June 1, 2022

```
[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
     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:  #              dataframe
         country.append(i['provinceName'])
         confirmed.append(i['confirmedCount'])
         lived.append(i['curedCount'])
         dead.append(i['deadCount'])

     data_world = pd.DataFrame()
     data_world['  '] = country
     data_world['  '] = confirmed
     data_world['  '] = lived
     data_world['  '] = dead
     data_world.head(5)
```

```
[1]:
     0      29583616   368023  149044
     1      26244107  4328400  138864
     2      18086462   336548   24167
```

```
3       22455392  6491069  178880
4      12326264   150376  106341
```

[2]:
```
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
```

[2]:
```
                                                               \
2016-03-31  162410.0   8312.7   61106.8   92990.5    8665.5  53666.4  45784.0
2016-06-30  181408.2  12555.9   73416.5   95435.8   13045.5  60839.2  52378.3
2016-09-30  191010.6  17542.4   75400.5   98067.8   18162.2  61902.5  52468.3
2016-12-31  211566.2  21728.2   85504.1  104334.0   22577.8  68998.4  58878.4
2017-03-31  181867.7   8205.9   69315.5  104346.3    8595.8  60909.3  51419.7
2017-06-30  201950.3  12644.9   82323.0  106982.4   13204.2  68099.8  58172.1
2017-09-30  212789.3  18255.8   84574.1  109959.5   18944.2  69327.2  58632.6
2017-12-31  235428.7  22992.9   95368.0  117067.8   23915.8  76782.9  65652.1
2018-03-31  202035.7   8575.7   76598.2  116861.8    9005.8  66905.6  56631.9
2018-06-30  223962.2  13003.8   91100.6  119857.8   13662.2  75122.1  64294.9
2018-09-30  234474.3  18226.9   93112.5  123134.9   18961.8  76239.6  64348.2
2018-12-31  258808.9  24938.7  104023.9  129846.2   25929.0  82822.1  70662.1
2019-03-31  218062.8   8769.4   81806.5  127486.9    9249.4  71064.5  60357.1
2019-06-30  242573.8  14437.6   97315.6  130820.6   15108.7  79820.7  68041.8
2019-09-30  252208.7  19798.0   97790.4  134620.4   20629.0  79501.8  66823.8
2019-12-31  278019.7  27461.6  109252.8  141305.2   28579.9  86721.6  73952.4
2020-03-31  206504.3  10186.2   73638.0  122680.1   10708.4  64642.0  53852.0
2020-06-30  250110.1  15866.8   99120.9  135122.3   16596.4  80402.4  69258.8

                                                          \
2016-03-31   7763.0    16847.5          7180.5     3181.6  15340.4  11283.0
2016-06-30  12943.8    17679.8          8295.0     3112.3  14811.7  12209.7
2016-09-30  13870.6    18513.0          8591.6     3473.2  14945.4  12615.3
2016-12-31  16921.5    20684.1          8961.6     3840.7  14866.4  13861.4
2017-03-31   8725.3    18608.9          8094.5     3536.5  16758.8  13047.0
2017-06-30  14574.4    19473.6          9397.7     3440.9  15856.3  14059.0
2017-09-30  15590.1    20342.9          9688.7     3838.5  16290.4  14054.9
2017-12-31  19015.8    22731.1          9940.9     4240.1  15938.8  15925.1
2018-03-31  10073.8    20485.5          8806.5     3887.8  18050.6  14863.5
2018-06-30  16404.3    21374.2         10174.9     3779.6  17401.0  16176.1
2018-09-30  17294.5    22334.1         10582.3     4212.6  17780.6  15914.0
2018-12-31  21720.4    24710.0         10773.5     4640.6  17378.1  17669.5
2019-03-31  11143.1    21959.2          9386.6     4234.9  19650.1  15979.2
2019-06-30  17954.2    23097.0         10861.3     4123.0  19064.9  17484.4
2019-09-30  18734.6    23993.6         11310.2     4610.5  19388.3  17369.0
2019-12-31  23072.4    26795.9         11244.0     5071.2  18973.8  18798.9
```

```
2020-03-31   9377.8    18749.6          7865.1     2820.9  21346.8  15268.3
2020-06-30  19156.8    23696.1         10650.0     3481.3  20954.7  18593.6


2016-03-31              5128.8       4985.3  28368.1
2016-06-30              5130.7       5075.1  28265.4
2016-09-30              4662.3       5452.4  28822.1
2016-12-31              5202.3       6015.8  29636.1
2017-03-31              5915.2       5811.9  31864.3
2017-06-30              5977.9       5868.4  31998.1
2017-09-30              5539.8       6464.6  32708.0
2017-12-31              6376.0       7128.4  33433.7
2018-03-31              7212.2       6879.5  35864.9
2018-06-30              7309.6       6885.3  35673.1
2018-09-30              6690.9       7533.3  36930.6
2018-12-31              7520.8       8170.4  37474.6
2019-03-31              8424.8       7665.1  39306.0
2019-06-30              8395.6       7596.7  39067.3
2019-09-30              7528.1       8409.1  40734.5
2019-12-31              8341.3       9262.5  41158.2
2020-03-31              8928.0       7137.9  39659.6
2020-06-30              9573.0       7174.4  39831.4
```

[3]:
```
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')
```

[4]:
```
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'])  #
data_area_times.index = time  #
data_area_times = data_area_times.drop('updateTime', axis=1)
data_area_times.head(5)

data_area_times.isnull().any()  #
```

[4]:
```
countryName                False
province_confirmedCount    False
province_curedCount        False
province_deadCount         False
dtype: bool
```

```
[5]: data_news_times = data_news[['pubDate', 'title', 'summary']]
     time = pd.DatetimeIndex(data_news_times['pubDate'])
     data_news_times.index = time  #
     data_news_times = data_news_times.drop('pubDate', axis=1)
     data_news_times.head(5)
```

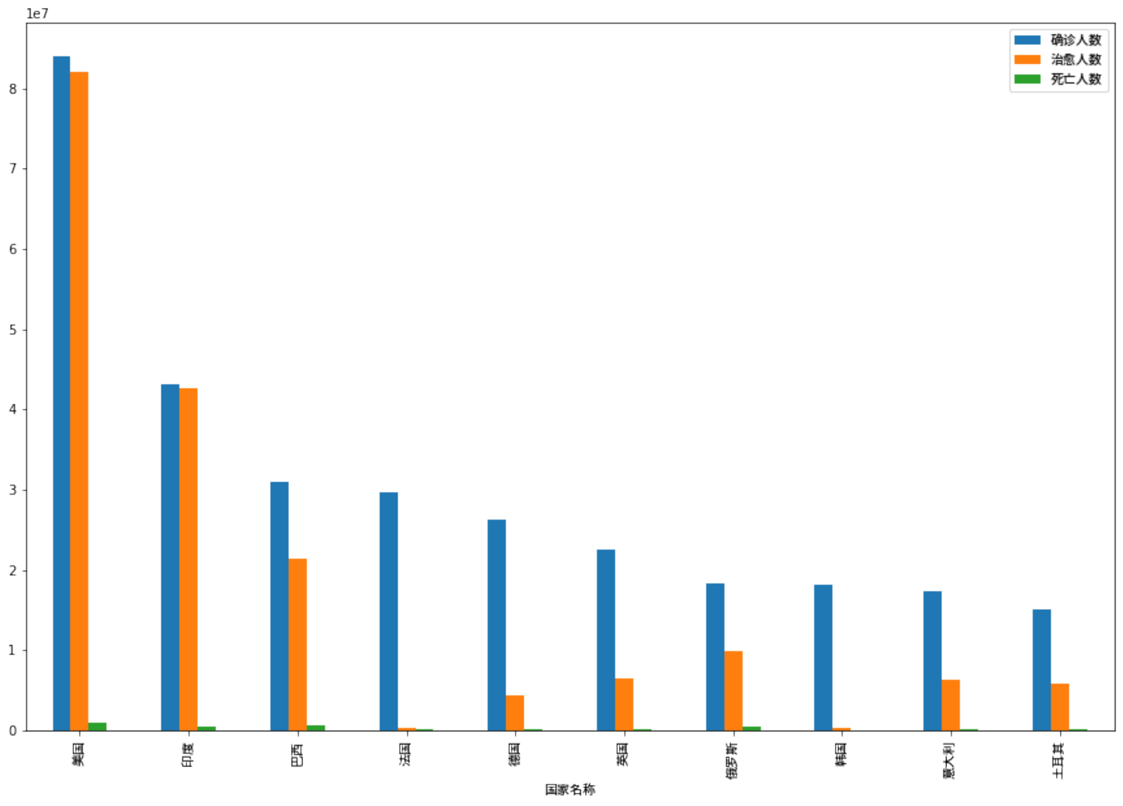
```
[5]:                                                  title  \
     pubDate
     2020-07-17 05:40:08     71434        354
     2020-07-17 06:06:49        201
     2020-07-16 22:31:00      493        26165
     2020-07-16 22:29:48     791        57668
     2020-07-16 21:26:54      777        35003


                                                                     summary
     pubDate
     2020-07-17 05:40:08      ·            7 16 17:33    17 0…
     2020-07-17 06:06:49     7 16 18         45403    2012151 …
     2020-07-16 22:31:00    7 16               24         …
     2020-07-16 22:29:48       16       24    791        …
     2020-07-16 21:26:54      7 16     24  19097        777 …
```

```
[6]: print(data_world.isnull().any())
     print(data_economy.isnull().any())
     print(data_area_times.isnull().any())
     print(data_news_times.isnull().any())  #
```

```
      False
      False
      False
      False
dtype: bool
                         False
                        False
                        False
                        False
                       False
                          False
                         False
                         False
                       False
                   False
                       False
                         False
                        False
                  False
                      False
                  False
dtype: bool
countryName                False
province_confirmedCount    False
province_curedCount        False
province_deadCount         False
dtype: bool
title      False
summary    False
dtype: bool
```

[7]:
```python
import matplotlib.pyplot as plt
import matplotlib
import os

%matplotlib inline
#
fpath = os.path.join("./NotoSansCJK.otf")
myfont = matplotlib.font_manager.FontProperties(fname=fpath)
#
data_world = data_world.sort_values(by='   ', ascending=False)  #
data_world_set = data_world[['   ', '   ', '   ']]
data_world_set.index = data_world['   ']
data_world_set.head(10).plot(kind='bar', figsize=(15, 10))  #
plt.xlabel('   ', fontproperties=myfont)
plt.xticks(fontproperties=myfont)
plt.legend(fontsize=30, prop=myfont)  #
```

[7]:
```
<matplotlib.legend.Legend at 0x1bd59422940>
```



[8]:
```
!pip install pyecharts==1.7.1
```

```
Requirement already satisfied: pyecharts==1.7.1 in e:\anaconda3\lib\site-
packages (1.7.1)
Requirement already satisfied: prettytable in
c:\users\12131\appdata\roaming\python\python39\site-packages (from
pyecharts==1.7.1) (3.3.0)
Requirement already satisfied: jinja2 in
c:\users\12131\appdata\roaming\python\python39\site-packages (from
pyecharts==1.7.1) (3.1.2)
Requirement already satisfied: simplejson in
c:\users\12131\appdata\roaming\python\python39\site-packages (from
pyecharts==1.7.1) (3.17.6)
Requirement already satisfied: MarkupSafe>=2.0 in
c:\users\12131\appdata\roaming\python\python39\site-packages (from
jinja2->pyecharts==1.7.1) (2.1.1)
Requirement already satisfied: wcwidth in
c:\users\12131\appdata\roaming\python\python39\site-packages (from
prettytable->pyecharts==1.7.1) (0.2.5)

WARNING: There was an error checking the latest version of pip.
```

```
[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 = {  #
         'Singapore Rep.': '  ',
         'Dominican Rep.': '  ',
         'Palestine': '  ',
         'Bahamas': '  ',
         'Timor-Leste': '  ',
         'Afghanistan': '  ',
         'Guinea-Bissau': '   ',
         "Côte d'Ivoire": '  ',
         'Siachen Glacier': '   ',
         "Br. Indian Ocean Ter.": '    ',
         'Angola': '  ',
         'Albania': '   ',
         'United Arab Emirates': '  ',
         'Argentina': '  ',
         'Armenia': '  ',
         'French Southern and Antarctic Lands': '     ',
         'Australia': '  ',
         'Austria': '  ',
         'Azerbaijan': '  ',
         'Burundi': '  ',
         'Belgium': '  ',
         'Benin': ' ',
         'Burkina Faso': '   ',
         'Bangladesh': '  ',
         'Bulgaria': '  ',
         'The Bahamas': '  ',
         'Bosnia and Herz.': '     ',
         'Belarus': '  ',
         'Belize': '  ',
         'Bermuda': '  ',
         'Bolivia': '  ',
         'Brazil': ' ',
         'Brunei': ' ',
         'Bhutan': ' ',
         'Botswana': '  ',
         'Central African Rep.': ' ',
         'Canada': '  ',
         'Switzerland': ' ',
         'Chile': ' ',
         'China': ' ',
         'Ivory Coast': '  ',
    'Cameroon': '  ',
    'Dem. Rep. Congo': '    ',
    'Congo': ' ',
    'Colombia': '   ',
    'Costa Rica': '   ',
    'Cuba': ' ',
    'N. Cyprus': '   ',
    'Cyprus': '  ',
    'Czech Rep.': ' ',
    'Germany': ' ',
    'Djibouti': '  ',
    'Denmark': ' ',
    'Algeria': '   ',
    'Ecuador': '  ',
    'Egypt': ' ',
    'Eritrea': '   ',
    'Spain': '  ',
    'Estonia': '  ',
    'Ethiopia': '   ',
    'Finland': ' ',
    'Fiji': ' ',
    'Falkland Islands': '   ',
    'France': ' ',
    'Gabon': ' ',
    'United Kingdom': ' ',
    'Georgia': '  ',
    'Ghana': ' ',
    'Guinea': '  ',
    'Gambia': '  ',
    'Guinea Bissau': '   ',
    'Eq. Guinea': '   ',
    'Greece': ' ',
    'Greenland': '  ',
    'Guatemala': '  ',
    'French Guiana': '   ',
    'Guyana': '  ',
    'Honduras': '  ',
    'Croatia': '  ',
    'Haiti': ' ',
    'Hungary': '  ',
    'Indonesia': '   ',
    'India': ' ',
    'Ireland': '  ',
    'Iran': ' ',
    'Iraq': '  ',
    'Iceland': ' ',
    'Israel': '  ',
    'Italy': '  ',
    'Jamaica': '  ',
    'Jordan': ' ',
    'Japan': ' ',
    'Kazakhstan': '   ',
    'Kenya': '  ',
    'Kyrgyzstan': '   ',
    'Cambodia': '  ',
    'Korea': ' ',
    'Kosovo': '  ',
    'Kuwait': '  ',
    'Lao PDR': ' ',
    'Lebanon': '  ',
    'Liberia': '  ',
    'Libya': '  ',
    'Sri Lanka': '  ',
    'Lesotho': '  ',
    'Lithuania': '  ',
    'Luxembourg': '  ',
    'Latvia': '  ',
    'Morocco': '  ',
    'Moldova': '  ',
    'Madagascar': '   ',
    'Mexico': '  ',
    'Macedonia': '  ',
    'Mali': ' ',
    'Myanmar': ' ',
    'Montenegro': ' ',
    'Mongolia': ' ',
    'Mozambique': '  ',
    'Mauritania': '   ',
    'Malawi': '  ',
    'Malaysia': '  ',
    'Namibia': '  ',
    'New Caledonia': '   ',
    'Niger': '  ',
    'Nigeria': '  ',
    'Nicaragua': '  ',
    'Netherlands': ' ',
    'Norway': ' ',
    'Nepal': '  ',
    'New Zealand': '  ',
    'Oman': ' ',
    'Pakistan': '  ',
    'Panama': '  ',
    'Peru': ' ',
    'Philippines': '  ',
    'Papua New Guinea': '    ',
    'Poland': ' ',
    'Puerto Rico': '  ',
    'Dem. Rep. Korea': ' ',
    'Portugal': '  ',
    'Paraguay': '  ',
    'Qatar': '  ',
    'Romania': '  ',
    'Russia': '  ',
    'Rwanda': '  ',
    'W. Sahara': '  ',
    'Saudi Arabia': '   ',
    'Sudan': ' ',
    'S. Sudan': '  ',
    'Senegal': '  ',
    'Solomon Is.': '   ',
    'Sierra Leone': '  ',
    'El Salvador': '  ',
    'Somaliland': '  ',
    'Somalia': '  ',
    'Serbia': '  ',
    'Suriname': '  ',
    'Slovakia': '  ',
    'Slovenia': '   ',
    'Sweden': ' ',
    'Swaziland': '  ',
    'Syria': '  ',
    'Chad': ' ',
    'Togo': ' ',
    'Thailand': ' ',
    'Tajikistan': '   ',
    'Turkmenistan': '   ',
    'East Timor': '  ',
    'Trinidad and Tobago': '    ',
    'Tunisia': '  ',
    'Turkey': '  ',
    'Tanzania': '  ',
    'Uganda': '  ',
    'Ukraine': '  ',
    'Uruguay': '  ',
    'United States': ' ',
    'Uzbekistan': '   ',
    'Venezuela': '  ',
    'Vietnam': ' ',
    'Vanuatu': '  ',
    'West Bank': ' ',
    'Yemen': ' ',
    'South Africa': ' ',
    'Zambia': '  ',
    'Zimbabwe': '   ',
    'Comoros': '  '
}

map = Map(init_opts=opts.InitOpts(width="1900px", height="900px",
                                  bg_color="#ADD8E6", page_title="    "))  #␣
↪
map.add("  ", [list(z) for z in zip(data_world['  '], data_world['  '])],
        is_map_symbol_show=False,  #
        #  name_map
        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),  #
)
map.render_notebook()  # notebook
```

```
[9]: <pyecharts.render.display.HTML at 0x1bd5d30c250>
```

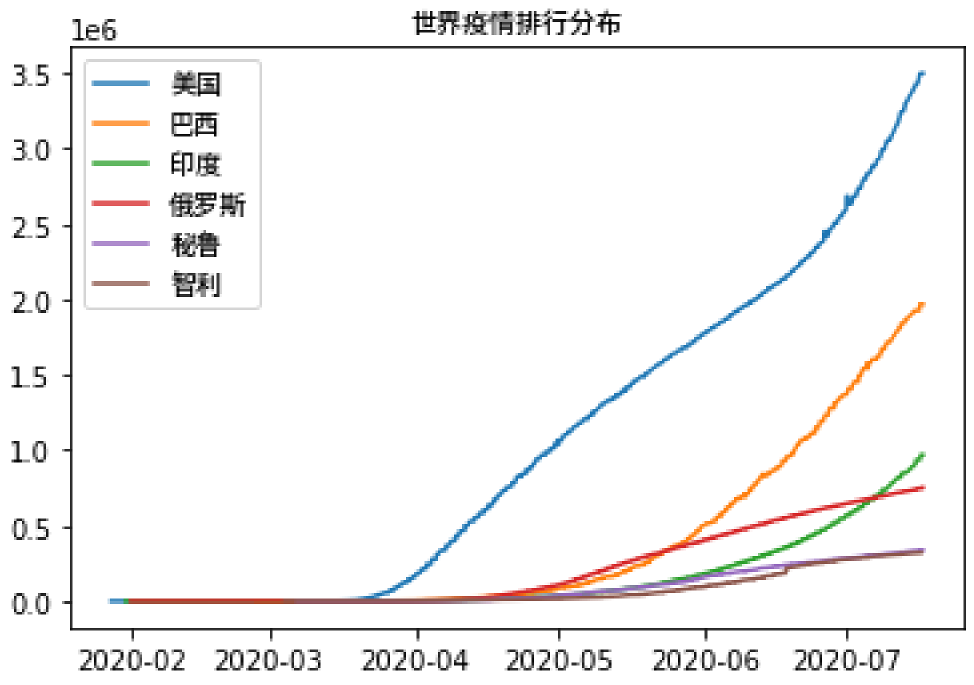
```
[10]: country = data_area_times.sort_values('province_confirmedCount',␣
      ↪ascending=False).drop_duplicates(
          subset='countryName', keep='first').head(6)['countryName']
      country = list(country)  #
      country
```

```
[10]: [' ', ' ', ' ', '  ', ' ', ' ']
```

```
[11]: data_America = data_area_times[data_area_times['countryName'] == '  ']
      data_Brazil = data_area_times[data_area_times['countryName'] == '  ']
      data_India = data_area_times[data_area_times['countryName'] == '  ']
      data_Russia = data_area_times[data_area_times['countryName'] == '   ']
      data_Peru = data_area_times[data_area_times['countryName'] == '  ']
      data_Chile = data_area_times[data_area_times['countryName'] == '  ']

      timeindex = data_area_times.index
      timeindex = timeindex.floor('D')  #
      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)  #
      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)
#
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)
#
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)
#
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)
#
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)
#
data_Chile.drop_duplicates(subset='updateTime', keep='first', inplace=True)
data_Chile.drop('updateTime', axis=1, inplace=True)

plt.title("    ", 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)
```

```
[11]: <matplotlib.legend.Legend at 0x1bd5d9b4ac0>
```



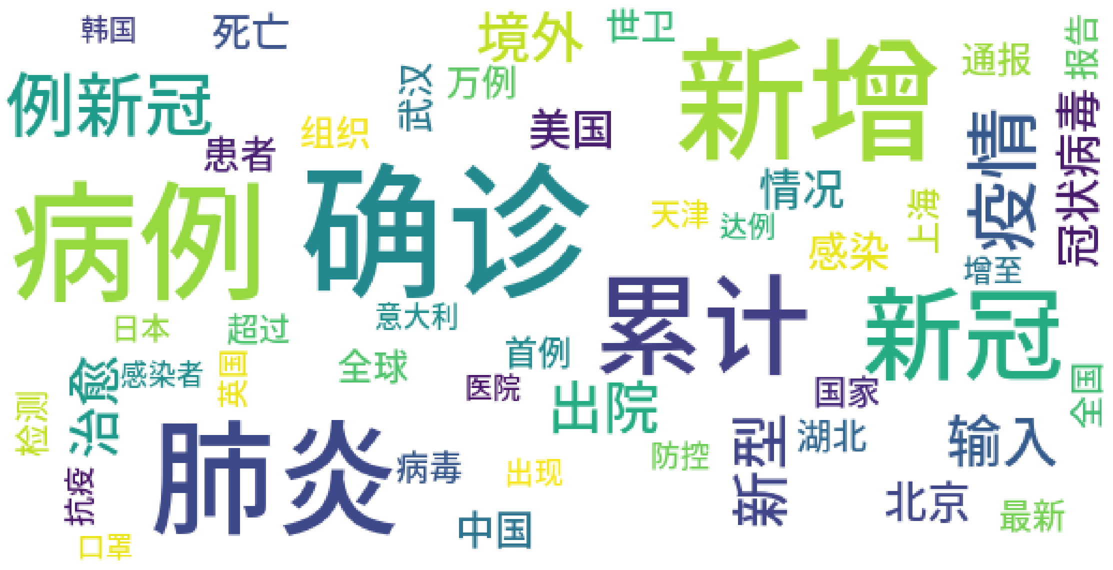
[12]:
```
pip install wordcloud==1.8.0
```

```
Collecting wordcloud==1.8.0
  Using cached wordcloud-1.8.0.tar.gz (217 kB)
  Preparing metadata (setup.py): started
  Preparing metadata (setup.py): finished with status 'done'
Requirement already satisfied: numpy>=1.6.1 in e:\anaconda3\lib\site-packages
(from wordcloud==1.8.0) (1.20.3)
Requirement already satisfied: pillow in e:\anaconda3\lib\site-packages (from
wordcloud==1.8.0) (8.4.0)
Requirement already satisfied: matplotlib in e:\anaconda3\lib\site-packages
(from wordcloud==1.8.0) (3.4.3)
Requirement already satisfied: pyparsing>=2.2.1 in e:\anaconda3\lib\site-
packages (from matplotlib->wordcloud==1.8.0) (3.0.4)
Requirement already satisfied: python-dateutil>=2.7 in e:\anaconda3\lib\site-
packages (from matplotlib->wordcloud==1.8.0) (2.8.2)
Requirement already satisfied: kiwisolver>=1.0.1 in e:\anaconda3\lib\site-
packages (from matplotlib->wordcloud==1.8.0) (1.3.1)
Requirement already satisfied: cycler>=0.10 in e:\anaconda3\lib\site-packages
(from matplotlib->wordcloud==1.8.0) (0.10.0)
Requirement already satisfied: six in e:\anaconda3\lib\site-packages (from
cycler>=0.10->matplotlib->wordcloud==1.8.0) (1.16.0)
Building wheels for collected packages: wordcloud
  Building wheel for wordcloud (setup.py): started
  Building wheel for wordcloud (setup.py): finished with status 'error'
  Running setup.py clean for wordcloud
Failed to build wordcloud
Installing collected packages: wordcloud
  Attempting uninstall: wordcloud
    Found existing installation: wordcloud 1.8.1
Note: you may need to restart the kernel to use updated packages.
  error: subprocess-exited-with-error

  python setup.py bdist_wheel did not run successfully.
  exit code: 1

  [20 lines of output]
  running bdist_wheel
  running build
  running build_py
  creating build
  creating build\lib.win-amd64-3.9
  creating build\lib.win-amd64-3.9\wordcloud
  copying wordcloud\color_from_image.py -> build\lib.win-amd64-3.9\wordcloud
  copying wordcloud\tokenization.py -> build\lib.win-amd64-3.9\wordcloud
  copying wordcloud\wordcloud.py -> build\lib.win-amd64-3.9\wordcloud
  copying wordcloud\wordcloud_cli.py -> build\lib.win-amd64-3.9\wordcloud
  copying wordcloud\_version.py -> build\lib.win-amd64-3.9\wordcloud
  copying wordcloud\__init__.py -> build\lib.win-amd64-3.9\wordcloud
  copying wordcloud\__main__.py -> build\lib.win-amd64-3.9\wordcloud
  copying wordcloud\stopwords -> build\lib.win-amd64-3.9\wordcloud
  copying wordcloud\DroidSansMono.ttf -> build\lib.win-amd64-3.9\wordcloud
  UPDATING build\lib.win-amd64-3.9\wordcloud/_version.py
  set build\lib.win-amd64-3.9\wordcloud/_version.py to '1.8.0'
  running build_ext
  building 'wordcloud.query_integral_image' extension
  error: Microsoft Visual C++ 14.0 or greater is required. Get it with
"Microsoft C++ Build Tools": https://visualstudio.microsoft.com/visual-cpp-
build-tools/
  [end of output]

  note: This error originates from a subprocess, and is likely not a problem
with pip.
  ERROR: Failed building wheel for wordcloud
  error: subprocess-exited-with-error

  Running setup.py install for wordcloud did not run successfully.
  exit code: 1

  [20 lines of output]
  running install
  running build
  running build_py
  creating build
  creating build\lib.win-amd64-3.9
  creating build\lib.win-amd64-3.9\wordcloud
  copying wordcloud\color_from_image.py -> build\lib.win-amd64-3.9\wordcloud
  copying wordcloud\tokenization.py -> build\lib.win-amd64-3.9\wordcloud
  copying wordcloud\wordcloud.py -> build\lib.win-amd64-3.9\wordcloud
  copying wordcloud\wordcloud_cli.py -> build\lib.win-amd64-3.9\wordcloud
  copying wordcloud\_version.py -> build\lib.win-amd64-3.9\wordcloud
  copying wordcloud\__init__.py -> build\lib.win-amd64-3.9\wordcloud
  copying wordcloud\__main__.py -> build\lib.win-amd64-3.9\wordcloud
  copying wordcloud\stopwords -> build\lib.win-amd64-3.9\wordcloud
  copying wordcloud\DroidSansMono.ttf -> build\lib.win-amd64-3.9\wordcloud
  UPDATING build\lib.win-amd64-3.9\wordcloud/_version.py
  set build\lib.win-amd64-3.9\wordcloud/_version.py to '1.8.0'
  running build_ext
  building 'wordcloud.query_integral_image' extension
  error: Microsoft Visual C++ 14.0 or greater is required. Get it with
"Microsoft C++ Build Tools": https://visualstudio.microsoft.com/visual-cpp-
build-tools/
  [end of output]

  note: This error originates from a subprocess, and is likely not a problem
with pip.
  WARNING: No metadata found in e:\anaconda3\lib\site-packages
error: legacy-install-failure

Encountered error while trying to install package.

wordcloud

note: This is an issue with the package mentioned above, not pip.
hint: See above for output from the failure.
WARNING: There was an error checking the latest version of pip.
```

```
    Uninstalling wordcloud-1.8.1:
      Successfully uninstalled wordcloud-1.8.1
  Running setup.py install for wordcloud: started
  Running setup.py install for wordcloud: finished with status 'error'
  Rolling back uninstall of wordcloud
  Moving to e:\anaconda3\lib\site-packages\wordcloud-1.8.1.dist-info\
   from E:\Anaconda3\Lib\site-packages\~ordcloud-1.8.1.dist-info
  Moving to e:\anaconda3\lib\site-packages\wordcloud\
   from E:\Anaconda3\Lib\site-packages\~ordcloud
  Moving to e:\anaconda3\scripts\wordcloud_cli.exe
   from C:\Users\12131\AppData\Local\Temp\pip-uninstall-
cjbz5v6u\wordcloud_cli.exe
```

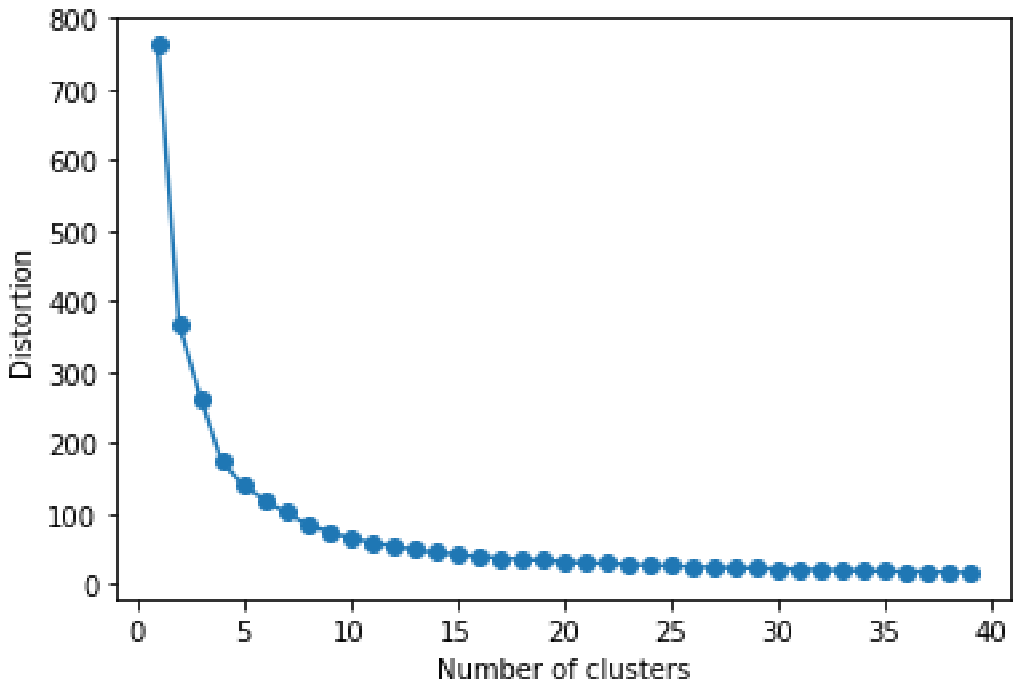
[14]:
```
import jieba
import re
from wordcloud import WordCloud

def word_cut(x): return jieba.lcut(x)

news = []
reg = "[^\u4e00-\u9fa5]"
for i in data_news['title']:
    if re.sub(reg, '', i) != '':
        news.append(re.sub(reg, '', i))

words = []
counts = {}
for i in news:
    words.append(word_cut(i))
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
words_sort = list(counts.items())
words_sort.sort(key=lambda x:[1],reverse=True)

newcloud = WordCloud(font_path="./NotoSansCJK.otf",
                     background_color="white",width=600,height=300,max_words=50)
newcloud.generate_from_frequencies(counts)
image = newcloud.to_image()
image
```

[14]:



```
[15]: 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, vector_size=300, window=5, min_count=5)  #
      keys = model.wv.key_to_index.keys()  #
      wordvector = []
      for key in keys:
          wordvector.append(model.wv[key])  #

      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)  #   1-40
          word_kmeans.fit(wordvector)
          distortions.append(word_kmeans.inertia_)  #

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

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



```
[16]: word_kmeans = KMeans(n_clusters=10)  #  10
      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])  #   10
          print(text)
```

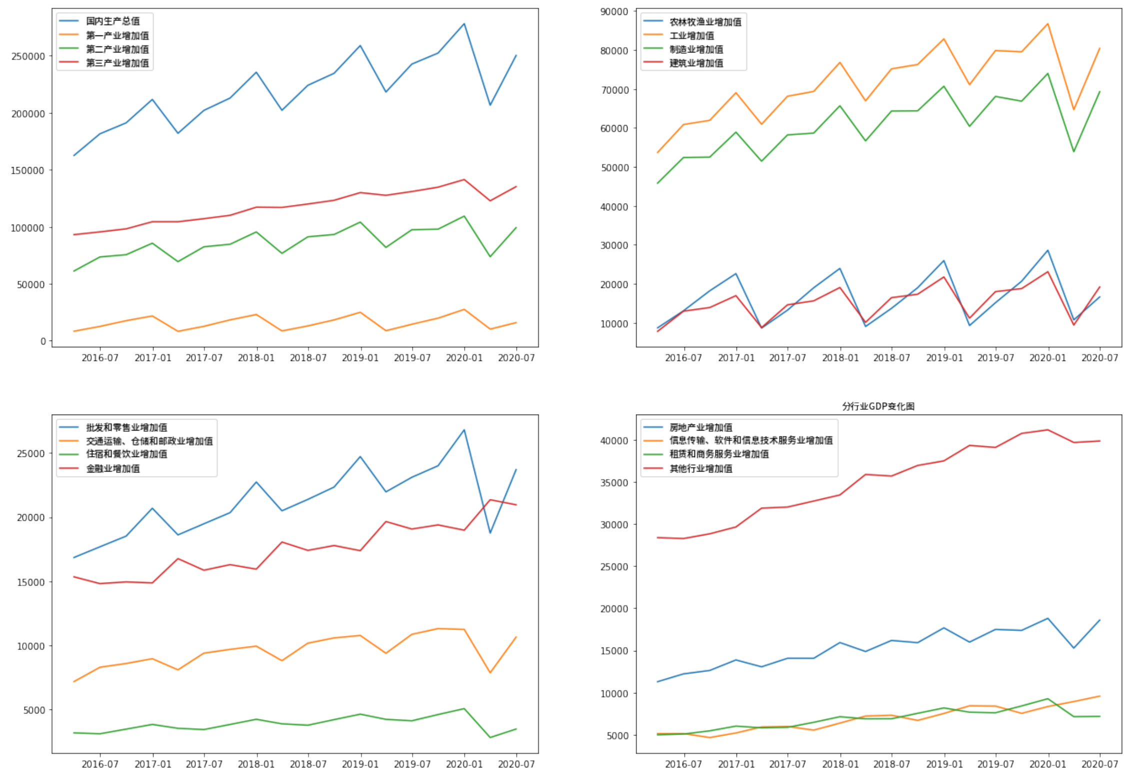
```
['  ', ' ', '  ', '  ', '  ', '  ', '   ', '  ', ' ', ' ', ' ', ' ', ' ',
' ', ' ', ' ', '  ', ' ', '  ', '  ', ' ', '  ', ' ', ' ', '   ',
'  ', ' ', ' ', ' ', ' ', '  ', ' ', ' ', ' ', ' ', ' ', ' ', ' ',
'  ', '', ' ', ' ', '  ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ',
'  ', ' ', ' ', ' ', '', ' ', ' ', '', ' ', '', ' ', ' ', ' ',
'  ', ' ', '', ' ', '  ', ' ', ' ', ' ', ' ', '   ', ' ', ' ', ' ',
'  ', ' ', ' ', '', ' ', '   ', ' ', ' ', '   ', ' ', '', ' ', ' ',
'  ', ' ', ' ', ' ', '', ' ', '  ', ' ', '', ' ', ' ', ' ', '',
'  ', '   ', ' ', ' ', ' ', ' ', '', '', ' ', ' ', ' ', '  ', ' ',
'  ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '', '  ', ' ', ' ',
'  ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '', ' ',
'  ', ' ', ' ', ' ', ' ', '', ' ', ' ', ' ', ' ', '', ' ', '   ',
'  ', '  ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '', '', ' ',
' ', ' ', ' ', ' ', ' ', ' ', '  ', ' ', ' ', ' ', ' ', ' ', '  ',
' ', ' ', ' ', ' ', ' ', '  ', ' ', ' ', ' ', ' ', ' ', ' ', ' ',
' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ']
[' ', '  ', ' ', ' ', ' ', ' ', ' ', '  ', ' ', ' ', ' ', '  ',
' ', '  ', ' ', ' ', ' ', ' ', '  ', ' ', ' ', ' ', ' ', ' ', ' ',
'  ', ' ', ' ', '  ', ' ', ' ', ' ', '  ', ' ', ' ', ' ', ' ',
' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '  ',
' ', ' ', '  ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ',
' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '  ', ' ', ' ',
' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ',
' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '  ',
'', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '  ',
'  ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '  ', ' ', ' ']
[' ', '  ', ' ', ' ', '  ', ' ', '  ', '  ', '  ', '  ', '   ',
' ', ' ', '  ', ' ', '  ', '  ', '  ', '  ', '  ', '  ', '   ',
'  ', ' ', '  ', ' ', '  ', '  ', '  ', '   ', '  ', '  ', '   ',
'  ', ' ', '  ', ' ', ' ', ' ', '   ', '   ', ' ', '  ', ' ',
' ', ' ', ' ', '  ', ' ', '  ', '    ', '  ', ' ', '  ', '  ',
'  ', '  ', '  ', ' ', ' ', '  ', ' ', ' ', ' ', '  ', ' ', ' ',
' ', '  ', ' ', ' ', ' ', '  ', ' ', ' ', '  ', ' ', ' ', ' ',
'  ', ' ', '  ', '  ', ' ', '  ', ' ', '   ', ' ', ' ', ' ', '  ',
' ', ' ', ' ', ' ', ' ', ' ', '  ', ' ', ' ', ' ', ' ', ' ', ' ',
' ', ' ', '  ', ' ', ' ', ' ', '  ', ' ', ' ', ' ', ' ', ' ', ' ', ' ',
' ', '  ', ' ', ' ', '  ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '  ',
'  ', ' ', ' ', '  ', ' ', ' ', '  ', '  ', '  ', ' ', ' ', ' ',
' ', ' ', ' ', ' ', ' ', ' ', ' ', '  ', ' ', ' ', ' ', '  ', ' ',
' ', ' ', ' ', ' ', ' ', ' ', '  ', ' ', ' ', ' ', ' ', ' ', ' ',
'  ', '  ', ' ', ' ', '  ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '  ',
' ', ' ', ' ', ' ', ' ', '  ', ' ', ' ', '  ', ' ', '  ', ' ',
' ', '  ', ' ', ' ', ' ', '  ', ' ', '  ', ' ', ' ', ' ', ' ', '  ',
'  ', ' ', ' ', '    ', '   ', ' ', ' ', '  ', ' ', ' ', ' ',
' ', ' ', ' ', '   ', ' ', '  ', ' ', ' ', ' ', '  ', ' ', '  ',
' ', ' ', ' ', ' ', ' ', ' ', '  ', ' ', ' ', ' ', ' ', ' ', ' ',
' ', ' ', ' ', ' ', ' ', ' ', ' ', '  ', ' ', ' ', ' ', ' ', ' ',
'   ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ',
' ', ' ', '  ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ',
' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ',
' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '  ', ' ', ' ', ' ',
'  ', ' ', ' ', ' ', ' ', ' ', ' ', '  ', ' ', '  ', ' ', ' ',
' ', ' ', ' ', ' ', ' ', ' ', '  ', ' ', ' ', ' ', ' ', ' ', ' ', ' ',
'', '  ', ' ', ' ', '  ', ' ', ' ']
[' ', '  ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ']
['', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ',
'', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ',
'', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ',
' ', ' ', ' ', ' ', ' ', ' ', '  ', ' ', ' ', ' ', ' ', ' ', ' ', ' ',
' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ']
[' ', ' ', ' ', '  ', ' ', ' ', '  ', '  ', '  ', '  ', '  ',
'  ', '  ', ' ', ' ', '  ', '  ', ' ', '  ', '  ', ' ', '  ', '  ',
' ', ' ', ' ', ' ', ' ', '   ', '   ', '    ', '  ', ' ', ' ',
' ', '  ', ' ', ' ', ' ', '  ', ' ', '  ', '  ', ' ', ' ', ' ', ' ',
' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '  ', '  ', ' ',
' ', ' ', ' ', ' ', '  ', '  ', ' ', ' ', ' ', ' ', ' ', ' ', ' ',
' ', ' ', ' ', ' ', ' ', '  ', ' ', ' ', ' ', ' ', ' ', '    ', ' ',
' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ',
' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ',
' ', ' ', '  ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '  ',
' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ',
'  ', ' ', ' ', ' ', '  ', ' ', ' ', ' ', ' ', ' ', ' ', ' ',
'   ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '   ', ' ', ' ',
' ', ' ', ' ', ' ', ' ', ' ', '  ', '  ', ' ', ' ', ' ', ' ', ' ',
' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '  ',
' ', ' ', ' ', ' ', ' ', ' ', ' ', '  ', '  ', ' ', ' ', ' ', ' ',
'  ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ',
' ', ' ', ' ', ' ', ' ', ' ', '  ', ' ', ' ', ' ', ' ', ' ', ' ',
' ', ' ', ' ', ' ', ' ', ' ', ' ', '  ', ' ', ' ', '  ', ' ', ' ',
' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ',
' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ',
' ', ' ', ' ', ' ', ' ', ' ', ' ', '  ', ' ', ' ', ' ', ' ', ' ',
' ', ' ', ' ', '    ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ',
' ', ' ', ' ', ' ', ' ', ' ', '   ', ' ', ' ', ' ', ' ', ' ', ' ',
' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ',
' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ',
' ', ' ', ' ', ' ', '  ', ' ', ' ', '  ', ' ', '  ', ' ', '  ', ' ',
' ', ' ']
[' ', ' ', ' ', ' ', ' ', ' ']
[' ', ' ', ' ', ' ', ' ', ' ']
[' ', ' ', ' ', ' ', ' ', ' ', '  ', ' ', ' ', '  ', '  ', ' ',
' ', ' ', ' ', ' ', ' ', '  ', ' ', ' ', ' ', '  ', ' ', '  ', ' ',
' ', ' ', ' ', ' ', ' ', ' ', ' ', '  ', ' ', ' ', ' ', ' ', '  ',
' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ']
[' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ',
' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ']
```

```
[17]: sum_GDP = ['    ', '     ', '     ', '    ']
      industry_GDP = ['     ', '   ', '    ', '   ']
      industry2_GDP = ['      ', '         ', '      ', '    ']
      industry3_GDP = ['     ', '          ',
                       '      ', '   ']  #

      fig = plt.figure()
      fig, axes = plt.subplots(2, 2, figsize=(21, 15))  #

      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('  GDP  ', fontproperties=myfont)
```

[17]:
```
Text(0.5, 1.0, '  GDP  ')
```

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



[18]:
```
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['    '][:-2])  #     16
a, b = arma_order_select_ic(data_arma, ic='hqic')['hqic_min_order']
arma = ARMA(data_arma, order=(a, b)).fit()  #  ARMA
rate1 = list(data_economy['    '][-2] /
             arma.forecast(steps=1)[0])  #
rate1  #
```

```
[18]: [0.8273539514507257]
```

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

      c = (
          Liquid()
          .add("  /  ", rate1, is_outline_show=False)
          .set_global_opts(title_opts=opts.TitleOpts(title="          ",
                                                     pos_left="center"))
      )
      c.render_notebook()
```

```
[19]: <pyecharts.render.display.HTML at 0x1bd6685e070>
```

```
[20]: warnings.filterwarnings('ignore')
      data_arma = pd.DataFrame(data_economy['   '][:-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['   '][-2]/arma.forecast(steps=1)[0])
      c = (
          Liquid()
          .add("  /  ", rate2, is_outline_show=False)
          .set_global_opts(title_opts=opts.TitleOpts(title="    ", pos_left="center"))
      )
      c.render_notebook()
```

```
[20]: <pyecharts.render.display.HTML at 0x1bd67335e80>
```

```
[21]: warnings.filterwarnings('ignore')
      data_arma = pd.DataFrame(data_economy['    '][:-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['    '][-2]/arma.forecast(steps=1)[0])
      c = (
          Liquid()
          .add("  /  ", rate3, is_outline_show=False)
          .set_global_opts(title_opts=opts.TitleOpts(title="   ", pos_left="center"))
      )
      c.render_notebook()
```

```
[21]: <pyecharts.render.display.HTML at 0x1bd6737dfd0>
```

```
[22]: data_arma = pd.DataFrame(data_economy['    '][:-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['    '][-2]/arma.forecast(steps=1)[0])
```

```
c = (
    Liquid()
    .add("  /  ", rate, is_outline_show=False)
    .set_global_opts(title_opts=opts.TitleOpts(title="    ", pos_left="center"))
)
c.render_notebook()
```

```
[22]: <pyecharts.render.display.HTML at 0x1e2c9cf9910>
```

```
[ ]:
```