coronavirus_analysis_v0_2020-03-14

March 14, 2020

[1]:
```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt
import matplotlib
from scipy.optimize import curve_fit
import datetime
import matplotlib.dates as mdates
```

[2]:
```
matplotlib.rcParams["figure.figsize"] = 20,9
import warnings
warnings.filterwarnings('ignore')
```

1 Load the data

[3]:
```
# import data from github url, url1 -Confirmed, url2 -Deaths. url3 -Recovered
data_path = './data/COVID-19/csse_covid_19_data/csse_covid_19_time_series/'
file1 = data_path + 'time_series_19-covid-Confirmed.csv'
file2 = data_path + 'time_series_19-covid-Deaths.csv'
file3 = data_path + 'time_series_19-covid-Recovered.csv'

df1 = pd.read_csv(file1, error_bad_lines=False)
df2 = pd.read_csv(file2, error_bad_lines=False)
df3 = pd.read_csv(file3, error_bad_lines=False)

#display(df1.head(), df2.head(), df3.head())
```

Convert date columns to a time series

[4]:
```
df1u = pd.melt(df1, id_vars=['Province/State', 'Country/Region', 'Lat','Long'],␣
↪var_name='DateTime', value_name='Confirmed')
df2u = pd.melt(df2, id_vars=['Province/State', 'Country/Region', 'Lat','Long'],␣
↪var_name='DateTime', value_name='Deaths')
df3u = pd.melt(df3, id_vars=['Province/State', 'Country/Region', 'Lat','Long'],␣
↪var_name='DateTime', value_name='Recovered')

df1u['DateTime'] = pd.to_datetime(df1u['DateTime'])
```

```
df2u['DateTime'] = pd.to_datetime(df2u['DateTime'])
df3u['DateTime'] = pd.to_datetime(df3u['DateTime'])

data = pd.concat([df1u, df2u, df3u], axis=1, join='inner')
data = data.loc[:,~data.columns.duplicated()] # Remove columns that were␣
 ↪repeated in the three datasets
data['mortality_fraction'] = data['Deaths']/data['Confirmed']
data['Province/State'][data['Province/State'].isna()] = ''
```

[5]:
```
data
```

[5]:
```
       Province/State       Country/Region      Lat      Long   DateTime  \
0                                 Thailand  15.0000  101.0000 2020-01-22
1                                    Japan  36.0000  138.0000 2020-01-22
2                                Singapore   1.2833  103.8333 2020-01-22
3                                    Nepal  28.1667   84.2500 2020-01-22
4                                 Malaysia   2.5000  112.5000 2020-01-22
…                 …                    …       …        …        …
21887                                Aruba  12.5211  -69.9683 2020-03-13
21888  Grand Princess                Canada  37.6489 -122.6655 2020-03-13
21889                                Kenya  -0.0236   37.9062 2020-03-13
21890                  Antigua and Barbuda  17.0608  -61.7964 2020-03-13
21891         Alabama                   US  32.3182  -86.9023 2020-03-13

       Confirmed  Deaths  Recovered  mortality_fraction
0              2       0          0                 0.0
1              2       0          0                 0.0
2              0       0          0                 NaN
3              0       0          0                 NaN
4              0       0          0                 NaN
…            …     …        …                   …
21887          2       0          0                 0.0
21888          2       0          0                 0.0
21889          1       0          0                 0.0
21890          1       0          0                 0.0
21891          5       0          0                 0.0

[21892 rows x 9 columns]
```

I need to remove a few points manually that I identified as bad ones

[29]:
```
data.drop(data[(data['Country/Region']=='Spain') & (data['DateTime'] ==␣
 ↪datetime.datetime.strptime('2020-03-12', '%Y-%m-%d'))].index, inplace=True)
data.drop(data[(data['Country/Region']=='Italy') & (data['DateTime'] ==␣
 ↪datetime.datetime.strptime('2020-03-12', '%Y-%m-%d'))].index, inplace=True)
```

```
data.drop(data[(data['Country/Region']=='United Kingdom') & (data['Province/
↪State']=='United Kingdom') & (data['DateTime'] == datetime.datetime.
↪strptime('2020-03-12', '%Y-%m-%d'))].index, inplace=True)
```

Names of available countries (Note, some countries are divided in province/State

[7]:
```
data['Country/Region'].unique()
```

[7]:
```
array(['Thailand', 'Japan', 'Singapore', 'Nepal', 'Malaysia', 'Canada',
       'Australia', 'Cambodia', 'Sri Lanka', 'Germany', 'Finland',
       'United Arab Emirates', 'Philippines', 'India', 'Italy', 'Sweden',
       'Spain', 'Belgium', 'Egypt', 'Lebanon', 'Iraq', 'Oman',
       'Afghanistan', 'Bahrain', 'Kuwait', 'Algeria', 'Croatia',
       'Switzerland', 'Austria', 'Israel', 'Pakistan', 'Brazil',
       'Georgia', 'Greece', 'North Macedonia', 'Norway', 'Romania',
       'Estonia', 'Netherlands', 'San Marino', 'Belarus', 'Iceland',
       'Lithuania', 'Mexico', 'New Zealand', 'Nigeria', 'Ireland',
       'Luxembourg', 'Monaco', 'Qatar', 'Ecuador', 'Azerbaijan',
       'Armenia', 'Dominican Republic', 'Indonesia', 'Portugal',
       'Andorra', 'Latvia', 'Morocco', 'Saudi Arabia', 'Senegal',
       'Argentina', 'Chile', 'Jordan', 'Ukraine', 'Hungary',
       'Liechtenstein', 'Poland', 'Tunisia', 'Bosnia and Herzegovina',
       'Slovenia', 'South Africa', 'Bhutan', 'Cameroon', 'Colombia',
       'Costa Rica', 'Peru', 'Serbia', 'Slovakia', 'Togo',
       'French Guiana', 'Malta', 'Martinique', 'Bulgaria', 'Maldives',
       'Bangladesh', 'Paraguay', 'Albania', 'Cyprus', 'Brunei', 'US',
       'Burkina Faso', 'Holy See', 'Mongolia', 'Panama', 'China', 'Iran',
       'Korea, South', 'France', 'Cruise Ship', 'Denmark', 'Czechia',
       'Taiwan*', 'Vietnam', 'Russia', 'Moldova', 'Bolivia', 'Honduras',
       'United Kingdom', 'Congo (Kinshasa)', "Cote d'Ivoire", 'Jamaica',
       'Reunion', 'Turkey', 'Cuba', 'Guyana', 'Kazakhstan',
       'Cayman Islands', 'Guadeloupe', 'Ethiopia', 'Sudan', 'Guinea',
       'Aruba', 'Kenya', 'Antigua and Barbuda'], dtype=object)
```

List of most affected countries (sorted by number of confirmed cases)

[8]:
```
data.groupby(['Country/Region'])[['Confirmed', 'Deaths', 'Recovered']].sum().
↪sort_values(by='Confirmed')[::-1][0:15]
```

[8]:

Country/Region	Confirmed	Deaths	Recovered
China	2719449	87028	1089238
Korea, South	96672	661	2339
Italy	84484	4505	6893
Iran	75645	2855	21060
Cruise Ship	19172	116	1269
France	15660	304	258
Germany	14611	17	439

Spain	12851	288	542
US	9163	263	191
Japan	8380	146	1270
Switzerland	4492	29	35
Singapore	3683	0	1769
United Kingdom	3643	42	294
Norway	3574	0	5
Netherlands	3274	32	0

List of most affected countries/provinces (sorted by number of deaths)

[9]:
```
data.groupby(['Country/Region', 'Province/State'])[['Confirmed', 'Deaths',␣
↪'Recovered']].sum().sort_values(by='Deaths')[::-1][0:15]
```

[9]:

Country/Region	Province/State	Confirmed	Deaths	Recovered
China	Hubei	2216904	83706	809753
Italy		84484	4505	6893
Iran		75645	2855	21060
Korea, South		96672	661	2339
China	Henan	49080	636	29745
	Heilongjiang	17162	428	8220
France	France	15627	304	258
Spain		12851	288	542
China	Beijing	16200	202	7220
	Chongqing	22897	197	12268
	Anhui	37747	191	21849
	Guangdong	53211	185	28163
	Hebei	11591	184	7333
	Hainan	6564	177	3739
	Shandong	25981	151	13117

[42]:
```
def select_country(data, country='', province='', start_date='2020-02-15',␣
↪end_date='2021-03-11', show_numbers=False):
    print(f'{country} {province}')
    # Select by country
    if country != '':
        cond1 = data['Country/Region'] == country
    else:
        cond1 = data['Country/Region'].notna()  # Select any country
    # Select by province
    if province != '':
        cond2 = data['Province/State'] == province
    else:
        cond2 = data['Province/State'] != np.nan  # Select any province
    # Select by time range
    if start_date != '':
```

```
        t0 = datetime.datetime.strptime(start_date, '%Y-%m-%d')
    else:
        t0 = datetime.datetime.strptime('2000-01-01', '%Y-%m-%d')
    if end_date != '':
        t1 = datetime.datetime.strptime(end_date, '%Y-%m-%d')
    else:
        t1 = datetime.datetime.strptime('9999-01-01', '%Y-%m-%d')
    cond3 = (data['DateTime'] >= t0) & (data['DateTime'] < t1)
    selection = cond1&cond2&cond3
    if show_numbers:
        print('Number of entries per country: {}'.format(cond1.sum()))
        print('Number of entries per province: {}'.format(cond2.sum()))
        print('Number of entries per time range: {}'.format(cond3.sum()))
        print('Number of entries selected: {}'.format(selection.sum()))
    return data[selection]
```

[43]:
```
def my_exponential(t,  b, alpha):
    return  b * np.exp(alpha * (t-t[0])/t[0])

def my_logistic(x, L, k, x0):
    return L/(1+np.exp(-k*(x-x0)))

def fit_data(data):
    # Will try to fit a logistic growth. If not possible fit an exponential
    for column in ['Confirmed', 'Deaths', 'Recovered']:
        x = mdates.date2num(data['DateTime'])
        y = data[column].values
        try:
            popt, pcov = curve_fit(my_logistic, x, y, p0=(np.max(y), 1, x[-3]))
            func = my_logistic
        except RuntimeError:
                popt, pcov = curve_fit(my_exponential, x, y, p0=(1., 1))
                func = my_exponential
        c_fit = func(x, *popt)
        data.loc[:, column+'_fit'] = c_fit
    return data

#def fit_data(data, fit='exp'):
#    # Choose between exponential or logistic fit
#    for column in ['Confirmed', 'Deaths', 'Recovered']:
#        try:
#            x = mdates.date2num(data['DateTime'])
#            y = data[column].values
#            if fit=='exp':
#                popt, pcov = curve_fit(my_exponential, x, y, p0=(1., 1))
#                func = my_exponential
#            elif fit=='logistic':
```

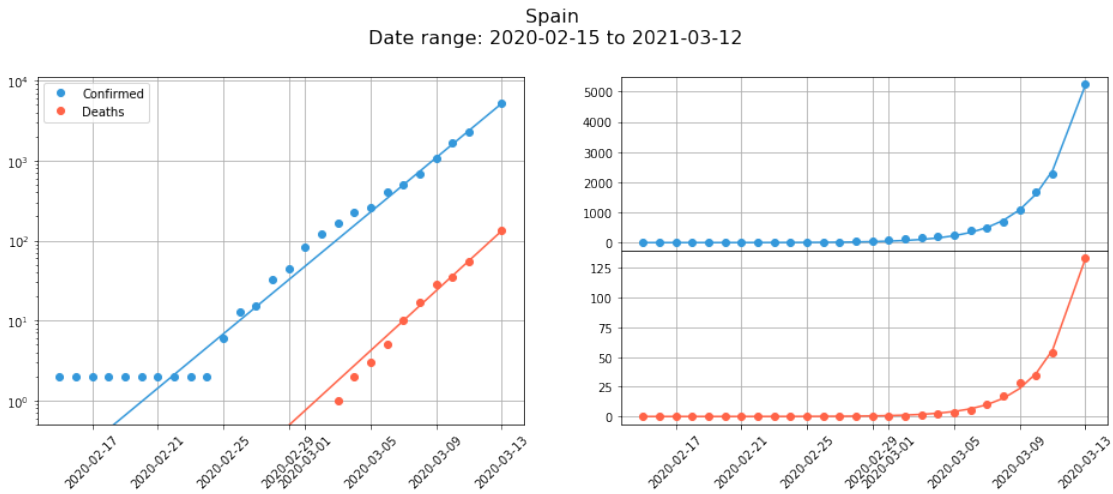
```
#                 popt, pcov = curve_fit(my_logistic, x, y, p0=(np.max(y), 1,␣
 ↪x[-3]))
#                 func = my_logistic
#             c_fit = func(x, *popt)
#             data.loc[:, column+'_fit'] = c_fit
#         except RuntimeError:
#             data.loc[:, column+'_fit'] = np.nan
#     return data
```

[46]:
```
def plot_country(data, country='Spain', province='', start_date='2020-02-15',␣
 ↪end_date='2021-03-12', show_numbers=False):
    data_country = fit_data(select_country(data, country=country,␣
 ↪province=province, start_date=start_date, end_date=end_date,␣
 ↪show_numbers=show_numbers))

    gridsize = (3, 2)
    fig = plt.figure(figsize=(16, 8))
    ax1 = plt.subplot2grid(gridsize, (0, 0), colspan=1, rowspan=2)
    ax2 = plt.subplot2grid(gridsize, (0, 1))
    ax3 = plt.subplot2grid(gridsize, (1, 1))
    plt.subplots_adjust(hspace=0)

    ax1.plot(data_country['DateTime'], data_country['Confirmed'], 'o',␣
 ↪color='#3498db', label='Confirmed')
    ax1.plot(data_country['DateTime'], data_country['Deaths'], 'o',␣
 ↪color='tomato', label='Deaths')
    ax1.plot(data_country['DateTime'], data_country['Confirmed_fit'], '-',␣
 ↪color='#3498db')
    ax1.plot(data_country['DateTime'], data_country['Deaths_fit'], '-',␣
 ↪color='tomato')
    ax2.plot(data_country['DateTime'], data_country['Confirmed'], 'o',␣
 ↪color='#3498db')
    ax2.plot(data_country['DateTime'], data_country['Confirmed_fit'], '-',␣
 ↪color='#3498db')
    ax3.plot(data_country['DateTime'], data_country['Deaths'], 'o',␣
 ↪color='tomato')
    ax3.plot(data_country['DateTime'], data_country['Deaths_fit'], '-',␣
 ↪color='tomato')

    ax1.xaxis.set_tick_params(rotation=45)
    ax2.xaxis.set_ticklabels([])
    ax3.xaxis.set_tick_params(rotation=45)
    ax1.grid()
    ax2.grid()
    ax3.grid()
```

```
    fig.suptitle(f'{country} {province}\nDate range: {start_date} to␣
↪{end_date}', fontsize=16)
    #ax1.set_xscale('log')
    ax1.set_yscale('log')
    ax1.legend()
    ax1.set_ylim(0.5,)
    fig.savefig(f'./plots/{country}{province}.png', bbox_inches='tight',␣
↪dpi=150)
    return data_country
```

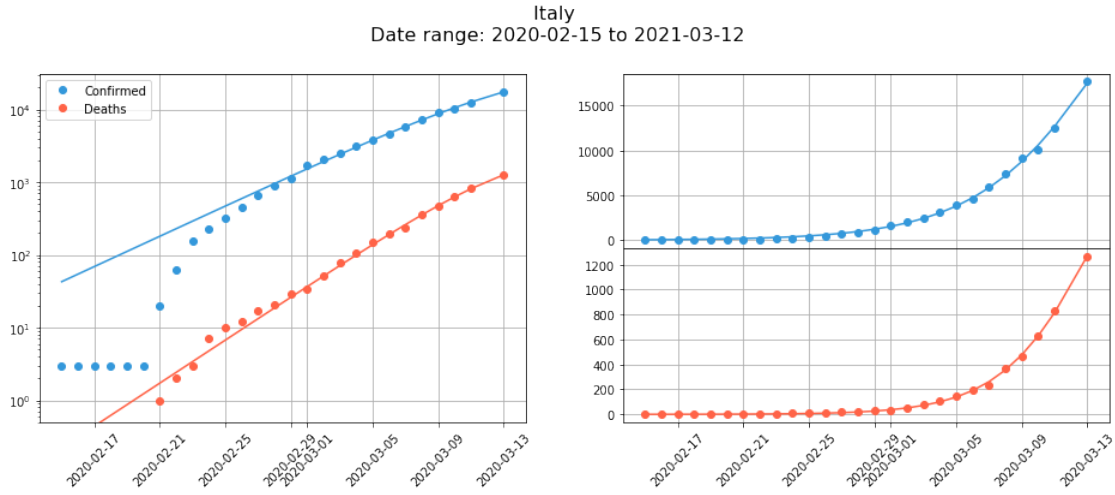
[47]:
```
spain = plot_country(data, country='Spain')
```

```
Spain
```



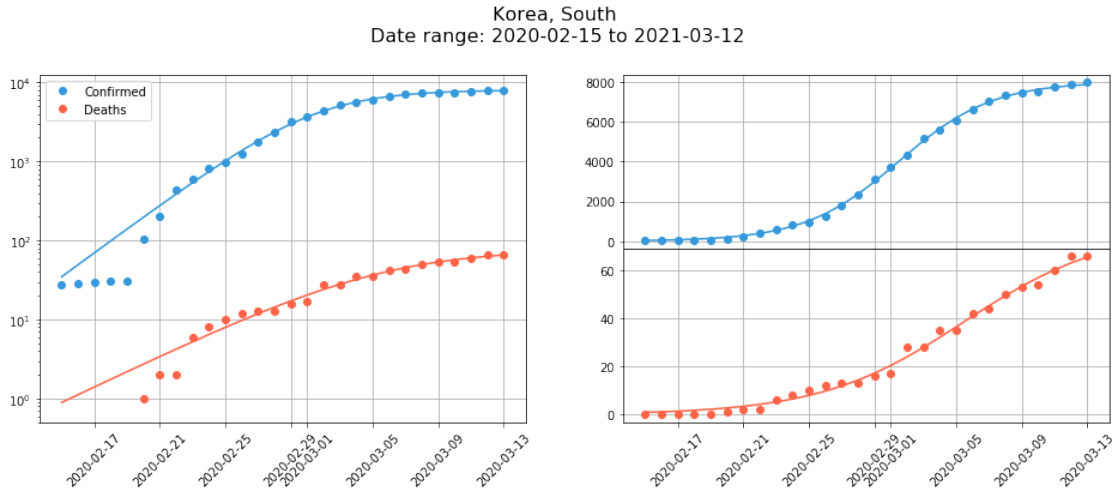
[48]:
```
italy = plot_country(data, country='Italy')
```

```
Italy
```



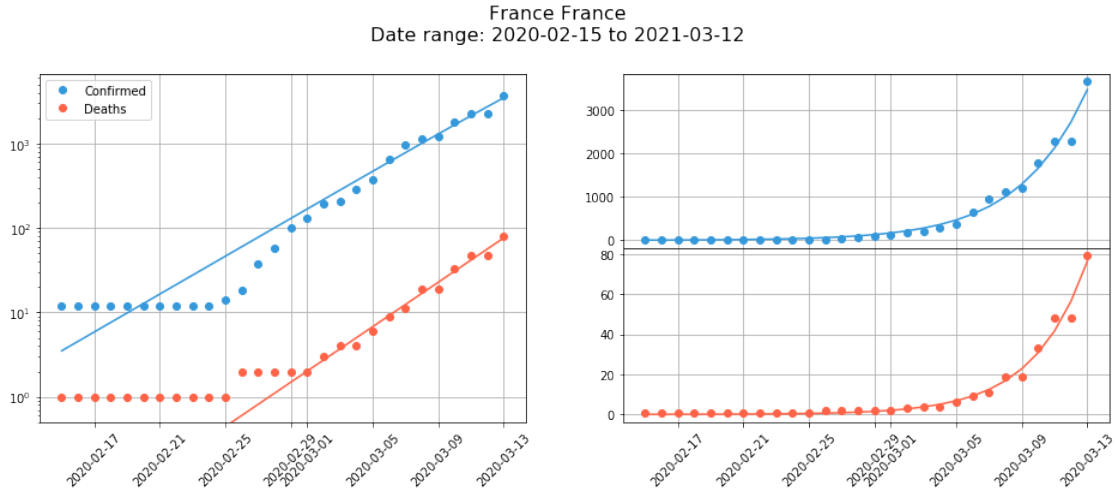
[49]:
```
korea = plot_country(data, country='Korea, South')
```

```
Korea, South
```



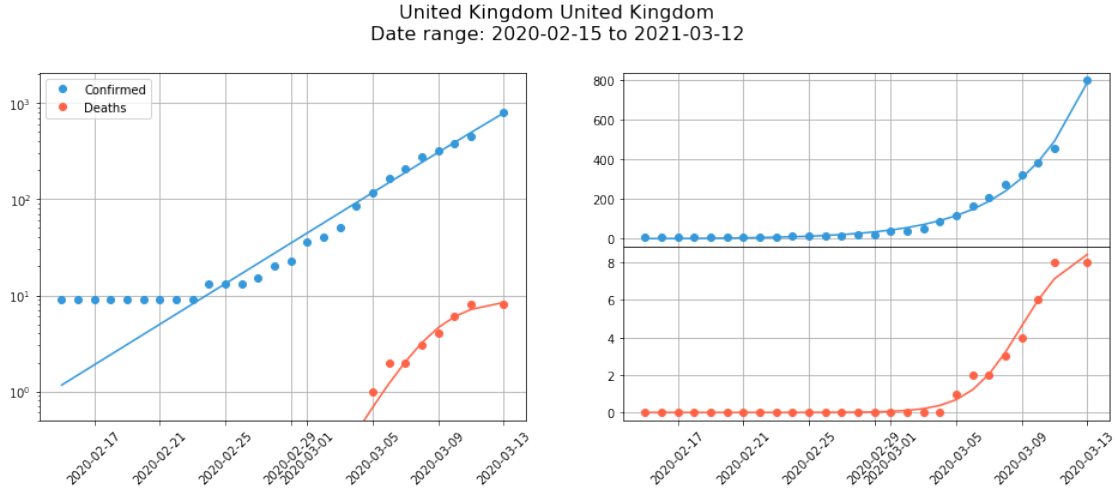
[50]:
```
france = plot_country(data, country='France', province='France')
```

```
France France
```



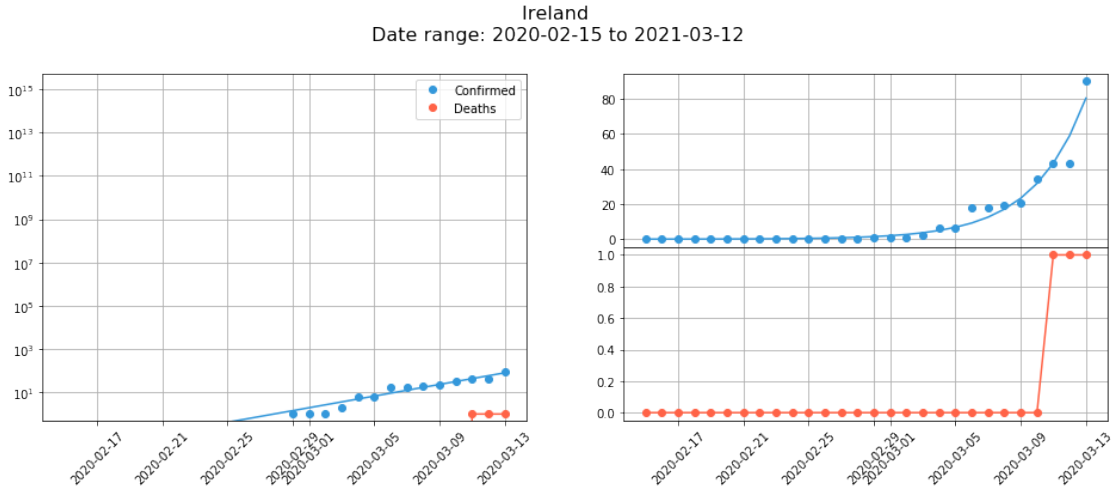
[51]:
```
uk = plot_country(data, country='United Kingdom', province='United Kingdom')
```

```
United Kingdom United Kingdom
```



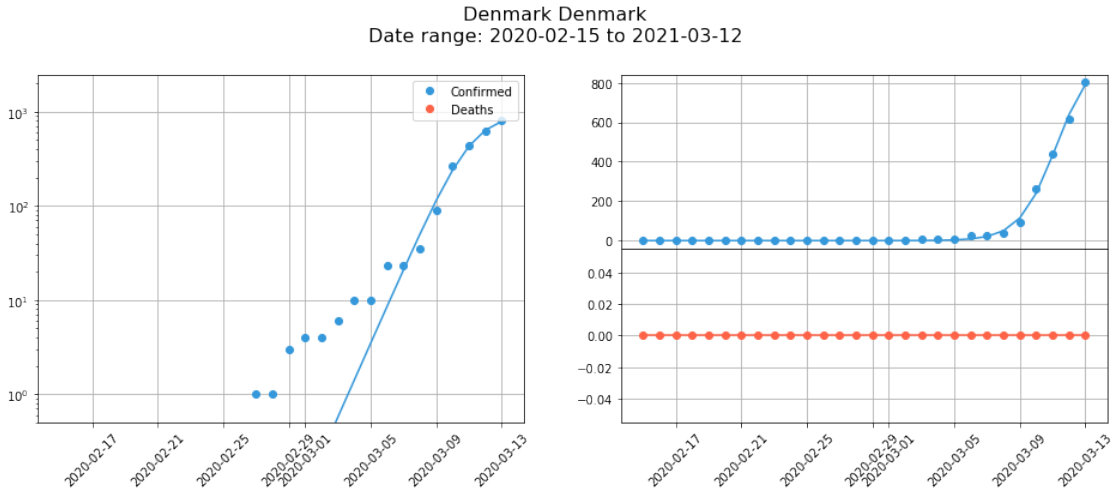
[52]:
```
ireland = plot_country(data, country='Ireland')
```

```
Ireland
```



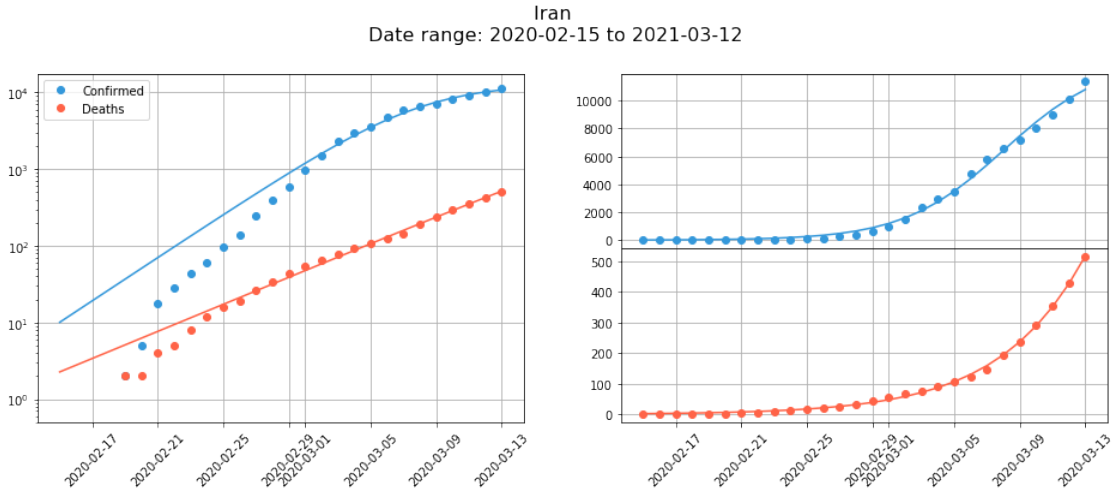
[53]:
```
denmark = plot_country(data, country='Denmark', province='Denmark')
```

```
Denmark Denmark
```



[54]:
```
iran = plot_country(data, country='Iran')
```

```
Iran
```



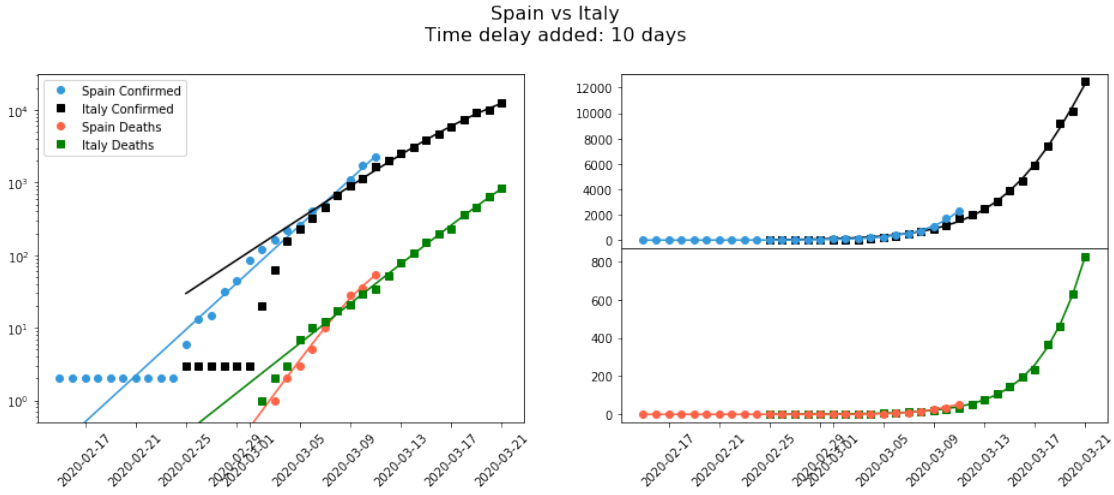
2 1-1 Comparison of two countries, with a manual time delay to align them

[55]:
```
def compare_countries(data, country1='Spain', province1='',␣
↪start_date1='2020-02-15', end_date1='2020-03-12',
                      country2='Spain', province2='',␣
↪start_date2='2020-02-15', end_date2='2020-03-12',
                      delay=0, show_numbers=False):
   data_country1 = fit_data(select_country(data, country=country1,␣
↪province=province1, start_date=start_date1, end_date=end_date1,␣
↪show_numbers=False))
   data_country2 = fit_data(select_country(data, country=country2,␣
↪province=province2, start_date=start_date2, end_date=end_date2,␣
↪show_numbers=False))

   gridsize = (3, 2)
   fig = plt.figure(figsize=(16, 8))
   ax1 = plt.subplot2grid(gridsize, (0, 0), colspan=1, rowspan=2)
   ax2 = plt.subplot2grid(gridsize, (0, 1))
   ax3 = plt.subplot2grid(gridsize, (1, 1))
   plt.subplots_adjust(hspace=0)
   ax1.plot(data_country1['DateTime'], data_country1['Confirmed'], 'o',␣
↪color='#3498db', label=country1+' Confirmed')
   ax1.plot(data_country1['DateTime'], data_country1['Confirmed_fit'], '-',␣
↪color='#3498db')
   ax1.plot(data_country2['DateTime']+datetime.timedelta(days=delay),␣
↪data_country2['Confirmed'], 's', color='k', label=country2+' Confirmed')
```

```
    ax1.plot(data_country2['DateTime']+datetime.timedelta(days=delay),␣
↪data_country2['Confirmed_fit'], '-', color='k')
    ax1.plot(data_country1['DateTime'], data_country1['Deaths'], 'o',␣
↪color='tomato', label=country1+' Deaths')
    ax1.plot(data_country1['DateTime'], data_country1['Deaths_fit'], '-',␣
↪color='tomato')
    ax1.plot(data_country2['DateTime']+datetime.timedelta(days=delay),␣
↪data_country2['Deaths'], 's', color='g', label=country2+' Deaths')
    ax1.plot(data_country2['DateTime']+datetime.timedelta(days=delay),␣
↪data_country2['Deaths_fit'], '-', color='g')
    ax1.set_yscale('log')

    ax2.plot(data_country2['DateTime']+datetime.timedelta(days=delay),␣
↪data_country2['Confirmed'], 's', color='k', label=country2)
    ax2.plot(data_country2['DateTime']+datetime.timedelta(days=delay),␣
↪data_country2['Confirmed_fit'], '-', color='k')
    ax2.plot(data_country1['DateTime'], data_country1['Confirmed'], 'o',␣
↪color='#3498db', label=country1)
    ax2.plot(data_country1['DateTime'], data_country1['Confirmed_fit'], '-',␣
↪color='#3498db')

    ax3.plot(data_country2['DateTime']+datetime.timedelta(days=delay),␣
↪data_country2['Deaths'], 's', color='g', label=country2)
    ax3.plot(data_country2['DateTime']+datetime.timedelta(days=delay),␣
↪data_country2['Deaths_fit'], '-', color='g')
    ax3.plot(data_country1['DateTime'], data_country1['Deaths'], 'o',␣
↪color='tomato', label=country1)
    ax3.plot(data_country1['DateTime'], data_country1['Deaths_fit'], '-',␣
↪color='tomato')

    ax1.legend()
    ax1.set_ylim(0.5,)

    ax1.xaxis.set_tick_params(rotation=45)
    ax2.xaxis.set_ticklabels([])
    ax3.xaxis.set_tick_params(rotation=45)

    fig.suptitle(f'{country1} vs {country2}\nTime delay added: {delay} days',␣
↪fontsize=16)
    fig.savefig(f'./plots/{country1}{province1}_{country2}{province2}.png',␣
↪bbox_inches='tight', dpi=150)
```

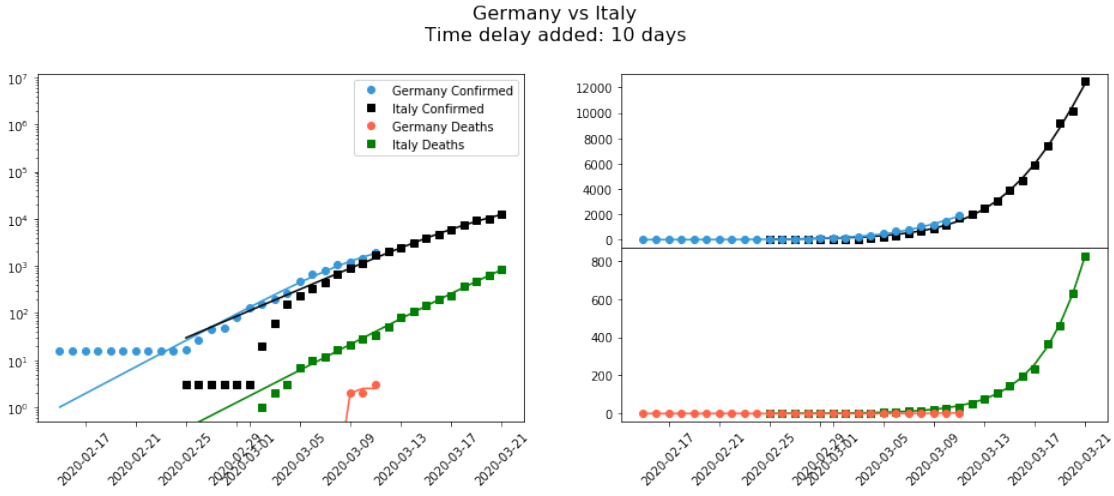
[56]:
```
compare_countries(data, country2='Italy', delay=10, show_numbers=False)
```

```
Spain
Italy
```

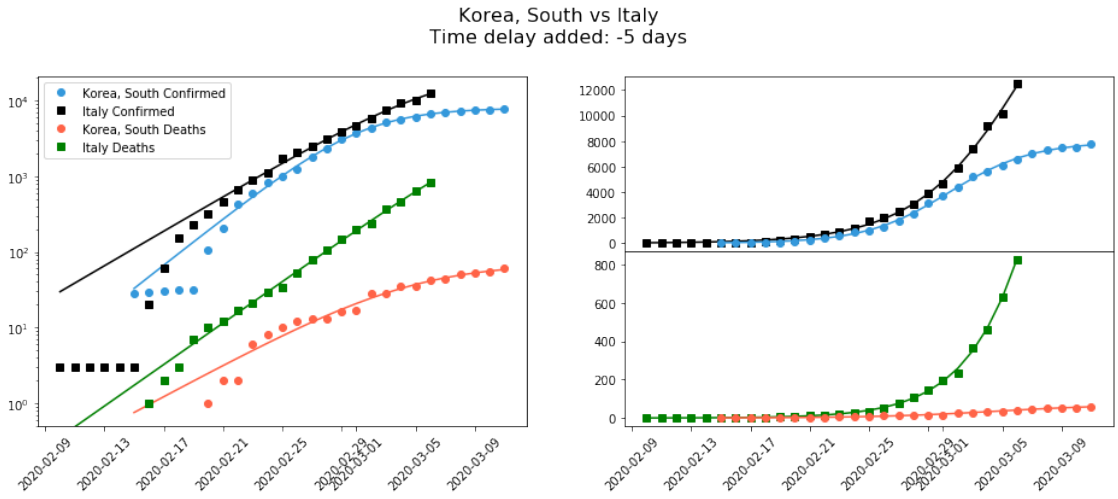


[57]:
```
compare_countries(data, country1='Germany', country2='Italy', delay=10)
```

```
Germany
Italy
```



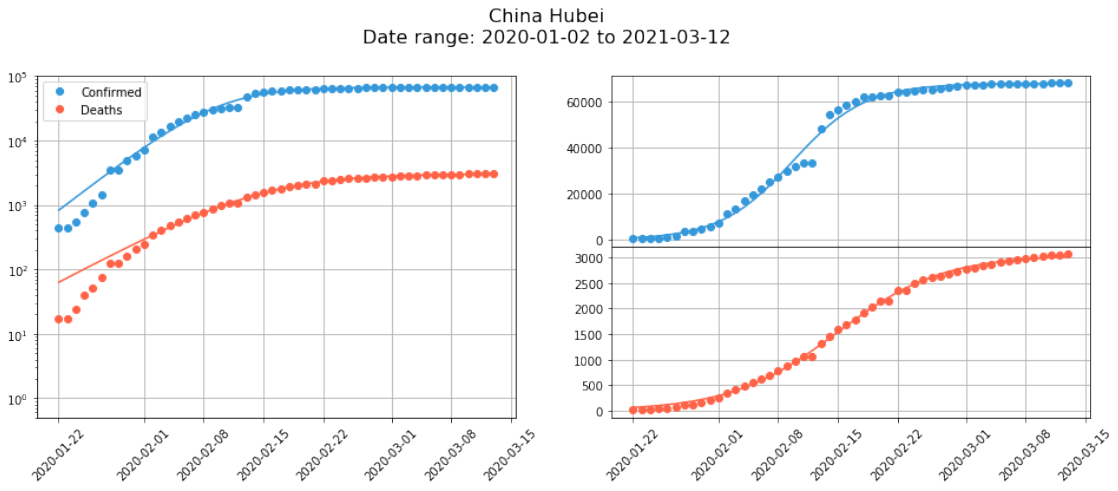
[20]:
```
compare_countries(data, country1='Korea, South', country2='Italy', delay=-5)
```



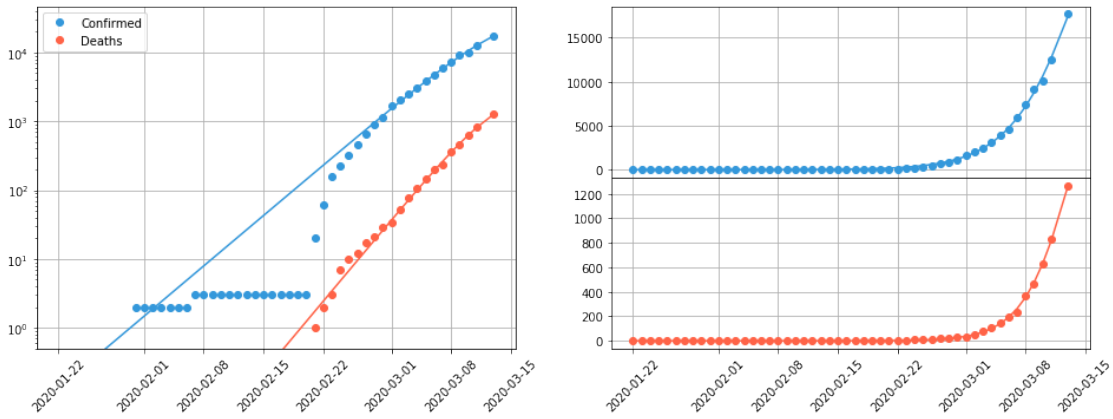
3 Other countries

Show plots for the most affected country/province

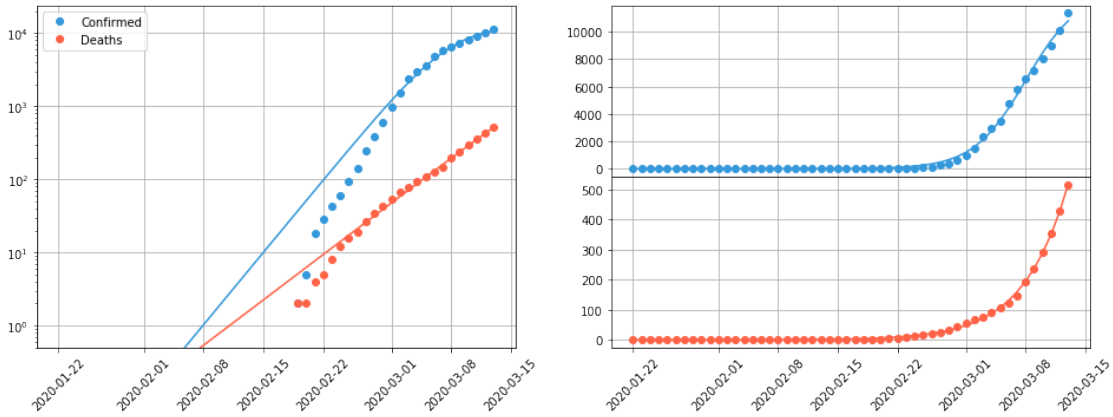
[39]:
```
idx = data.groupby(['Country/Region', 'Province/
↪State'])[['Confirmed','Deaths']].sum().sort_values(by='Deaths')[::-1][0:20].
↪index
for i in idx:
    country, province = i
    plot_country(data, country=country, province=province,␣
↪start_date='2020-01-02', show_numbers=False)
```



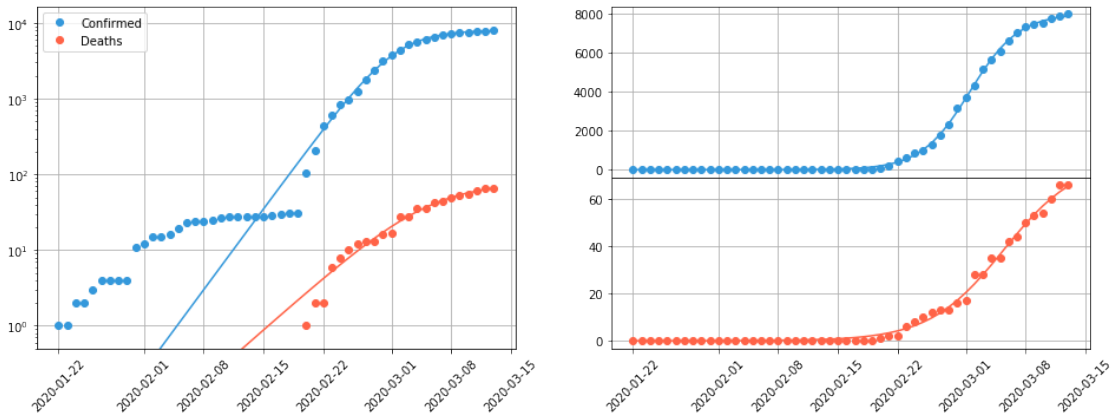
Italy
Date range: 2020-01-02 to 2021-03-12



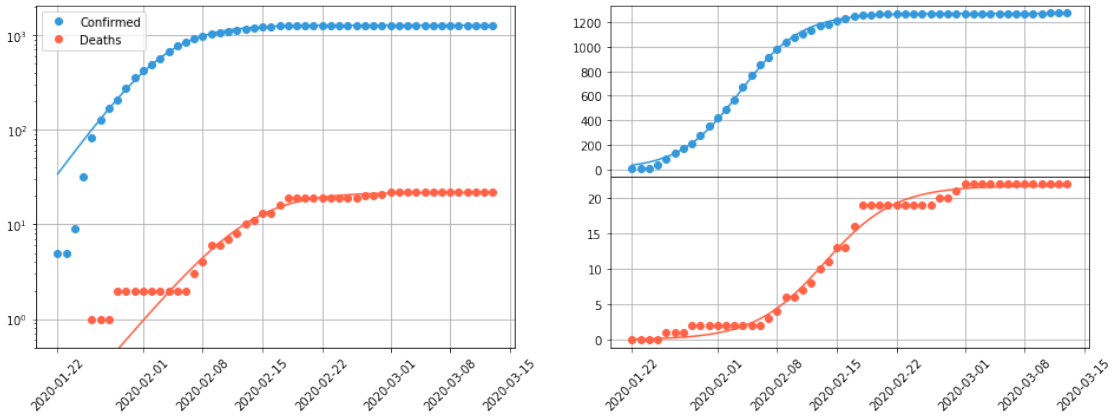
Iran
Date range: 2020-01-02 to 2021-03-12



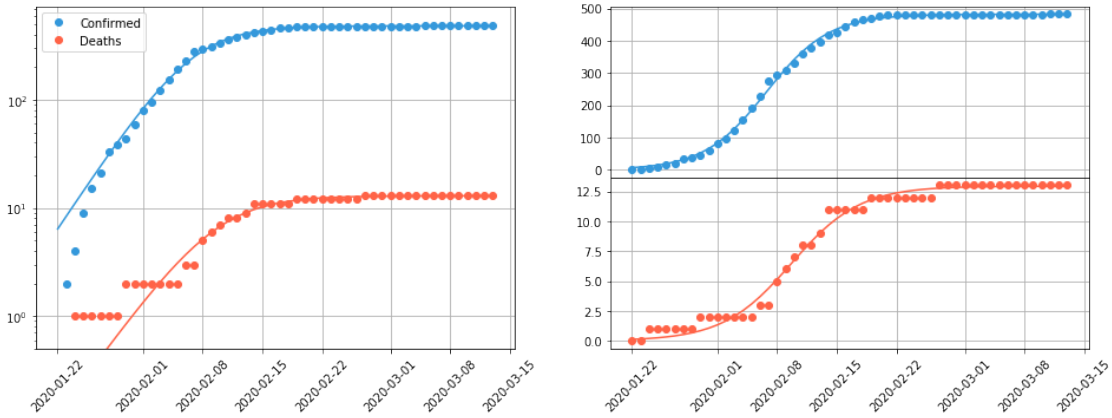
Korea, South
Date range: 2020-01-02 to 2021-03-12



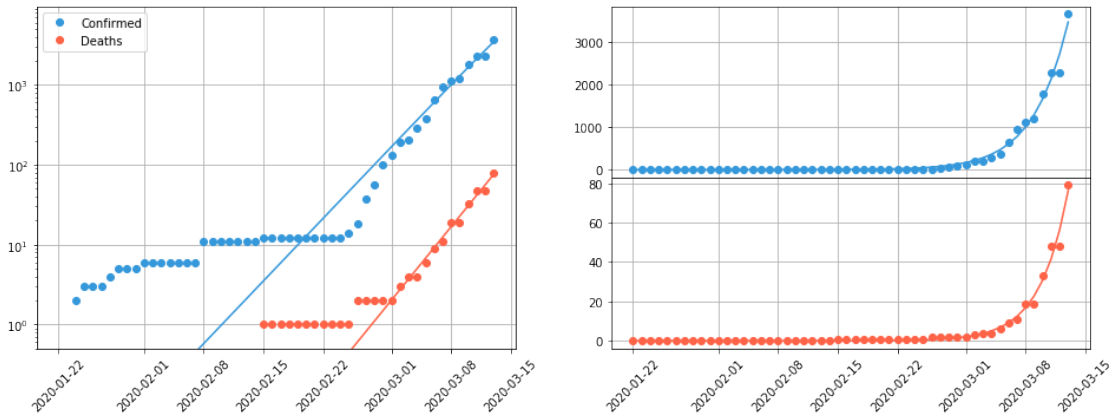
China Henan
Date range: 2020-01-02 to 2021-03-12



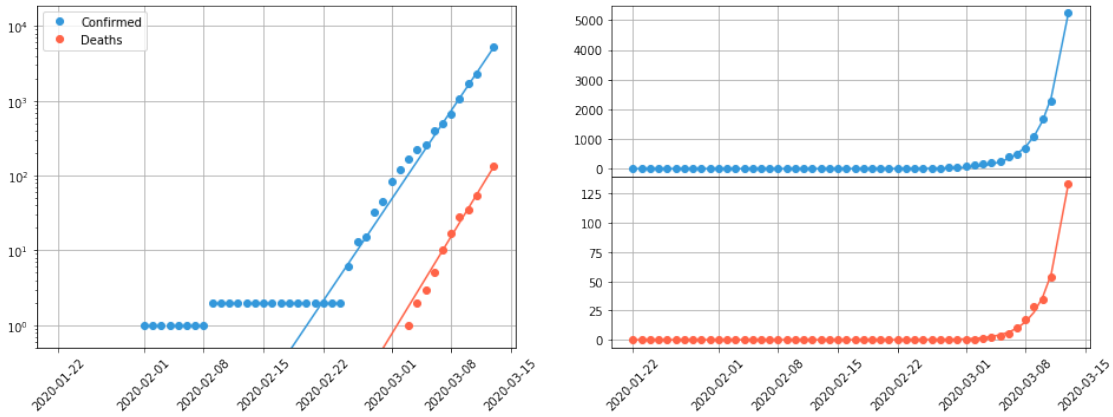
China Heilongjiang
Date range: 2020-01-02 to 2021-03-12



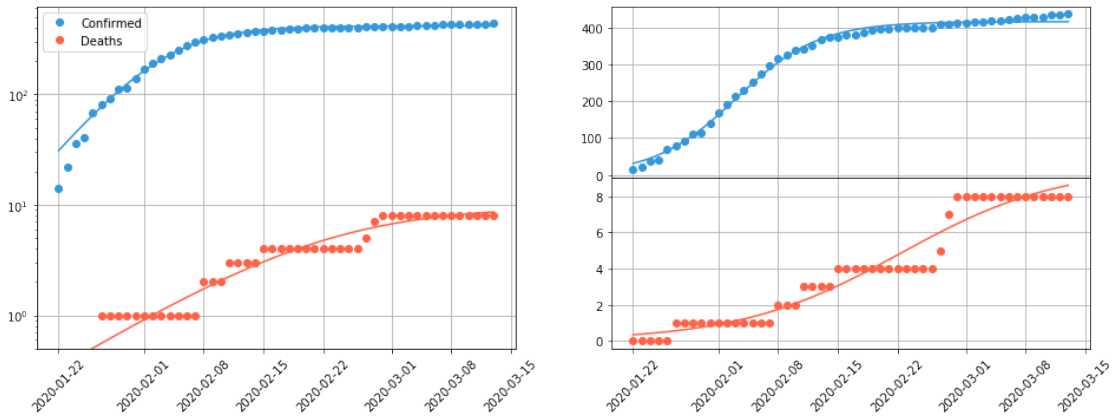
France France
Date range: 2020-01-02 to 2021-03-12



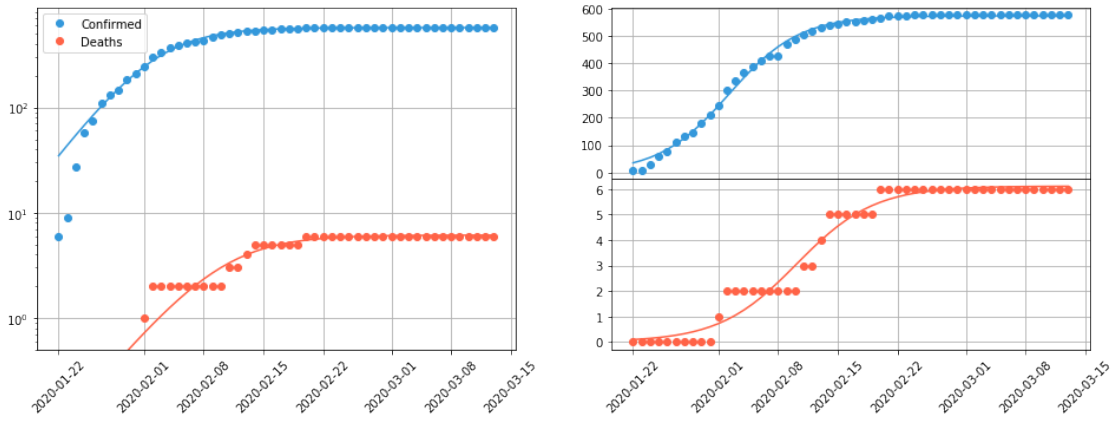
Spain
Date range: 2020-01-02 to 2021-03-12



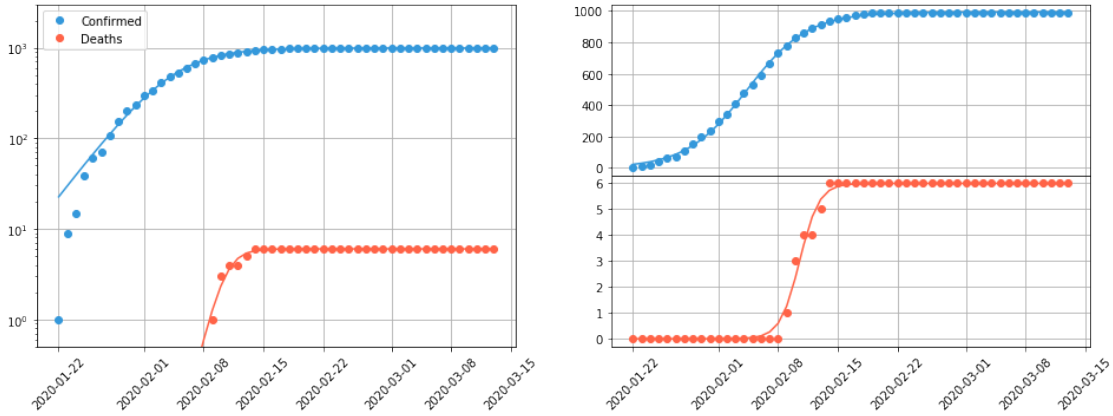
China Beijing
Date range: 2020-01-02 to 2021-03-12



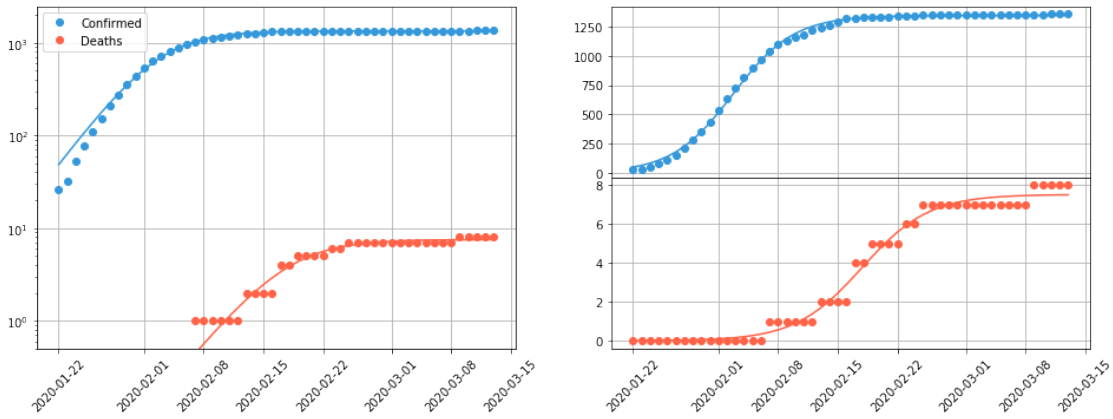
China Chongqing
Date range: 2020-01-02 to 2021-03-12



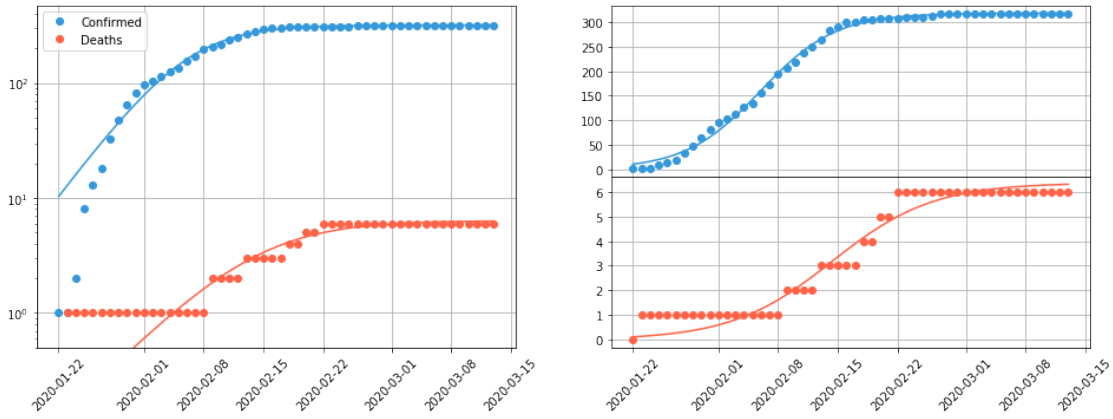
China Anhui
Date range: 2020-01-02 to 2021-03-12



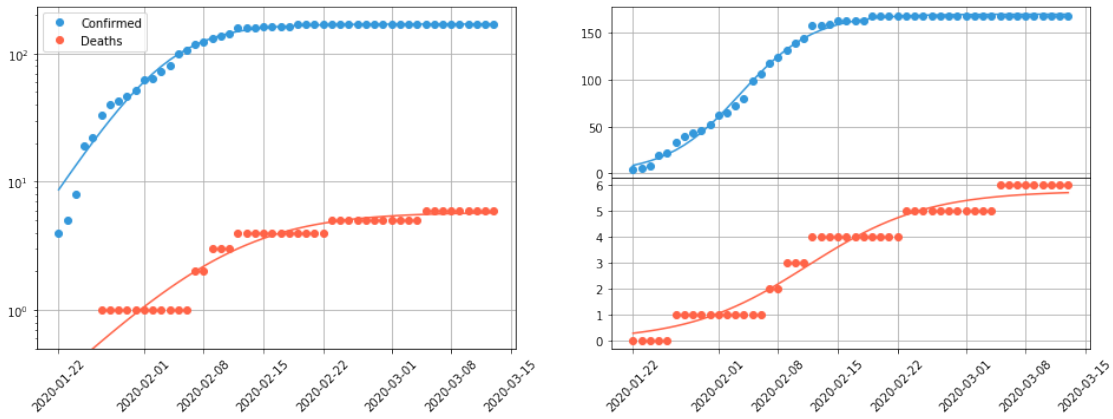
China Guangdong
Date range: 2020-01-02 to 2021-03-12



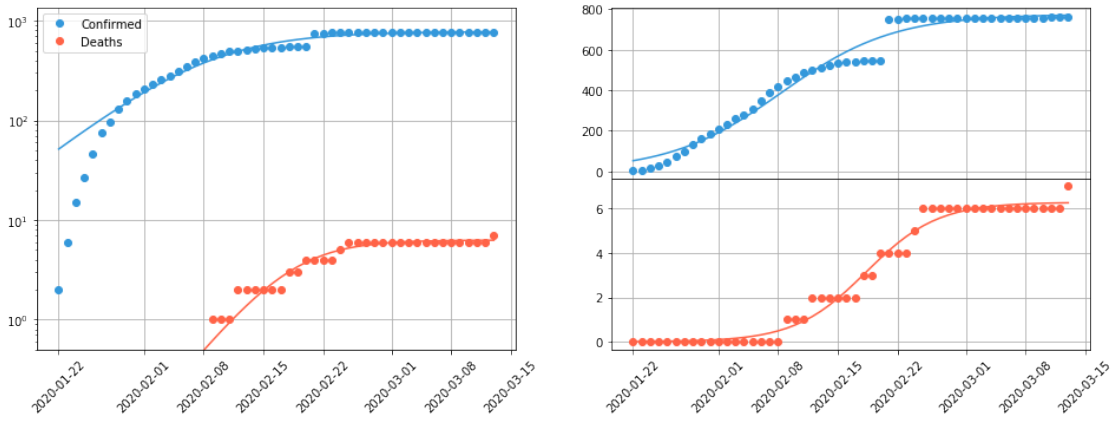
China Hebei
Date range: 2020-01-02 to 2021-03-12



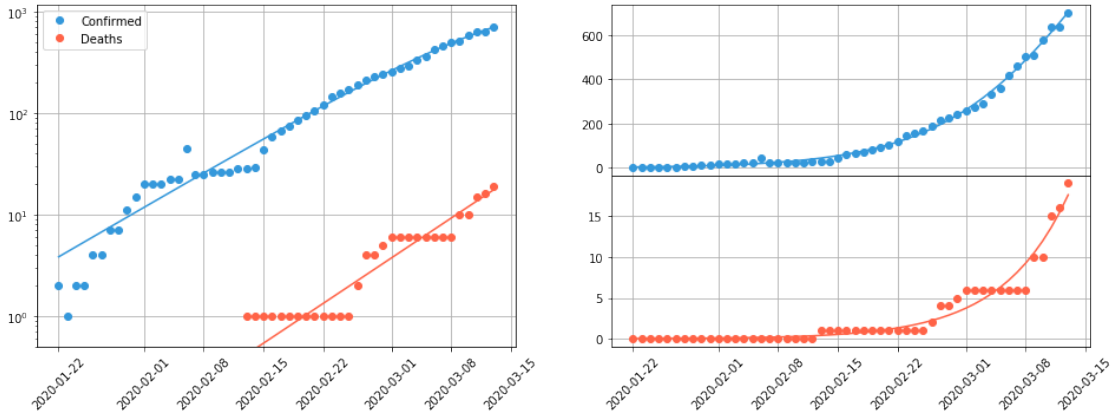
China Hainan
Date range: 2020-01-02 to 2021-03-12



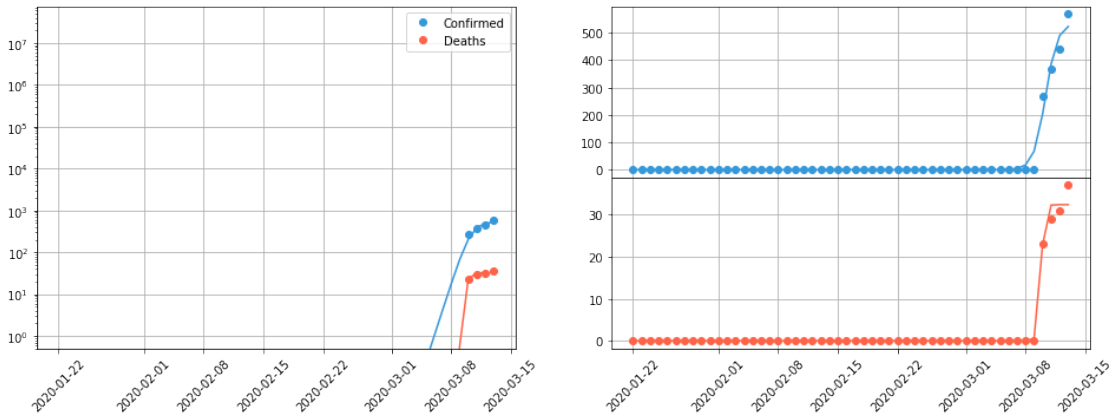
China Shandong
Date range: 2020-01-02 to 2021-03-12



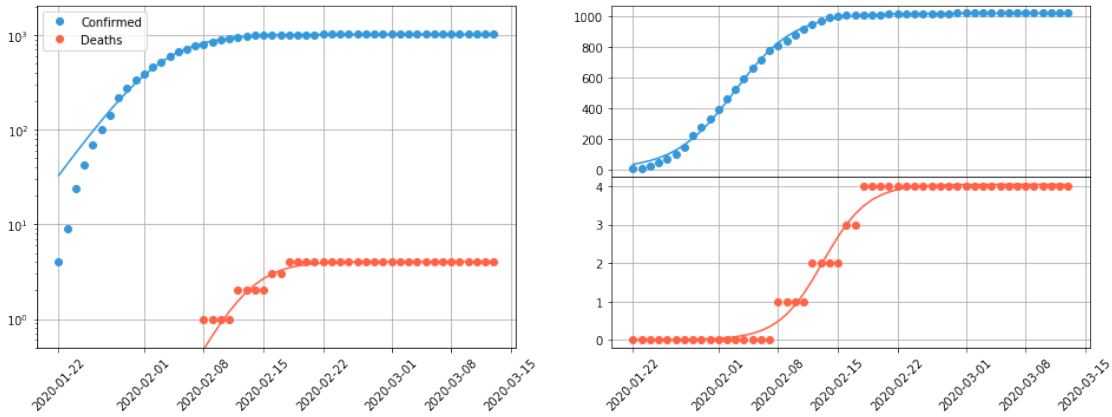
Japan
Date range: 2020-01-02 to 2021-03-12



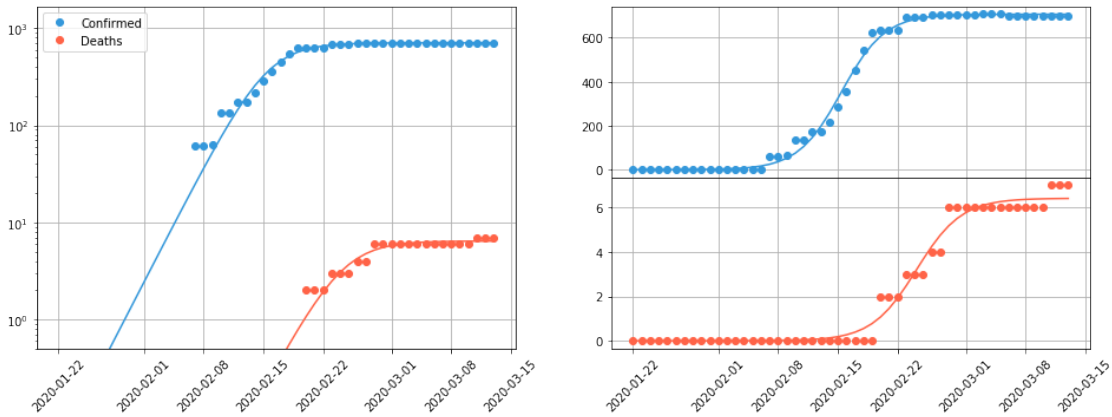
US Washington
Date range: 2020-01-02 to 2021-03-12

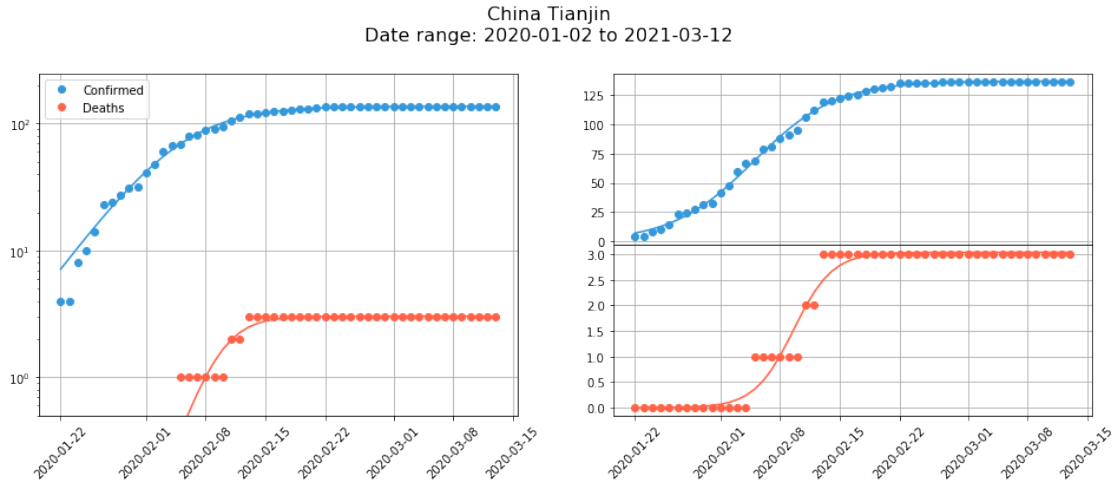


China Hunan
Date range: 2020-01-02 to 2021-03-12



Cruise Ship Diamond Princess
Date range: 2020-01-02 to 2021-03-12




China Tianjin
Date range: 2020-01-02 to 2021-03-12
Confirmed
Deaths
10^2
10^1
10^0
125
100
75
50
25
0
3.0
2.5
2.0
1.5
1.0
0.5
0.0
2020-01-22
2020-02-01
2020-02-08
2020-02-15
2020-02-22
2020-03-01
2020-03-08
2020-03-15

[ ]: