
lived = []
dead = []

for i in ncov_world:  # 分离国家名称，确诊人数，治愈人数和死亡人数并存入dataframe里备用
    country.append(i['provinceName'])
    confirmed.append(i['confirmedCount'])
    lived.append(i['curedCount'])
    dead.append(i['deadCount'])

data_world = pd.DataFrame()
data_world['国家名称'] = country
data_world['确诊人数'] = confirmed
data_world['治愈人数'] = lived
data_world['死亡人数'] = dead
data_world.head(5)
```

Out[1]:

	国家名称	确诊人数	治愈人数	死亡人数
0	法国	29583616	368023	149044
1	德国	26200663	4328400	138781
2	韩国	18053287	336548	24103

	国家名称	确诊人数	治愈人数	死亡人数
3	英国	22455392	6491069	178880
4	西班牙	12311477	150376	106105

In [2]:
```
import pandas as pd
data_world = pd.read_csv('D:\专业课\数据分析技术\data_world.csv')
data_world.head(5)
```

Out[2]:

	国家名称	确诊人数	治愈人数	死亡人数
0	法国	27626578	368023	144130
1	德国	23376879	4328400	132929
2	韩国	16212751	336548	20889
3	英国	21819851	6491069	171560
4	西班牙	11662214	150376	103266

In [3]:
```
data_economy = pd.read_csv(
    "https://labfile.oss.aliyuncs.com/courses/2791/gpd_2016_2020.csv", index_col=0)
time_index = pd.date_range(start='2016', periods=18, freq='Q')
data_economy.index = time_index
data_economy
```

Out[3]:

	国内生产总值	第一产业增加值	第二产业增加值	第三产业增加值	农林牧渔业增加值	工业增加值	制造业增加值	建筑业增加值	批发和零售业增加值	交通运输、仓储和邮政业增加值	住宿和餐饮业增加值	金融业增加值	房地产业增加值	信息传输、软件和信息技术服务业增加值	租赁和商务服务业增加值	其他行业增加值
2016-03-31	162410.0	8312.7	61106.8	92990.5	8665.5	53666.4	45784.0	7763.0	16847.5	7180.5	3181.6	15340.4	11283.0	5128.8	4985.3	28368.
2016-06-30	181408.2	12555.9	73416.5	95435.8	13045.5	60839.2	52378.3	12943.8	17679.8	8295.0	3112.3	14811.7	12209.7	5130.7	5075.1	28265.
2016-09-30	191010.6	17542.4	75400.5	98067.8	18162.2	61902.5	52468.3	13870.6	18513.0	8591.6	3473.2	14945.4	12615.3	4662.3	5452.4	28822.
2016-12-31	211566.2	21728.2	85504.1	104334.0	22577.8	68998.4	58878.4	16921.5	20684.1	8961.6	3840.7	14866.4	13861.4	5202.3	6015.8	29636.
2017-03-31	181867.7	8205.9	69315.5	104346.3	8595.8	60909.3	51419.7	8725.3	18608.9	8094.5	3536.5	16758.8	13047.0	5915.2	5811.9	31864.
2017-06-30	201950.3	12644.9	82323.0	106982.4	13204.2	68099.8	58172.1	14574.4	19473.6	9397.7	3440.9	15856.3	14059.0	5977.9	5868.4	31998.
2017-09-30	212789.3	18255.8	84574.1	109959.5	18944.2	69327.2	58632.6	15590.1	20342.9	9688.7	3838.5	16290.4	14054.9	5539.8	6464.6	32708.
2017-12-31	235428.7	22992.9	95368.0	117067.8	23915.8	76782.9	65652.1	19015.8	22731.1	9940.9	4240.1	15938.8	15925.1	6376.0	7128.4	33433.
2018-03-31	202035.7	8575.7	76598.2	116861.8	9005.8	66905.6	56631.9	10073.8	20485.5	8806.5	3887.8	18050.6	14863.5	7212.2	6879.5	35864.
2018-06-30	223962.2	13003.8	91100.6	119857.8	13662.2	75122.1	64294.9	16404.3	21374.2	10174.9	3779.6	17401.0	16176.1	7309.6	6885.3	35673.
2018-09-30	234474.3	18226.9	93112.5	123134.9	18961.8	76239.6	64348.2	17294.5	22334.1	10582.3	4212.6	17780.6	15914.0	6690.9	7533.3	36930.
2018-12-31	258808.9	24938.7	104023.9	129846.2	25929.0	82822.1	70662.1	21720.4	24710.0	10773.5	4640.6	17378.1	17669.5	7520.8	8170.4	37474.

	国内生产总值	第一产业增加值	第二产业增加值	第三产业增加值	农林牧渔业增加值	工业增加值	制造业增加值	建筑业增加值	批发和零售业增加值	交通运输、仓储和邮政业增加值	住宿和餐饮业增加值	金融业增加值	房地产业增加值	信息传输、软件和信息技术服务业增加值	租赁和商务服务业增加值	其他行业增加值
2019-03-31	218062.8	8769.4	81806.5	127486.9	9249.4	71064.5	60357.1	11143.1	21959.2	9386.6	4234.9	19650.1	15979.2	8424.8	7665.1	39306.(
2019-06-30	242573.8	14437.6	97315.6	130820.6	15108.7	79820.7	68041.8	17954.2	23097.0	10861.3	4123.0	19064.9	17484.4	8395.6	7596.7	39067.3
2019-09-30	252208.7	19798.0	97790.4	134620.4	20629.0	79501.8	66823.8	18734.6	23993.6	11310.2	4610.5	19388.3	17369.0	7528.1	8409.1	40734.5
2019-12-31	278019.7	27461.6	109252.8	141305.2	28579.9	86721.6	73952.4	23072.4	26795.9	11244.0	5071.2	18973.8	18798.9	8341.3	9262.5	41158.3
2020-03-31	206504.3	10186.2	73638.0	122680.1	10708.4	64642.0	53852.0	9377.8	18749.6	7865.1	2820.9	21346.8	15268.3	8928.0	7137.9	39659.(
2020-06-30	250110.1	15866.8	99120.9	135122.3	16596.4	80402.4	69258.8	19156.8	23696.1	10650.0	3481.3	20954.7	18593.6	9573.0	7174.4	39831.4

In [4]:
```
data_area = pd.read_csv('https://labfile.oss.aliyuncs.com/courses/2791/DXYArea.csv')
data_news = pd.read_csv('https://labfile.oss.aliyuncs.com/courses/2791/DXYNews.csv')
```

In [5]:
```
data_area = data_area.loc[data_area['countryName'] == data_area['provinceName']]
data_area_times = data_area[['countryName', 'province_confirmedCount',
                             'province_curedCount', 'province_deadCount', 'updateTime']]

time = pd.DatetimeIndex(data_area_times['updateTime'])  # 根据疫情的更新时间来生成时间序列
data_area_times.index = time  # 生成索引
data_area_times = data_area_times.drop('updateTime', axis=1)
data_area_times.head(5)

data_area_times.isnull().any()  # 查询是否有空值
```

Out[5]:
```
countryName              False
province_confirmedCount  False
province_curedCount      False
province_deadCount       False
dtype: bool
```

In [6]:
```
data_news_times = data_news[['pubDate', 'title', 'summary']]
time = pd.DatetimeIndex(data_news_times['pubDate'])
data_news_times.index = time  # 生成新闻数据的时间索引
data_news_times = data_news_times.drop('pubDate', axis=1)
data_news_times.head(5)
```

Out[6]:

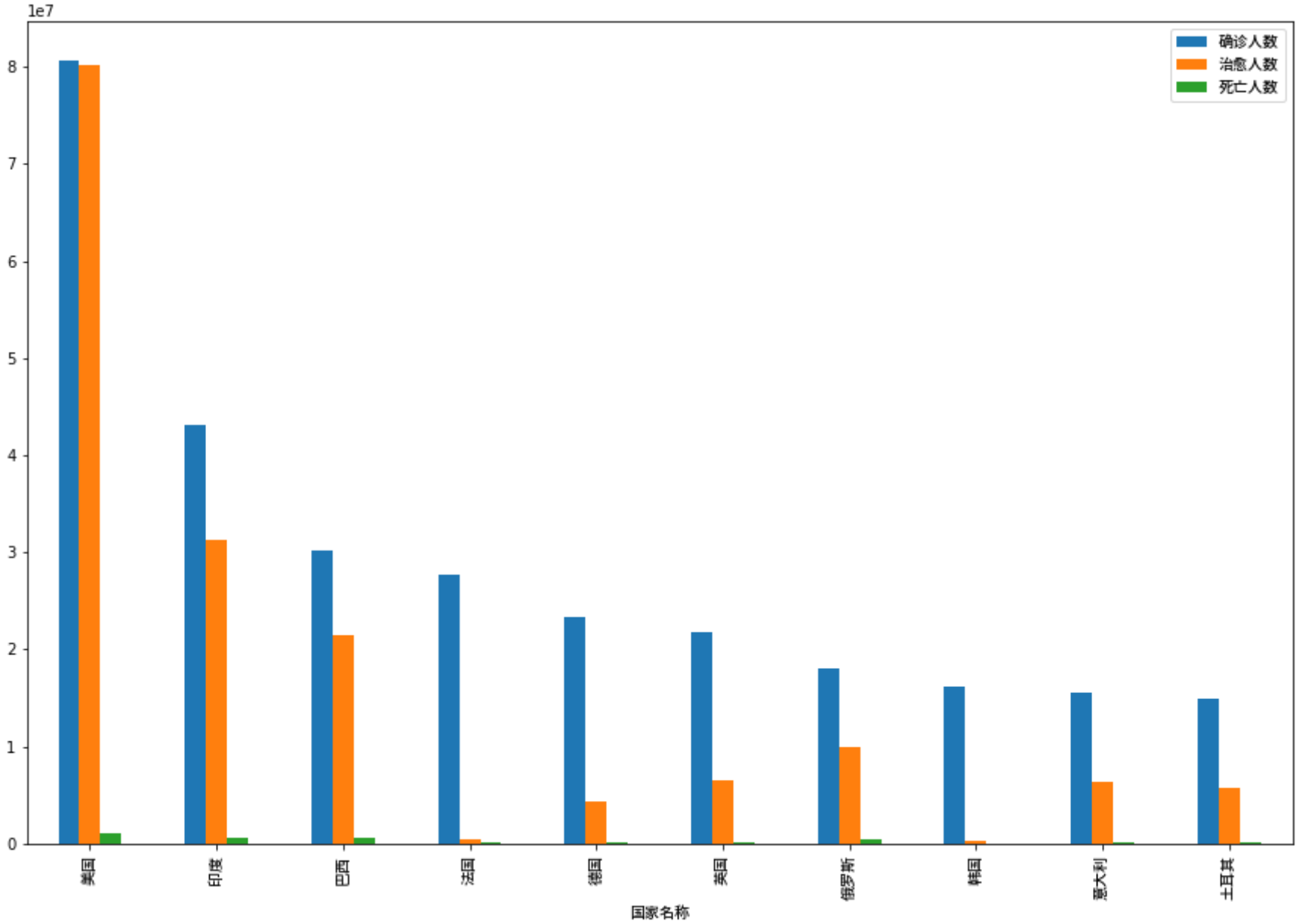
pubDate	title	summary
2020-07-17 05:40:08	美国新增71434例新冠肺炎确诊病例，累计确诊超354万例	据美国约翰斯·霍普金斯大学统计数据显示，截至美东时间7月16日17:33时（北京时间17日0...
2020-07-17 06:06:49	巴西新冠肺炎确诊病例破201万，近六成大城市确诊病例加速增长	截至当地时间7月16日18时，巴西新增新冠肺炎确诊病例45403例，累计确诊2012151例...
2020-07-16 22:31:00	阿塞拜疆新增493例新冠肺炎确诊病例 累计确诊26165例	当地时间7月16日，阿塞拜疆国家疫情防控指挥部发布消息，在过去24小时内，阿塞拜疆新增新冠肺...
2020-07-16 22:29:48	科威特新增791例新冠肺炎确诊病例 累计确诊57668例	科威特卫生部当地时间16日下午发布通告，确认过去24小时境内新增791例新冠肺炎确诊病例，同...
2020-07-16 21:26:54	罗马尼亚新增777例新冠肺炎确诊病例 累计确诊35003例	据罗马尼亚政府7月16日公布的数据，过去24小时对19097人进行新冠病毒检测，确诊777例...

In [7]:
```
print(data_world.isnull().any())
print(data_economy.isnull().any())
print(data_area_times.isnull().any())
print(data_news_times.isnull().any())  # 确认各个数据集是否空集
```

```
国家名称     False
确诊人数     False
治愈人数     False
死亡人数     False
dtype: bool
国内生产总值                False
第一产业增加值               False
第二产业增加值               False
第三产业增加值               False
农林牧渔业增加值              False
工业增加值                 False
制造业增加值                False
建筑业增加值                False
批发和零售业增加值              False
交通运输、仓储和邮政业增加值         False
住宿和餐饮业增加值              False
金融业增加值                False
房地产业增加值                False
信息传输、软件和信息技术服务业增加值    False
租赁和商务服务业增加值             False
其他行业增加值                False
dtype: bool
countryName               False
province_confirmedCount   False
province_curedCount       False
province_deadCount        False
dtype: bool
title      False
summary    False
dtype: bool
```

In [8]:
```
import matplotlib.pyplot as plt
import matplotlib
import os

%matplotlib inline
# 指定中文字体
fpath = os.path.join(r"D:\专业课\数据分析技术\NotoSansCJK.otf")
myfont = matplotlib.font_manager.FontProperties(fname=fpath)
# 绘图
data_world = data_world.sort_values(by='确诊人数', ascending=False)  # 按确诊人数进行排序
data_world_set = data_world[['确诊人数', '治愈人数', '死亡人数']]
data_world_set.index = data_world['国家名称']
data_world_set.head(10).plot(kind='bar', figsize=(15, 10))  # 对排序前十的国家数据进行绘图
plt.xlabel('国家名称', fontproperties=myfont)
plt.xticks(fontproperties=myfont)
plt.legend(fontsize=30, prop=myfont)  # 设置图例
```

Out[8]: <matplotlib.legend.Legend at 0x173d1fe3df0>


1e7
8
7
6
5
4
3
2
1
0
确诊人数
治愈人数
死亡人数
美国
印度
巴西
法国
德国
英国
俄罗斯
韩国
意大利
土耳其
国家名称

In [9]:
```
from pyecharts.charts import Map
from pyecharts import options as opts
from pyecharts.globals import CurrentConfig, NotebookType

CurrentConfig.NOTEBOOK_TYPE = NotebookType.JUPYTER_NOTEBOOK
name_map = {  # 世界各国数据的中英文对比
    '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',
```

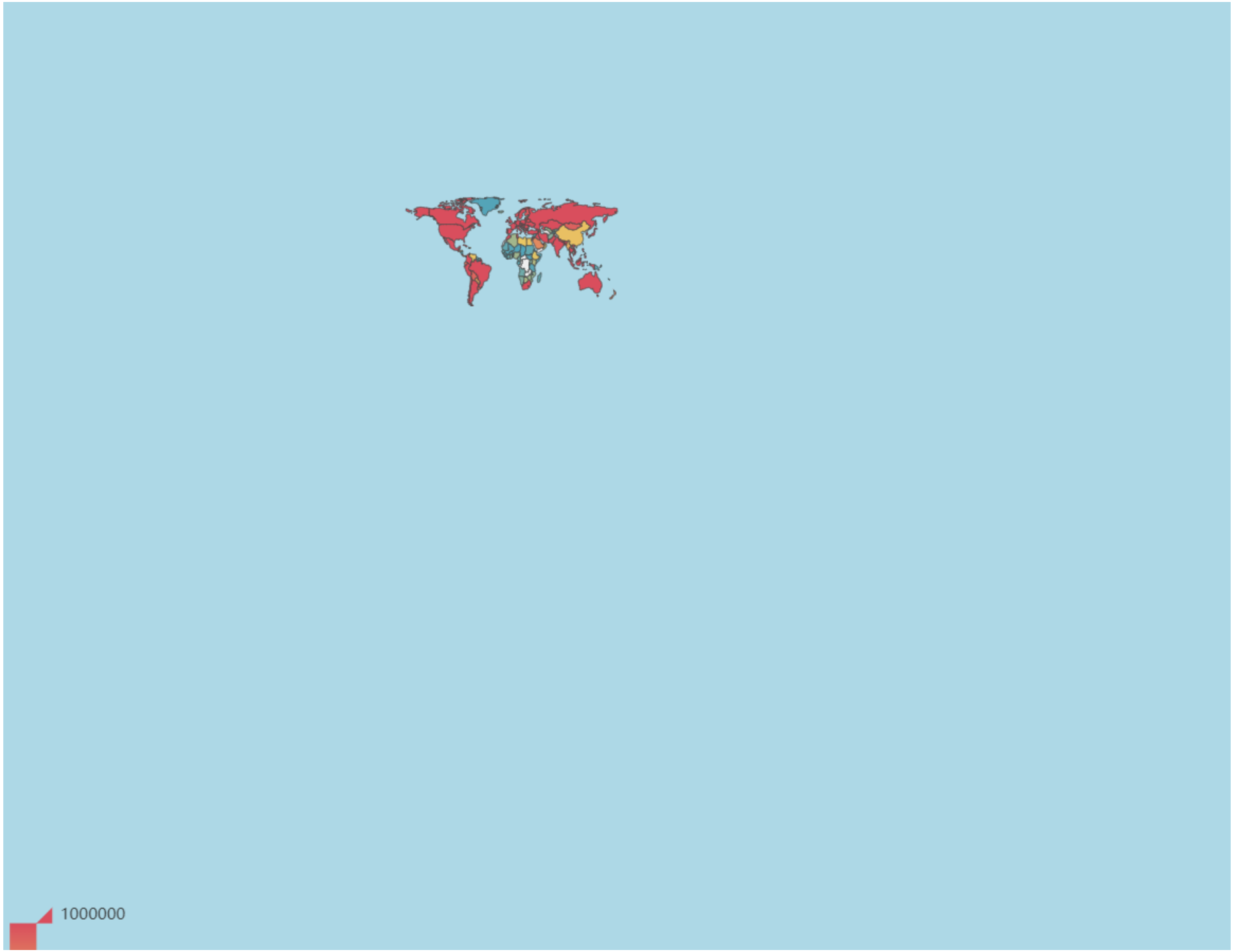
```
    'Central African Rep.': '中非',
    'Canada': '加拿大',
    'Switzerland': '瑞士',
    'Chile': '智利',
    'China': '中国',
    'Ivory Coast': '象牙海岸',
    'Cameroon': '喀麦隆',
    'Dem. Rep. Congo': '刚果民主共和国',
    'Congo': '刚果',
    'Colombia': '哥伦比亚',
    'Costa Rica': '哥斯达黎加',
    'Cuba': '古巴',
    'N. 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'
}

map = Map(init_opts=opts.InitOpts(width="1900px", height="900px",
                                  bg_color="#ADD8E6", page_title="全球疫情确诊人数"))  # 获得世界地图数据
map.add("确诊人数", [list(z) for z in zip(data_world['国家名称'], data_world['确诊人数'])],
        is_map_symbol_show=False,  # 添加确诊人数信息
        # 通过name_map来转化国家的中英文名称方便显示
        maptype="world", label_opts=opts.LabelOpts(is_show=False), name_map=name_map,
        itemstyle_opts=opts.ItemStyleOpts(color="rgb(49,60,72)"),
        ).set_global_opts(
    visualmap_opts=opts.VisualMapOpts(max_=1000000),  # 对视觉映射进行配置
)
map.render_notebook()  # 在notebook中显示
```

Out[9]:


1000000

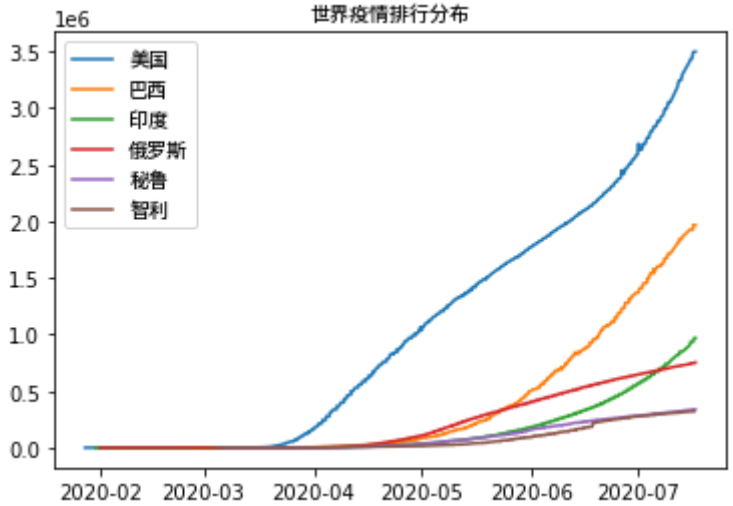
In [10]:
```
country = data_area_times.sort_values('province_confirmedCount', ascending=False).drop_duplicates(
    subset='countryName', keep='first').head(6)['countryName']
country = list(country)  # 对于同一天采集的多个数据，只保留第一次出现的数据也就是最后一次更新的数据
country
```

Out[10]: ['美国', '巴西', '印度', '俄罗斯', '秘鲁', '智利']

In [11]:
```
data_America = data_area_times[data_area_times['countryName'] == '美国']
data_Brazil = data_area_times[data_area_times['countryName'] == '巴西']
data_India = data_area_times[data_area_times['countryName'] == '印度']
data_Russia = data_area_times[data_area_times['countryName'] == '俄罗斯']
data_Peru = data_area_times[data_area_times['countryName'] == '秘鲁']
data_Chile = data_area_times[data_area_times['countryName'] == '智利']

timeindex = data_area_times.index
timeindex = timeindex.floor('D')  # 对于日期索引，只保留具体到哪一天
data_area_times.index = timeindex

timeseries = pd.DataFrame(data_America.index)
timeseries.index = data_America.index
data_America = pd.concat([timeseries, data_America], axis=1)
data_America.drop_duplicates(
    subset='updateTime', keep='first', inplace=True)  # 对美国数据进行处理，获得美国确诊人数的时间序列
data_America.drop('updateTime', axis=1, inplace=True)

timeseries = pd.DataFrame(data_Brazil.index)
timeseries.index = data_Brazil.index
data_Brazil = pd.concat([timeseries, data_Brazil], axis=1)
# 对巴西数据进行处理，获得巴西确诊人数的时间序列
data_Brazil.drop_duplicates(subset='updateTime', keep='first', inplace=True)
data_Brazil.drop('updateTime', axis=1, inplace=True)

timeseries = pd.DataFrame(data_India.index)
timeseries.index = data_India.index
data_India = pd.concat([timeseries, data_India], axis=1)
# 对印度数据进行处理，获得印度确诊人数的时间序列
data_India.drop_duplicates(subset='updateTime', keep='first', inplace=True)
data_India.drop('updateTime', axis=1, inplace=True)

timeseries = pd.DataFrame(data_Russia.index)
timeseries.index = data_Russia.index
data_Russia = pd.concat([timeseries, data_Russia], axis=1)
# 对俄罗斯数据进行处理，获得俄罗斯确诊人数的时间序列
data_Russia.drop_duplicates(subset='updateTime', keep='first', inplace=True)
data_Russia.drop('updateTime', axis=1, inplace=True)

timeseries = pd.DataFrame(data_Peru.index)
timeseries.index = data_Peru.index
```

```
data_Peru = pd.concat([timeseries, data_Peru], axis=1)
# 对秘鲁数据进行处理，获得秘鲁确诊人数的时间序列
data_Peru.drop_duplicates(subset='updateTime', keep='first', inplace=True)
data_Peru.drop('updateTime', axis=1, inplace=True)

timeseries = pd.DataFrame(data_Chile.index)
timeseries.index = data_Chile.index
data_Chile = pd.concat([timeseries, data_Chile], axis=1)
# 对智利数据进行处理，获得智利确诊人数的时间序列
data_Chile.drop_duplicates(subset='updateTime', keep='first', inplace=True)
data_Chile.drop('updateTime', axis=1, inplace=True)

plt.title("世界疫情排行分布", fontproperties=myfont)
plt.plot(data_America['province_confirmedCount'])
plt.plot(data_Brazil['province_confirmedCount'])
plt.plot(data_India['province_confirmedCount'])
plt.plot(data_Russia['province_confirmedCount'])
plt.plot(data_Peru['province_confirmedCount'])
plt.plot(data_Chile['province_confirmedCount'])
plt.legend(country, prop=myfont)
```

Out[11]: <matplotlib.legend.Legend at 0x173d7989e80>



In [12]:
```
import jieba
import re
from wordcloud import WordCloud

def word_cut(x): return jieba.lcut(x)  # 进行结巴分词

news = []
reg = "[^\u4e00-\u9fa5]"
for i in data_news['title']:
    if re.sub(reg, '', i) != '':  # 去掉英文数字和标点等无关字符，仅保留中文词组
        news.append(re.sub(reg, '', i))  # 用news列表汇总处理后的新闻标题

words = []
counts = {}
for i in news:
    words.append(word_cut(i))  # 对所有新闻进行分词
for word in words:
    for a_word in word:
        if len(a_word) == 1:
            continue
        else:
            counts[a_word] = counts.get(a_word, 0)+1  # 用字典存储对应分词的词频
words_sort = list(counts.items())
words_sort.sort(key=lambda x: x[1], reverse=True)

newcloud = WordCloud(font_path=r"D:\专业课\数据分析技术\NotoSansCJK.otf",
                     background_color="white", width=600, height=300, max_words=50)  # 生成词云
newcloud.generate_from_frequencies(counts)
image = newcloud.to_image()  # 转换成图片
image
```

```
Building prefix dict from the default dictionary ...
Loading model from cache C:\Users\BAIHAI~1\AppData\Local\Temp\jieba.cache
Loading model cost 0.619 seconds.
Prefix dict has been built successfully.
```

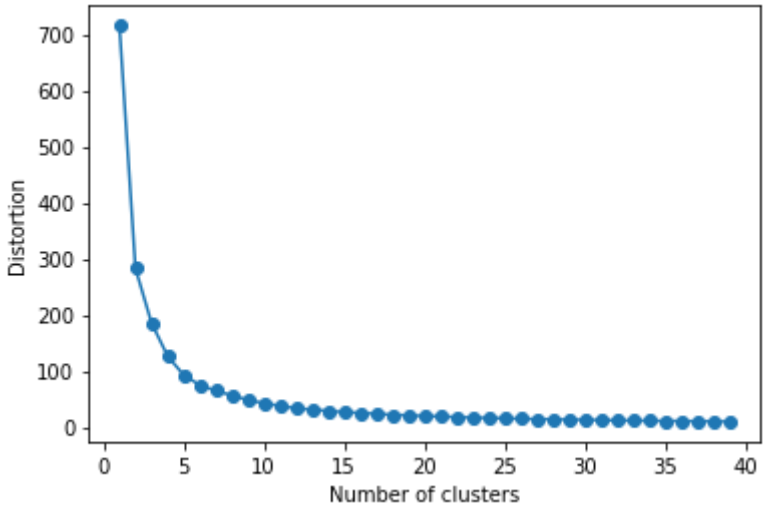
Out[12]:


检测
报告
感染
疫情
出院
新冠
武汉
超过
北京
通报
天津
中国
首例
医院
最新
冠状病毒
输入
世卫
抗疫
英国
肺炎
全球
例新冠
防控
韩国
全国
病毒
境外
组织
日本
确诊
新型
患者
感染者
达例
病例
治愈
新增
累计
出现
美国
增至
湖北
万例
意大利
死亡
口罩
情况

In [19]:
```
from gensim.models import Word2Vec
from sklearn.cluster import KMeans
import warnings
warnings.filterwarnings('ignore')

words = []

for i in news:
    words.append(word_cut(i))
model = Word2Vec(words, sg=0, size=300, window=5, min_count=5)  # 词向量进行训练
keys = model.wv.vocab.keys()  # 获取词汇列表
wordvector = []
for key in keys:
    wordvector.append(model[key])  # 对词汇列表里的所有的词向量进行整合

distortions = []
for i in range(1, 40):
    word_kmeans = KMeans(n_clusters=i,
                         init='k-means++',
                         n_init=10,
                         max_iter=300,
                         random_state=0)  # 分别聚成1-40类
    word_kmeans.fit(wordvector)
    distortions.append(word_kmeans.inertia_)  # 算出样本距离最近的聚类中心的距离总和

plt.plot(range(1, 40), distortions, marker='o')  # 绘图
plt.xlabel('Number of clusters')
plt.ylabel('Distortion')
```

```
Slow version of gensim.models.doc2vec is being used
```

Out[19]:
```
Text(0, 0.5, 'Distortion')
```


Distortion
Number of clusters
700
600
500
400
300
200
100
0
0
5
10
15
20
25
30
35
40

In [20]:
```
word_kmeans = KMeans(n_clusters=10)  # 聚成10类
word_kmeans.fit(wordvector)

labels = word_kmeans.labels_

for num in range(0, 10):
    text = []
    for i in range(len(keys)):
        if labels[i] == num:
            text.append(list(keys)[i])  # 分别获得10类的聚类结果
    print(text)
```

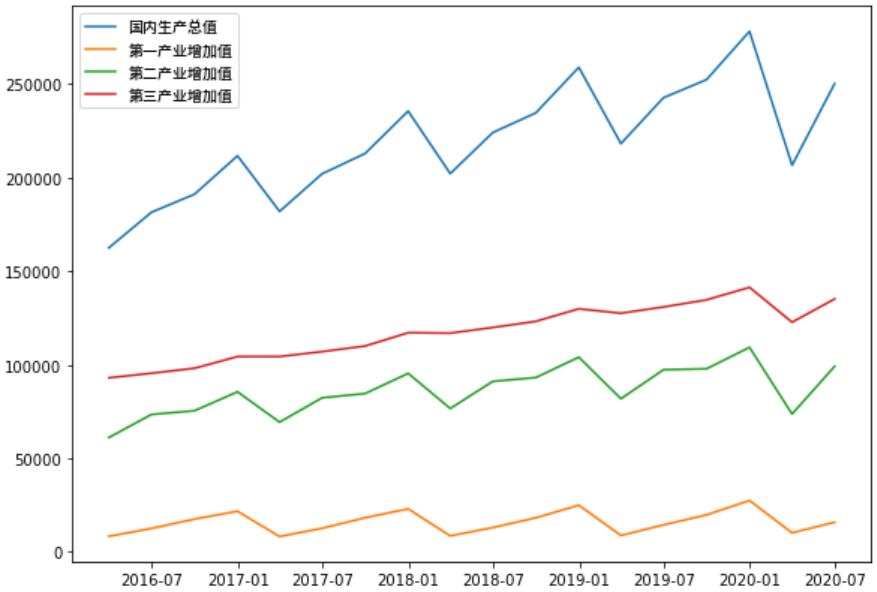
```
['新增', '例新冠', '肺炎', '确诊', '病例', '累计', '新冠', '例', '无', '治愈', '出院', '境外', '输入']
['超', '万例', '日', '西班牙', '报告', '升至', '超过', '达例', '重庆', '印度', '上海', '本地', '日本', '德国', '法国', '意大利',
'天津', '时', '死亡', '单日', '广东', '通报', '无症状', '感染者', '韩国', '疑似病例', '增至', '黑龙江', '山东']
['达', '佩戴', '举行', '悼念', '万人', '没有', '受', '新加坡', '呈', '重启', '及', '卫生部长', '超万', '提供', '委员会', '暴发',
'以来', '均', '于', '复课', '调整', '戴', '发生', '性', '封锁', '提醒', '继续', '其中', '因', '高风险', '推迟', '泰国', '约', '最',
'研究', '呼吁', '今日', '观察', '海外', '直播', '反弹', '特朗普', '全面', '年月日时', '小时', '内', '发地', '北京市', '解除', '管
理', '岁', '以上', '合作', '管控', '非洲', '数据', '来', '大使馆', '公民', '回国', '扩大', '美', '土耳其', '开展', '成', '亿', '允
许', '解封', '社区', '医生', '首批', '延期', '大', '安徽', '社会', '市场', '截至', '清零', '来自', '假期', '关闭', '相关', '应急',
'全部', '有例', '日起', '结束', '加强', '做好', '再次', '出台', '捐赠', '医护人员', '援助', '学校', '时间', '机场', '显示', '奥运
会', '发热', '天无', '支援', '好消息', '实行', '景区', '队员', '逝世', '一级', '五一', '牺牲', '公主']
['逝者', '仪式', '状态', '上调', '警惕', '多国', '下跌', '一个月', '餐厅', '一周', '塞尔维亚', '波兰', '压力', '加大', '超例', '回
升', '阿根廷', '省份', '住院', '葡萄牙', '证明', '坚决', '例均', '工作人员', '警告', '大厅', '返回', '展开', '筛查', '岗位', '工
人', '发出', '主要', '欧盟', '外长', '准备', '任何', '很', '有效', '不是', '一例', '传染病', '重开', '多项', '男子', '酒店', '总干
事', '之下', '接待', '减少', '奥地利', '多家', '两', '行动', '地方', '加剧', '出席', '做', '指南', '经', '转机', '这些', '等级',
'荷兰', '使馆', '如何', '落实', '欧元', '级', '大区', '再度', '史', '所', '给', '中国政府', '援', '举办', '会议', '统计', '匈牙利',
'执行', '秘书', '墨西哥', '保护', '恢复正常', '呼吸机', '提升', '比', '建设', '座', '印尼', '吴尊友', '万个', '考试', '智利', '区
域', '基本', '详情', '解禁', '增幅', '多地', '省', '家中', '禁足', '厄瓜多尔', '使用', '规模', '卫健委日', '交易', '削减', '监狱',
'追加', '世界卫生组织', '办事处', '主任', '超人', '化', '其他', '迪士尼', '视频', '波', '事态', '包括', '名新冠', '副', '过去', '罚
款', '哥伦比亚', '针对', '主流', '团结', '全员', '型', '辽宁大连', '环境', '全体', '中考', '默哀', '市民', '看', '其', '移动', '纽
约市', '公务员', '防止', '进出', '处以', '最小', '出租车', '运营', '启用', '发', '生命', '保持', '三级', '奥组委', '考虑', '认为',
'万多', '流感', '采取', '此前', '诊断', '有关', '投资', '沙特', '联合', '旅游业', '变化', '赞比亚', '接收', '系统', '幼儿园', '启
程', '现', '发言人', '总领馆', '共计', '禁令', '强调', '分批', '孟加拉国', '系', '一年', '亿只', '临床试验', '境内', '治愈率',
'籍', '央视', '不明', '民航', '运输', '半数', '封城', '举措', '得到', '院士', '复阳', '人民', '破万', '紧张', '大会', '州长', '病毒
感染', '首相', '封闭式', '河北省', '航空公司', '省市', '合肥', '防护', '接近', '财政', '开', '鲍里斯', '销售', '普京', '轻症', '援
鄂', '首尔', '夜店', '白宫', '缅甸', '万次', '同时', '提高', '高三', '行业', '人士', '沪', '严禁', '城市', '供应链', '陆续', '复
学', '明确', '工资', '办理', '而', '英雄', '免疫', '入院', '同一', '指导', '危重', '收到', '告急', '回', '网友', '收治', '临床',
'返程', '严防', '不足', '暂', '哈尔滨', '让', '首日', '实现', '宣言', '急需', '派', '辽宁省', '撤离', '全力', '现在', '第二批', '野
生动物', '内蒙古自治区', '第一批', '雷', '神山', '江西省', '至例', '记者', '滞留', '福建省', '贵州省', '黎巴嫩', '山西省', '连降',
'安徽省', '日前', '湖南省', '台湾', '迎接', '黄石', '火神', '山', '婴儿', '急']
```

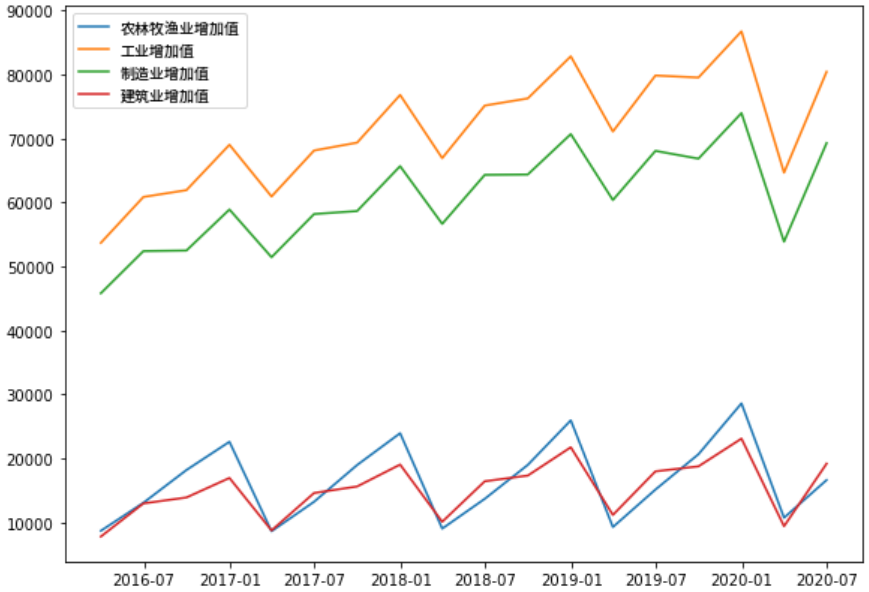
['摩洛哥', '封闭', '公共', '形势', '受新冠', '低', '冲击', '航空', '主席', '发展', '部长', '秘鲁', '客运', '酒吧', '英国首相', '未来', '调查', '哈萨克斯坦', '等国', '一名', '恶化', '但', '监护', '扩散', '引', '市', '公司', '迪拜', '以外', '申请', '严格', '省区市', '裁员', '中心', '养老院', '员工', '出行', '比利时', '今年', '万名', '毕业生', '失业', '宵禁', '抗议', '方式', '羟', '氯喹', '推动', '留学生', '无法', '尚未', '规定', '至时', '同比', '民航局', '官方', '放松', '若', '体温', '正常', '范围', '重点', '创新', '高', '两周', '社交', '距离', '自', '莫斯科', '升级', '购买', '线上', '复苏', '州', '第二阶段', '期', '外籍', '隐瞒', '密接', '症状', '两天', '分享', '排除', '月底', '边境', '每天', '多名', '市长', '破', '食品', '今起', '下调', '生活', '采购', '救治', '失业率', '项目', '价格', '说', '首个', '集体', '吨', '万份', '采样', '快递', '家', '团队', '实验室', '万人次', '研发', '积极', '流行', '澳门', '资金', '一季度', '批准', '下', '河北', '保障', '因新冠', '总', '经验', '总数', '甘肃', '军队', '医学观察', '下降', '更', '联合国', '发布会', '医用', '佛罗里达州', '刚果', '即将', '床位', '中方', '逼近', '张文宏', '关联', '处于', '海鲜', '样本', '补助', '吗', '任务', '安排', '大部分', '业务', '停止', '排查', '二级', '需要', '前往', '用', '力度', '关注', '缓解', '不得', '世界', '逾', '确定', '战疫', '生产', '瑞士', '病人', '快速', '都', '有序', '个人', '吉林市', '返京', '回家', '占', '领导', '药物', '试剂', '零', '机制', '召开', '群体', '去世', '今天', '撤侨', '趋缓', '当地', '救助', '行程', '政策', '降', '可以', '黑龙江省', '外出', '也', '阶段', '包机', '重要', '爱心', '俄', '测试', '航线', '以下', '至人', '代表', '网络', '轨迹', '乘', '吉林省', '鄂', '加快', '只', '避免', '表明', '试剂盒', '共同', '降至例', '绥芬河', '证据', '乘客', '表示', '外', '重庆市', '小汤山', '山西', '大臣', '诊疗', '金银', '潭', '天津市', '贵州', '湖南', '全区', '云南省', '志哀', '广东省', '河南省', '黄冈', '捐款', '两例']
['口罩', '将', '恢复', '与', '的', '在', '疫情', '防控', '病毒', '防疫', '措施', '中国', '抗疫', '和', '已', '隔离', '检测', '人员', '医院', '武汉', '不', '医疗']
['阿塞拜疆', '科威特', '塞内加尔', '白俄罗斯', '越南', '国际航班', '引发', '好', '营业', '乌克兰', '保加利亚', '多州', '乌兹别克斯坦', '严峻', '疾病', '希腊', '两个', '冠', '北美', '危机', '新西兰', '洛杉矶', '加纳', '阿曼', '关键', '心理', '须', '帮助', '肯尼亚', '死于', '倍', '老人', '圭亚那', '巴基斯坦', '案例', '具备', '吉尔吉斯斯坦', '挑战', '数超', '序列', '捷克', '名单', '近万', '外卖', '预测', '卫健委月', '上海市', '正在', '比赛', '进京', '量', '多数', '纳入', '明显', '海滩', '圈', '经济衰退', '疾控', '尚', '条', '建', '教育部', '蛋白质', '布', '全', '三个', '份', '沈阳', '水平', '变', '斯里兰卡', '停运', '也门', '贫民窟', '停课', '通告', '参加', '就诊', '作用', '供应', '不断', '突尼斯', '以色列', '参与', '复航', '流动', '牡丹江', '赤道几内亚', '金', '四川省', '近例', '乌拉圭', '尼日利亚', '次', '啦', '马里', '苏丹', '亚洲', '阿尔及利亚', '胜利', '近万人', '武汉协和医院', '叙利亚', '卡塔尔', '伊拉克', '各', '错峰', '发改委', '毛里求斯', '柳叶刀', '数量', '检疫', '埃塞俄比亚', '舒兰市', '西藏自治区', '日本政府', '江苏省', '情况通报', '补贴', '津巴布韦', '巴林', '亚美尼亚', '过万', '两万', '传染', '医务', '堂食', '构成', '上班', '浙江省', '文莱', '南京', '喀麦隆', '出征', '吉布提', '斯洛伐克', '凯旋', '卢森堡', '病情', '订正', '利比亚', '布基纳法索', '资助', '例例', '蒙古国', '青海省', '千例', '增例', '立陶宛', '安道尔', '坦桑尼亚', '阿尔巴尼亚', '格鲁吉亚', '老挝', '塞浦路斯', '有名', '襄阳', '深圳', '春节假期']
['美国', '起', '民众', '进入', '月', '国家', '驻', '患者', '人数', '北京', '向', '个', '风险', '影响', '至', '开放', '又', '一', '再', '延长', '出现', '后', '对', '应对', '世卫', '组织', '全球', '物资', '经济', '为', '感染', '核酸', '发布', '人', '名', '宣布', '万', '所有', '或', '限制', '有', '多', '了', '工作', '被', '可', '等', '新', '医疗队', '专家', '期间', '暂停', '复工', '国际', '公布', '响应', '湖北']
['加速', '增长', '下周', '必须', '外交部', '逐步', '香港', '到', '新疆', '最高', '月份', '曾', '建议', '澳大利亚', '机构', '结果', '感谢', '伊朗', '俄罗斯', '菲律宾', '决定', '昨日', '加州', '治疗', '正', '就', '问题', '至少', '阿联酋', '居民', '旅行', '安全', '控制', '严重', '例为', '本土', '需', '小区', '强制', '高考', '高校', '面临', '大规模', '进一步', '我', '现有', '最后', '蔓延', '达到', '每日', '场所', '辽宁', '旅游', '游客', '大幅', '卫生部', '加拿大', '学生', '数', '官员', '疾控中心', '放宽', '重新', '推出', '家庭', '各国', '考生', '注意', '江西', '是否', '人群', '地', '一个', '内蒙古', '康复', '回应', '就业', '会', '南非', '并', '能力', '发放', '阴性', '已有', '导致', '已经', '通过', '关于', '信息', '纽约', '禁止', '运抵', '助力', '方舱', '还', '常态', '集中', '福建', '蔬菜', '消费', '医疗机构', '共', '接触', '紧急状态', '预计', '成为', '第二', '密切接触', '抗击', '不会', '由', '江苏', '埃及', '一天', '马来西亚', '公共卫生', '门诊', '确认', '服务', '第例', '媒体', '非', '正式', '一线', '医务人员', '上', '师生', '河南', '事件', '海南', '四川', '中小学', '一律', '级别', '武汉市', '云南', '未', '陕西', '我国', '赴', '突破', '以', '复产', '接受',

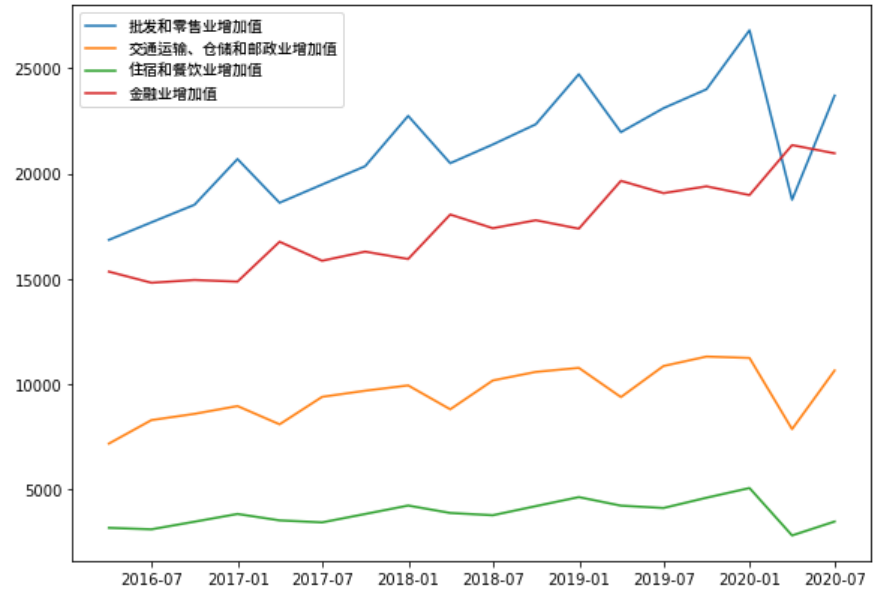
'同胞', '湖北省', '钟南山', '疑似', '中央', '临时', '国务院', '联防', '联控', '通知', '各地', '宁夏', '年级', '广州', '抵达', '约翰逊', '高峰', '广西', '铁路', '口岸', '儿童', '吉林', '纽约州', '部门', '工作者', '预约', '护士', '共有', '免费', '亿元', '邮轮', '痊愈', '除', '突发', '医护', '浙江', '驰援', '全省', '山东省', '含', '兵团', '下半旗', '烈士', '深切', '钻石', '西藏', '青海']
['巴西', '从', '部分', '地区', '持续', '中', '英国', '东京', '年', '发现', '总统', '病毒检测', '阳性', '国内', '航班', '全国', '首都', '进行', '首次', '情况', '可能', '连续', '天', '活动', '上升', '卫健委', '仍', '重症', '聚集', '实施', '国', '支持', '号', '开始', '传播', '入境', '取消', '称', '政府', '是', '丨', '旅客', '要求', '新型', '冠状病毒', '启动', '计划', '应', '要', '近', '增加', '最新', '日时', '返校', '令', '总理', '新闻', '前', '目前', '健康', '最大', '完成', '疫苗', '欧洲', '首例', '紧急', '者', '企业', '居家', '卫生', '专家组', '抗体', '开学', '年月日']

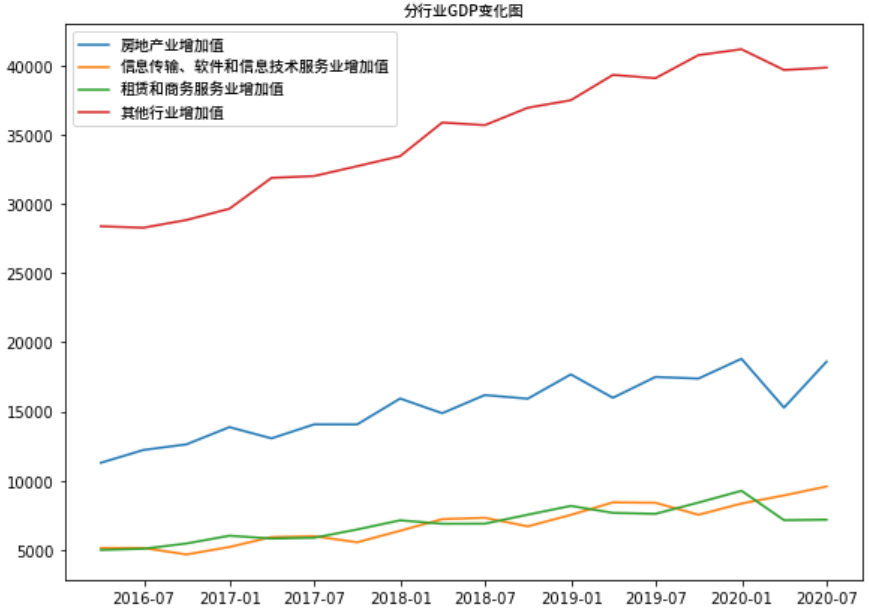
In [21]:
```
sum_GDP = ['国内生产总值', '第一产业增加值', '第二产业增加值', '第三产业增加值']
industry_GDP = ['农林牧渔业增加值', '工业增加值', '制造业增加值', '建筑业增加值']
industry2_GDP = ['批发和零售业增加值', '交通运输、仓储和邮政业增加值', '住宿和餐饮业增加值', '金融业增加值']
industry3_GDP = ['房地产业增加值', '信息传输、软件和信息技术服务业增加值',
                 '租赁和商务服务业增加值', '其他行业增加值']  # 对不同行业分四类来展现

fig = plt.figure()
fig, axes = plt.subplots(2, 2, figsize=(21, 15))  # 分别用四个子图来展现数据变化情况

axes[0][0].plot(data_economy[sum_GDP])
axes[0][0].legend(sum_GDP, prop=myfont)
axes[0][1].plot(data_economy[industry_GDP])
axes[0][1].legend(industry_GDP, prop=myfont)
axes[1][0].plot(data_economy[industry2_GDP])
axes[1][0].legend(industry2_GDP, prop=myfont)
axes[1][1].plot(data_economy[industry3_GDP])
axes[1][1].legend(industry3_GDP, prop=myfont)

plt.title('分行业GDP变化图', fontproperties=myfont)
```

Out[21]:
```
Text(0.5, 1.0, '分行业GDP变化图')

<Figure size 432x288 with 0 Axes>
```

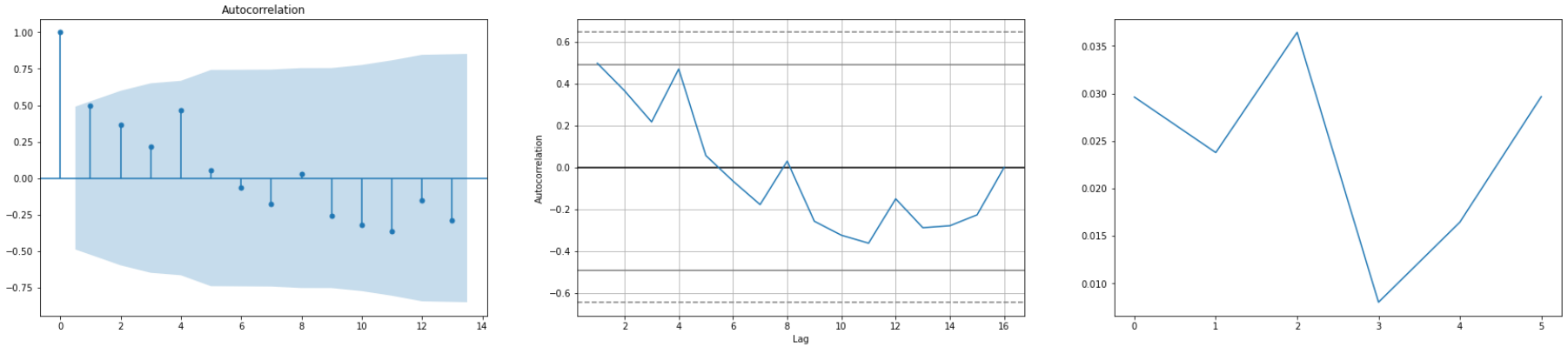

国内生产总值
第一产业增加值
第二产业增加值
第三产业增加值
250000
200000
150000
100000
50000
0
2016-07
2017-01
2017-07
2018-01
2018-07
2019-01
2019-07
2020-01
2020-07


农林牧渔业增加值
工业增加值
制造业增加值
建筑业增加值
90000
80000
70000
60000
50000
40000
30000
20000
10000
2016-07
2017-01
2017-07
2018-01
2018-07
2019-01
2019-07
2020-01
2020-07


批发和零售业增加值
交通运输、仓储和邮政业增加值
住宿和餐饮业增加值
金融业增加值
25000
20000
15000
10000
5000
2016-07
2017-01
2017-07
2018-01
2018-07
2019-01
2019-07
2020-01
2020-07


分行业GDP变化图
房地产业增加值
信息传输、软件和信息技术服务业增加值
租赁和商务服务业增加值
其他行业增加值
40000
35000
30000
25000
20000
15000
10000
5000
2016-07
2017-01
2017-07
2018-01
2018-07
2019-01
2019-07
2020-01
2020-07

In [22]:
```
from statsmodels.graphics.tsaplots import plot_acf
from pandas.plotting import autocorrelation_plot
from statsmodels.sandbox.stats.diagnostic import acorr_ljungbox


GDP_type = ['国内生产总值', '第一产业增加值', '第二产业增加值', '第三产业增加值',
            '农林牧渔业增加值', '工业增加值', '制造业增加值', '建筑业增加值', '批发和零售业增加值',
            '交通运输、仓储和邮政业增加值', '住宿和餐饮业增加值', '金融业增加值',
            '房地产业增加值', '信息传输、软件和信息技术服务业增加值', '租赁和商务服务业增加值', '其他行业增加值']

for i in GDP_type:
    each_data = data_economy[i][:-2]
    plt.figure(figsize=(30, 6))
    ax1 = plt.subplot(1, 3, 1)
    ax2 = plt.subplot(1, 3, 2)
    ax3 = plt.subplot(1, 3, 3)
    LB2, P2 = acorr_ljungbox(each_data)  # 进行纯随机性检验
    plot_acf(each_data, ax=ax1)
    autocorrelation_plot(each_data, ax=ax2)  # 进行平稳性检验
    ax3.plot(P2)
```

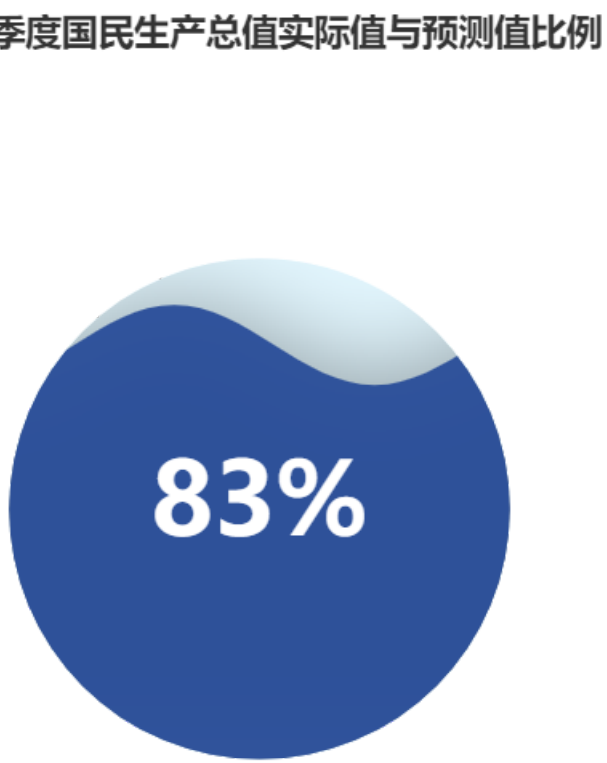


In [23]:
```
from statsmodels.tsa.arima_model import ARMA
from statsmodels.tsa.stattools import arma_order_select_ic

warnings.filterwarnings('ignore')
data_arma = pd.DataFrame(data_economy['国内生产总值'][:-2])  # 选取疫情期前的16个季度进行建模
a, b = arma_order_select_ic(data_arma, ic='hqic')['hqic_min_order']
arma = ARMA(data_arma, order=(a, b)).fit()  # 使用ARMA建模
rate1 = list(data_economy['国内生产总值'][-2] /
             arma.forecast(steps=1)[0])  # 获得疫情期当季度的预测值
rate1  # 实际值与预测值的比率
```

Out[23]: [0.82736646404142]

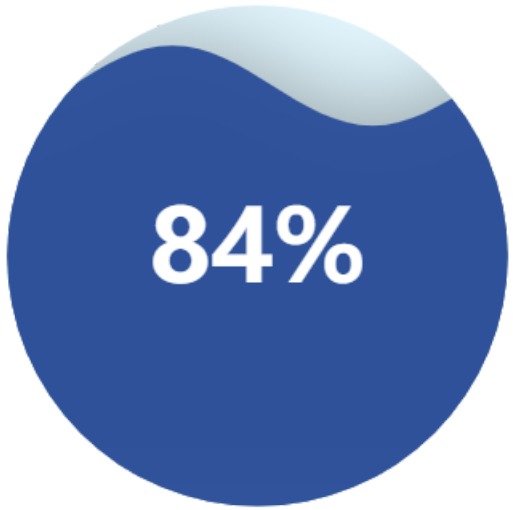
In [24]:
```
from pyecharts import options as opts
from pyecharts.charts import Liquid

c = (
    Liquid()
    .add("实际值/预测值", rate1, is_outline_show=False)
    .set_global_opts(title_opts=opts.TitleOpts(title="第一季度国民生产总值实际值与预测值比例",
                                               pos_left="center"))
)
c.render_notebook()
```

Out[24]:



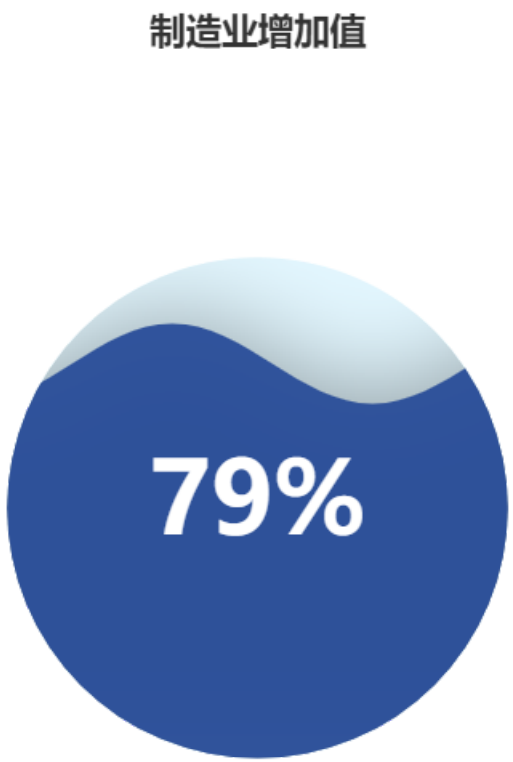
In [25]:
```
warnings.filterwarnings('ignore')
data_arma = pd.DataFrame(data_economy['工业增加值'][:-2])
a, b = arma_order_select_ic(data_arma, ic='hqic')['hqic_min_order']
arma = ARMA(data_arma, order=(a, b)).fit()
rate2 = list(data_economy['工业增加值'][-2]/arma.forecast(steps=1)[0])
c = (
    Liquid()
    .add("实际值/预测值", rate2, is_outline_show=False)
    .set_global_opts(title_opts=opts.TitleOpts(title="工业增加值比例", pos_left="center"))
)
c.render_notebook()
```

Out[25]:



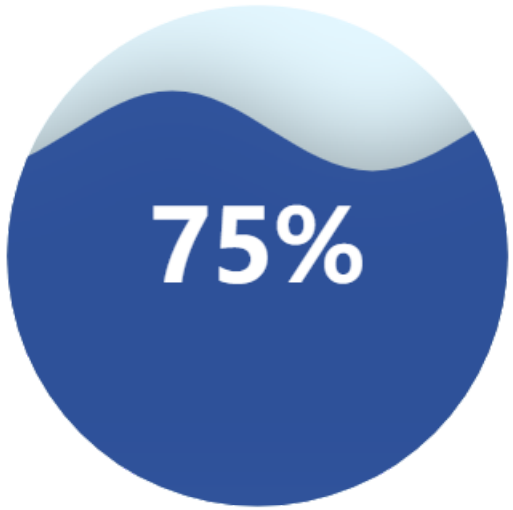
In [26]:
```
warnings.filterwarnings('ignore')
data_arma = pd.DataFrame(data_economy['制造业增加值'][:-2])
a, b = arma_order_select_ic(data_arma, ic='hqic')['hqic_min_order']
arma = ARMA(data_arma, order=(a, b)).fit()
rate3 = list(data_economy['制造业增加值'][-2]/arma.forecast(steps=1)[0])
c = (
    Liquid()
    .add("实际值/预测值", rate3, is_outline_show=False)
    .set_global_opts(title_opts=opts.TitleOpts(title="制造业增加值", pos_left="center"))
)
c.render_notebook()
```

Out[26]:



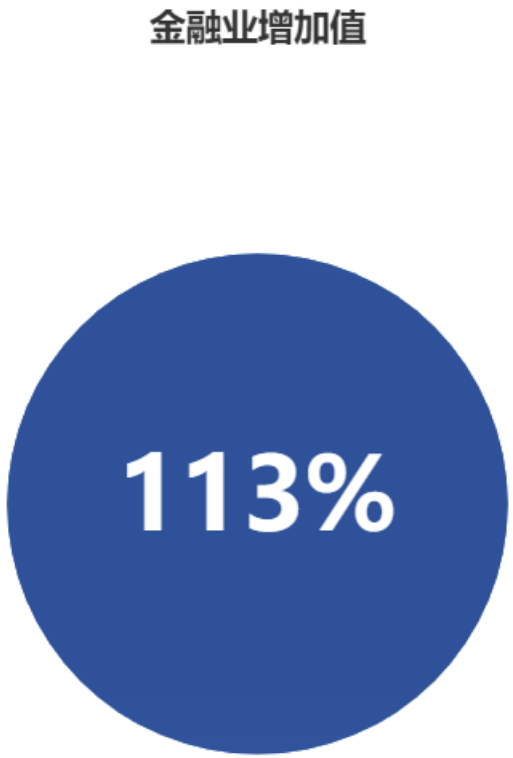
In [27]:
```
data_arma = pd.DataFrame(data_economy['批发和零售业增加值'][:-2])
a, b = arma_order_select_ic(data_arma, ic='hqic')['hqic_min_order']
arma = ARMA(data_arma, order=(a, b)).fit()
rate4 = list(data_economy['批发和零售业增加值'][-2]/arma.forecast(steps=1)[0])
c = (
    Liquid()
    .add("实际值/预测值", rate4, is_outline_show=False)
    .set_global_opts(title_opts=opts.TitleOpts(title="批发和零售业增加值", pos_left="center"))
)
c.render_notebook()
```

Out[27]:



In [28]:
```
data_arma = pd.DataFrame(data_economy['金融业增加值'][:-2])
a, b = arma_order_select_ic(data_arma, ic='hqic')['hqic_min_order']
arma = ARMA(data_arma, order=(a, b)).fit()
rate = list(data_economy['金融业增加值'][-2]/arma.forecast(steps=1)[0])
c = (
    Liquid()
    .add("实际值/预测值", rate, is_outline_show=False)
    .set_global_opts(title_opts=opts.TitleOpts(title="金融业增加值", pos_left="center"))
)
c.render_notebook()
```

Out[28]:



In [29]:
```
data_arma = pd.DataFrame(data_economy['信息传输、软件和信息技术服务业增加值'][:-2])
a, b = arma_order_select_ic(data_arma, ic='hqic')['hqic_min_order']
arma = ARMA(data_arma, order=(a, b)).fit()
rate = list(data_economy['信息传输、软件和信息技术服务业增加值'][-2]/arma.forecast(steps=1)[0])
c = (
    Liquid()
    .add("实际值/预测值", rate, is_outline_show=False)
    .set_global_opts(title_opts=opts.TitleOpts(title="信息传输、软件和信息技术服务业增加值",
                                               pos_left="center"))
)
c.render_notebook()
```

Out[29]:

In [ ]: