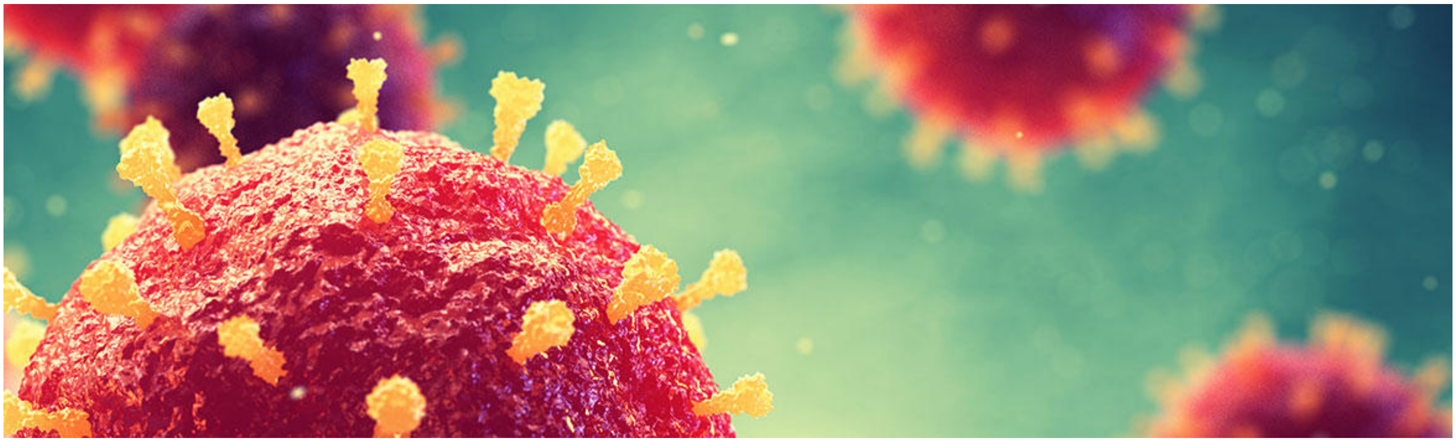


PART 2

Analyzing the epidemiological outbreak of COVID-19

A visual exploratory data analysis approach.

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.ticker as ticker
import seaborn as sns
import plotly.express as px
import theme

%matplotlib inline
```

Data loading and wrangling

We will load COVID-19 data from the GitHub data repository (https://github.com/CSSEGISandData/COVID-19) for the 2019 Novel Coronavirus Visual Dashboard operated by the Johns Hopkins University Center for Systems Science and Engineering (JHU CSSE). Also, Supported by ESRI Living Atlas Team and the Johns Hopkins University Applied Physics Lab (JHU APL).

As we already known the data, we'll go faster now:

In [2]:

```
COVID_CONFIRMED_URL = 'https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_time_series/time_series_19-covid-Confirmed.csv'
covid_confirmed = pd.read_csv(COVID_CONFIRMED_URL)

COVID_DEATHS_URL = 'https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_time_series/time_series_19-covid-Deaths.csv'
covid_deaths = pd.read_csv(COVID_DEATHS_URL)

COVID_RECOVERED_URL = 'https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_time_series/time_series_19-covid-Recovered.csv'
covid_recovered = pd.read_csv(COVID_RECOVERED_URL)
```

In [3]:

```
print(covid_confirmed.shape)
print(covid_deaths.shape)
print(covid_recovered.shape)
```

```
(463, 59)
(463, 59)
(463, 59)
```

In [4]:

```
covid_confirmed.head()
```

Out[4]:

	Province/State	Country/Region	Lat	Long	1/22/20	1/23/20	1/24/20	1/25/20	1/26/20
0	NaN	Thailand	15.000	101.000	2	3	5	7	8
1	NaN	Japan	36.000	138.000	2	1	2	2	4
2	NaN	Singapore	1.283	103.833	0	1	3	3	4
3	NaN	Nepal	28.167	84.250	0	0	0	1	1
4	NaN	Malaysia	2.500	112.500	0	0	0	3	4

First convert all the data to long format:

In [5]:

```
covid_confirmed_long = pd.melt(covid_confirmed,
                               id_vars=covid_confirmed.iloc[:, :4],
                               var_name='date',
                               value_name='confirmed')

covid_deaths_long = pd.melt(covid_deaths,
                               id_vars=covid_deaths.iloc[:, :4],
                               var_name='date',
                               value_name='deaths')

covid_recovered_long = pd.melt(covid_recovered,
                               id_vars=covid_recovered.iloc[:, :4],
                               var_name='date',
                               value_name='recovered')
```

In [6]:

```
covid_confirmed_long.shape
```

Out[6]:

```
(25465, 6)
```

In [7]:

```
covid_confirmed_long.head()
```

Out[7]:

	Province/State	Country/Region	Lat	Long	date	confirmed
0	NaN	Thailand	15.000	101.000	1/22/20	2
1	NaN	Japan	36.000	138.000	1/22/20	2
2	NaN	Singapore	1.283	103.833	1/22/20	0
3	NaN	Nepal	28.167	84.250	1/22/20	0
4	NaN	Malaysia	2.500	112.500	1/22/20	0

Why having three separated `DataFrame` s? Let's merge them.

> You can learn these advance Pandas topics in detail on our ***Data Wrangling*** course (https://rmotr.com/data-cleaning-with-pandas/)!

In [8]:

```
covid_df = covid_confirmed_long
covid_df['deaths'] = covid_deaths_long['deaths']
covid_df['recovered'] = covid_recovered_long['recovered']
```

In [9]:

```
print(covid_df.shape)

covid_df.head()
```

```
(25465, 8)
```

Out[9]:

	Province/State	Country/Region	Lat	Long	date	confirmed	deaths	recovered
0	NaN	Thailand	15.000	101.000	1/22/20	2	0	0
1	NaN	Japan	36.000	138.000	1/22/20	2	0	0
2	NaN	Singapore	1.283	103.833	1/22/20	0	0	0
3	NaN	Nepal	28.167	84.250	1/22/20	0	0	0
4	NaN	Malaysia	2.500	112.500	1/22/20	0	0	0

Add `active` column

Calculate a new `active` cases value with the following formula:
$$ Active = Confirmed - Deaths - Recovered $$

In [10]:

```
covid_df['active'] = covid_df['confirmed'] - covid_df['deaths'] - covid_df['reco
vered']
```

In [11]:

```
print(covid_df.shape)

covid_df.head()
```

```
(25465, 9)
```

Out[11]:

	Province/State	Country/Region	Lat	Long	date	confirmed	deaths	recovered	ac
0	NaN	Thailand	15.000	101.000	1/22/20	2	0	0	
1	NaN	Japan	36.000	138.000	1/22/20	2	0	0	
2	NaN	Singapore	1.283	103.833	1/22/20	0	0	0	
3	NaN	Nepal	28.167	84.250	1/22/20	0	0	0	
4	NaN	Malaysia	2.500	112.500	1/22/20	0	0	0	

Data cleaning

As we did before replace `Mainland china` with just `China` , and fill some missing values.

In [12]:

```
covid_df['Country/Region'].replace('Mainland China', 'China', inplace=True)
```

In [13]:

```
covid_df[['Province/State']] = covid_df[['Province/State']].fillna('')
```

In [14]:

```
covid_df.fillna(0, inplace=True)
```

Final checks:

In [15]:

```
covid_df.isna().sum().sum()
```

Out[15]:

```
0
```

Save `DataFrame` to CSV file

Now persist our `DataFrame` to disk using `to_csv()` pandas method.

In [16]:

```
covid_df.to_csv('covid_df.csv', index=None)
```

Load it again and check if everything is ok:

In [17]:

```
pd.read_csv('covid_df.csv')
```

```
Out[17]:
```

	Province/State	Country/Region	Lat	Long	date	confirmed	deaths	recovered
0	NaN	Thailand	15.000	101.000	1/22/20	2	0	0
1	NaN	Japan	36.000	138.000	1/22/20	2	0	0
2	NaN	Singapore	1.283	103.833	1/22/20	0	0	0
3	NaN	Nepal	28.167	84.250	1/22/20	0	0	0
4	NaN	Malaysia	2.500	112.500	1/22/20	0	0	0
...	...	...	...	...	...	...	...	...
25460	NaN	Somalia	5.152	46.200	3/16/20	1	0	0
25461	NaN	Tanzania	-6.369	34.889	3/16/20	1	0	0
25462	NaN	The Bahamas	24.250	-76.000	3/16/20	1	0	0
25463	Virgin Islands	US	18.336	-64.896	3/16/20	1	0	0
25464	Cayman Islands	United Kingdom	19.313	-81.255	3/16/20	1	1	0

25465 rows × 9 columns

Country analysis

Now we'll analyze COVID-19 cases for each country.

In [18]:

```
covid_df.head()
```

Out[18]:

	Province/State	Country/Region	Lat	Long	date	confirmed	deaths	recovered	ac
0		Thailand	15.000	101.000	1/22/20	2	0	0	
1		Japan	36.000	138.000	1/22/20	2	0	0	
2		Singapore	1.283	103.833	1/22/20	0	0	0	
3		Nepal	28.167	84.250	1/22/20	0	0	0	
4		Malaysia	2.500	112.500	1/22/20	0	0	0	

Now it's time to to aggregate the data by `Country/Region` and `Province/State` before continue.

First, group the data by `Country/Region` and `Province/State` at the same time, so we can get the `max()` value for each `Province/State` over the time.

In [19]:

```
covid_countries_df = covid_df.groupby(['Country/Region', 'Province/State']).max(
).reset_index()

covid_countries_df
```

Out[19]:

	Country/Region	Province/State	Lat	Long	date	confirmed	deaths	recovered	a
0	Afghanistan		33.000	65.000	3/9/20	21	0	1	
1	Albania		41.153	20.168	3/9/20	51	1	0	
2	Algeria		28.034	1.660	3/9/20	54	4	12	
3	Andorra		42.506	1.522	3/9/20	2	0	1	
4	Antigua and Barbuda		17.061	-61.796	3/9/20	1	0	0	
...	...	...	...	...	...	...	...	...	
458	Uruguay		-32.523	-55.766	3/9/20	8	0	0	
459	Uzbekistan		41.377	64.585	3/9/20	6	0	0	
460	Venezuela		6.424	-66.590	3/9/20	17	0	0	
461	Vietnam		16.000	108.000	3/9/20	61	0	16	
462	occupied Palestinian territory		31.952	35.233	3/9/20	0	0	0	

463 rows × 9 columns

Finally, group the data again by `Country/Region` and this time get the `sum()` of the cases of every `Province/State` over the country.

In [20]:

```
covid_countries_df = covid_countries_df.groupby('Country/Region').sum().reset_in
dex()

covid_countries_df
```

Out[20]:

	Country/Region	Lat	Long	confirmed	deaths	recovered	active
0	Afghanistan	33.000	65.000	21	0	1	20
1	Albania	41.153	20.168	51	1	0	50
2	Algeria	28.034	1.660	54	4	12	38
3	Andorra	42.506	1.522	2	0	1	1
4	Antigua and Barbuda	17.061	-61.796	1	0	0	1
...	...	...	...	...	...	...	...
151	Uruguay	-32.523	-55.766	8	0	0	8
152	Uzbekistan	41.377	64.585	6	0	0	6
153	Venezuela	6.424	-66.590	17	0	0	17
154	Vietnam	16.000	108.000	61	0	16	45
155	occupied Palestinian territory	31.952	35.233	0	0	0	0

156 rows × 7 columns

Remove unused `Lat` and `Long` columns:

In [21]:

```
covid_countries_df.drop(['Lat', 'Long'], axis=1, inplace=True)

covid_countries_df
```

Out[21]:

	Country/Region	confirmed	deaths	recovered	active
0	Afghanistan	21	0	1	20
1	Albania	51	1	0	50
2	Algeria	54	4	12	38
3	Andorra	2	0	1	1
4	Antigua and Barbuda	1	0	0	1
...	...	...	...	...	...
151	Uruguay	8	0	0	8
152	Uzbekistan	6	0	0	6
153	Venezuela	17	0	0	17
154	Vietnam	61	0	16	45
155	occupied Palestinian territory	0	0	0	0

156 rows × 5 columns

Done, we can now start getting insights from our `covid_countries_df` data.

Which are the top-10 countries with more `confirmed` cases?

In [22]:

```
top_10_confirmed = covid_countries_df.sort_values(by='confirmed', ascending=False).head(10)

top_10_confirmed
```

Out[22]:

	Country/Region	confirmed	deaths	recovered	active
28	China	81033	3217	67910	59961
72	Italy	27980	2158	2749	23073
68	Iran	14991	853	4590	9548
133	Spain	9942	342	530	9070
79	Korea, South	8236	75	1137	7577
52	Germany	7272	17	67	7188
48	France	6652	148	12	6492
147	US	5150	107	24	5023
138	Switzerland	2200	14	4	2182
150	United Kingdom	1551	56	21	1476

In [23]:

```
fig = px.bar(top_10_confirmed.sort_values(by='confirmed', ascending=True),
             x="confirmed", y="Country/Region",
             title='Confirmed Cases', text='confirmed',
             template='plotly_dark', orientation='h')

fig.update_traces(marker_color='#3498db', textposition='outside')

fig.show()
```

Which are the top-10 countries with more `recovered` cases?

In [24]:

```
top_10_recovered = covid_countries_df.sort_values(by='recovered', ascending=False).head(10)

top_10_recovered
```

Out[24]:

	Country/Region	confirmed	deaths	recovered	active
28	China	81033	3217	67910	59961
68	Iran	14991	853	4590	9548
72	Italy	27980	2158	2749	23073
79	Korea, South	8236	75	1137	7577
133	Spain	9942	342	530	9070
35	Cruise Ship	706	7	325	691
74	Japan	839	27	144	699
128	Singapore	243	0	109	134
11	Bahrain	214	1	77	166
52	Germany	7272	17	67	7188

In [25]:

```
fig = px.bar(top_10_recovered.sort_values(by='recovered', ascending=True),
             x="recovered", y="Country/Region",
             title='Recovered Cases', text='recovered',
             template='plotly_dark', orientation='h')

fig.update_traces(marker_color='#2ecc71', textposition='outside')

fig.show()
```

Which are the top-10 countries with more `death` cases?

In [26]:

```
top_10_deaths = covid_countries_df.sort_values(by='deaths', ascending=False).hea
d(10)

top_10_deaths
```

Out[26]:

	Country/Region	confirmed	deaths	recovered	active
28	China	81033	3217	67910	59961
72	Italy	27980	2158	2749	23073
68	Iran	14991	853	4590	9548
133	Spain	9942	342	530	9070
48	France	6652	148	12	6492
147	US	5150	107	24	5023
79	Korea, South	8236	75	1137	7577
150	United Kingdom	1551	56	21	1476
74	Japan	839	27	144	699
101	Netherlands	1414	24	2	1388

In [27]:

```
fig = px.bar(top_10_confirmed.sort_values(by='deaths', ascending=True),
             x="deaths", y="Country/Region",
             title='Death Cases', text='deaths',
             template='plotly_dark', orientation='h')

fig.update_traces(marker_color='#e74c3c', textposition='outside')

fig.show()
```

Which are the top-20 countries with higher `mortality rate`?

Analyze `mortality rate` just if the country has at least 100 confirmed cases.

In [28]:

```
covid_countries_df['mortality_rate'] = round(covid_countries_df['deaths'] / covid_countries_df['confirmed'] * 100, 2)

temp = covid_countries_df[covid_countries_df['confirmed'] > 100]

top_20_mortality_rate = temp.sort_values(by='mortality_rate', ascending=False).head(20)

top_20_mortality_rate
```

Out[28]:

	Country/Region	confirmed	deaths	recovered	active	mortality_rate
111	Philippines	142	12	2	128	8.450
69	Iraq	124	10	26	88	8.060
72	Italy	27980	2158	2749	23073	7.710
123	San Marino	109	7	4	98	6.420
68	Iran	14991	853	4590	9548	5.690
28	China	81033	3217	67910	59961	3.970
67	Indonesia	134	5	8	121	3.730
150	United Kingdom	1551	56	21	1476	3.610
133	Spain	9942	342	530	9070	3.440
74	Japan	839	27	144	699	3.220
65	Iceland	180	5	8	180	2.780
83	Lebanon	110	3	1	106	2.730
112	Poland	177	4	13	160	2.260
48	France	6652	148	12	6492	2.220
147	US	5150	107	24	5023	2.080
101	Netherlands	1414	24	2	1388	1.700
66	India	119	2	13	104	1.680
42	Egypt	150	2	27	121	1.330
54	Greece	331	4	8	319	1.210
70	Ireland	169	2	0	167	1.180

In [29]:

```
fig = px.bar(top_20_mortality_rate.sort_values(by='mortality_rate', ascending=True),
             x="mortality_rate", y="Country/Region",
             title='Mortality rate', text='mortality_rate',
             template='plotly_dark', orientation='h',
             width=700, height=600)

fig.update_traces(marker_color='#c0392b', textposition='outside')

fig.show()
```

Country analysis over the time

Another useful graphic could be exploring confirmed cases per country over the time.

Let's aggregate values, grouping by `Country/Region` and `date` .

> Hint! the `sort=False` parameter will keep our dates ordered.

In [30]:

```
covid_countries_date_df = covid_df.groupby(['Country/Region', 'date'], sort=False).sum().reset_index()
```

In [31]:

```
covid_countries_date_df
```

Out[31]:

	Country/Region	date	Lat	Long	confirmed	deaths	recovered	active
0	Thailand	1/22/20	15.000	101.000	2	0	0	2
1	Japan	1/22/20	36.000	138.000	2	0	0	2
2	Singapore	1/22/20	1.283	103.833	0	0	0	0
3	Nepal	1/22/20	28.167	84.250	0	0	0	0
4	Malaysia	1/22/20	2.500	112.500	0	0	0	0
...	...	...	...	...	...	...	...	...
8575	Mayotte	3/16/20	-12.828	45.166	1	0	0	1
8576	Republic of the Congo	3/16/20	-1.440	15.556	1	0	0	1
8577	Somalia	3/16/20	5.152	46.200	1	0	0	1
8578	Tanzania	3/16/20	-6.369	34.889	1	0	0	1
8579	The Bahamas	3/16/20	24.250	-76.000	1	0	0	1

8580 rows × 8 columns

Now just filter the data from countries you want to analyze:

In [32]:

```
covid_US = covid_countries_date_df[covid_countries_date_df['Country/Region'] ==
'US']

covid_US
```

Out[32]:

	Country/Region	date	Lat	Long	confirmed	deaths	recovered	active
88	US	1/22/20	9,531.003	-22,951.821	1	0	0	1
244	US	1/23/20	9,531.003	-22,951.821	1	0	0	1
400	US	1/24/20	9,531.003	-22,951.821	2	0	0	2
556	US	1/25/20	9,531.003	-22,951.821	2	0	0	2
712	US	1/26/20	9,531.003	-22,951.821	5	0	0	5
868	US	1/27/20	9,531.003	-22,951.821	5	0	0	5
1024	US	1/28/20	9,531.003	-22,951.821	5	0	0	5
1180	US	1/29/20	9,531.003	-22,951.821	5	0	0	5
1336	US	1/30/20	9,531.003	-22,951.821	5	0	0	5
1492	US	1/31/20	9,531.003	-22,951.821	7	0	0	7
1648	US	2/1/20	9,531.003	-22,951.821	8	0	0	8
1804	US	2/2/20	9,531.003	-22,951.821	8	0	0	8
1960	US	2/3/20	9,531.003	-22,951.821	11	0	0	11
2116	US	2/4/20	9,531.003	-22,951.821	11	0	0	11
2272	US	2/5/20	9,531.003	-22,951.821	11	0	0	11
2428	US	2/6/20	9,531.003	-22,951.821	11	0	0	11
2584	US	2/7/20	9,531.003	-22,951.821	11	0	0	11
2740	US	2/8/20	9,531.003	-22,951.821	11	0	0	11
2896	US	2/9/20	9,531.003	-22,951.821	11	0	3	8
3052	US	2/10/20	9,531.003	-22,951.821	11	0	3	8
3208	US	2/11/20	9,531.003	-22,951.821	12	0	3	9
3364	US	2/12/20	9,531.003	-22,951.821	12	0	3	9
3520	US	2/13/20	9,531.003	-22,951.821	13	0	3	10
3676	US	2/14/20	9,531.003	-22,951.821	13	0	3	10

3832	US	2/15/20	9,531.003	-22,951.821	13	0	3	10
3988	US	2/16/20	9,531.003	-22,951.821	13	0	3	10
4144	US	2/17/20	9,531.003	-22,951.821	13	0	3	10
4300	US	2/18/20	9,531.003	-22,951.821	13	0	3	10
4456	US	2/19/20	9,531.003	-22,951.821	13	0	3	10
4612	US	2/20/20	9,531.003	-22,951.821	13	0	3	10
4768	US	2/21/20	9,531.003	-22,951.821	15	0	5	10
4924	US	2/22/20	9,531.003	-22,951.821	15	0	5	10
5080	US	2/23/20	9,531.003	-22,951.821	15	0	5	10
5236	US	2/24/20	9,531.003	-22,951.821	51	0	5	46
5392	US	2/25/20	9,531.003	-22,951.821	51	0	6	45
5548	US	2/26/20	9,531.003	-22,951.821	57	0	6	51
5704	US	2/27/20	9,531.003	-22,951.821	58	0	6	52
5860	US	2/28/20	9,531.003	-22,951.821	60	0	7	53
6016	US	2/29/20	9,531.003	-22,951.821	68	1	7	60
6172	US	3/1/20	9,531.003	-22,951.821	74	1	7	66
6328	US	3/2/20	9,531.003	-22,951.821	98	6	7	85
6484	US	3/3/20	9,531.003	-22,951.821	118	7	7	104
6640	US	3/4/20	9,531.003	-22,951.821	149	11	7	131
6796	US	3/5/20	9,531.003	-22,951.821	217	12	7	198
6952	US	3/6/20	9,531.003	-22,951.821	262	14	7	241
7108	US	3/7/20	9,531.003	-22,951.821	402	17	7	378
7264	US	3/8/20	9,531.003	-22,951.821	518	21	7	490
7420	US	3/9/20	9,531.003	-22,951.821	583	22	7	554
7576	US	3/10/20	9,531.003	-22,951.821	959	28	8	923
7732	US	3/11/20	9,531.003	-22,951.821	1281	36	8	1237
7888	US	3/12/20	9,531.003	-22,951.821	1663	40	12	1611
8044	US	3/13/20	9,531.003	-22,951.821	2179	47	12	2120
8200	US	3/14/20	9,531.003	-22,951.821	2727	54	12	2661
8356	US	3/15/20	9,531.003	-22,951.821	3499	63	12	3424
8512	US	3/16/20	9,531.003	-22,951.821	4632	85	17	4530

In [33]:

```
covid_China = covid_countries_date_df[covid_countries_date_df['Country/Region'] == 'China']
covid_Italy = covid_countries_date_df[covid_countries_date_df['Country/Region'] == 'Italy']
covid_Germany = covid_countries_date_df[covid_countries_date_df['Country/Region'] == 'Germany']
covid_Spain = covid_countries_date_df[covid_countries_date_df['Country/Region'] == 'Spain']
covid_Argentina = covid_countries_date_df[covid_countries_date_df['Country/Region'] == 'Argentina']
```

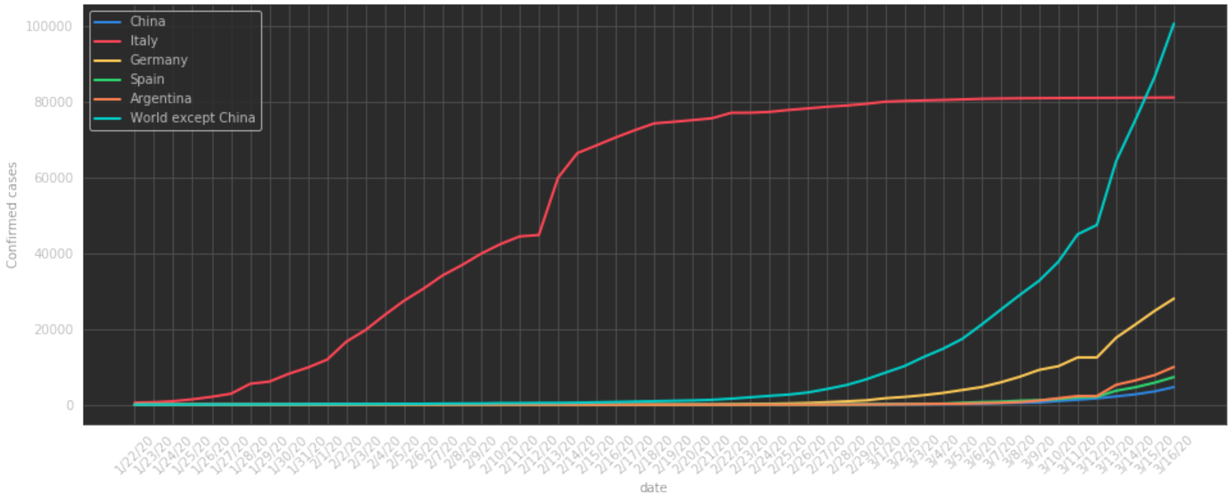
I also add a calculated `World except China` containing all the cases in the world excepting the cases in China.

In [34]:

```
covid_no_China = covid_countries_date_df[covid_countries_date_df['Country/Region'] != 'China']

covid_no_China = covid_no_China.groupby('date', sort=False).sum().reset_index()
```

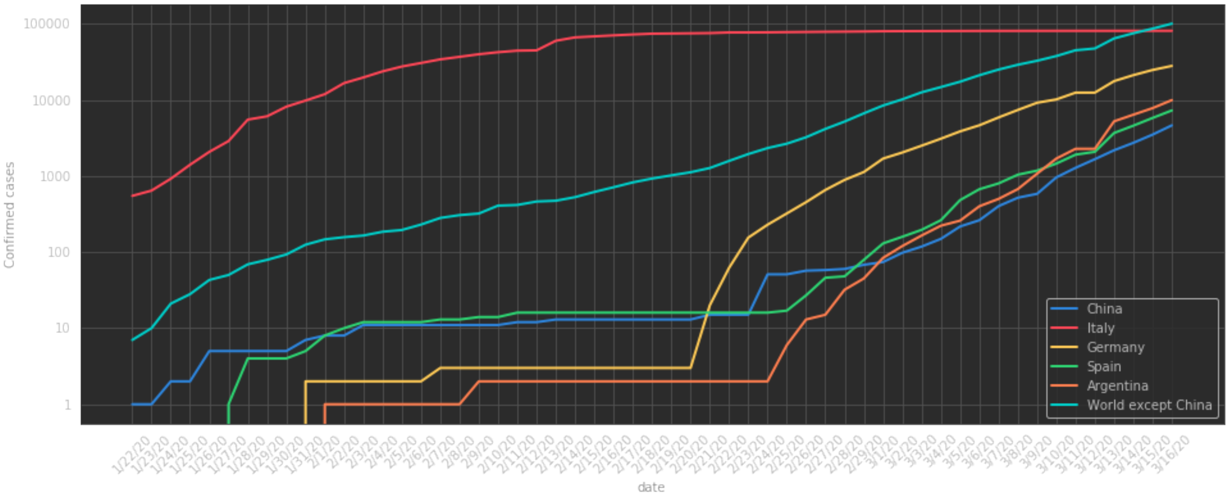
In [35]:

```
fig, ax = plt.subplots(figsize=(16, 6))

sns.lineplot(x=covid_US['date'], y=covid_US['confirmed'], sort=False, linewidth=2)
sns.lineplot(x=covid_China['date'], y=covid_China['confirmed'], sort=False, linewidth=2)
sns.lineplot(x=covid_Italy['date'], y=covid_Italy['confirmed'], sort=False, linewidth=2)
sns.lineplot(x=covid_Germany['date'], y=covid_Germany['confirmed'], sort=False, linewidth=2)
sns.lineplot(x=covid_Spain['date'], y=covid_Spain['confirmed'], sort=False, linewidth=2)
sns.lineplot(x=covid_no_China['date'], y=covid_no_China['confirmed'], sort=False, linewidth=2)

plt.suptitle("COVID-19 per country cases over the time", fontsize=16, fontweight='bold', color='white')

plt.xticks(rotation=45)
plt.ylabel('Confirmed cases')

ax.legend(['China', 'Italy', 'Germany', 'Spain', 'Argentina', 'World except China'])

plt.show()
```



In [36]:

```
fig, ax = plt.subplots(figsize=(16, 6))
ax.set(yscale="log")
ax.yaxis.set_major_formatter(ticker.FuncFormatter(lambda y, _: '{:g}'.format(y))
)

sns.lineplot(x=covid_US['date'], y=covid_US['confirmed'], sort=False, linewidth=
2)
sns.lineplot(x=covid_China['date'], y=covid_China['confirmed'], sort=False, line
width=2)
sns.lineplot(x=covid_Italy['date'], y=covid_Italy['confirmed'], sort=False, line
width=2)
sns.lineplot(x=covid_Germany['date'], y=covid_Germany['confirmed'], sort=False,
linewidth=2)
sns.lineplot(x=covid_Spain['date'], y=covid_Spain['confirmed'], sort=False, line
width=2)
sns.lineplot(x=covid_no_China['date'], y=covid_no_China['confirmed'], sort=False
, linewidth=2)

plt.suptitle("COVID-19 per country cases over the time", fontsize=16, fontweight
='bold', color='white')
plt.title("(logarithmic scale)", color='white')

plt.xticks(rotation=45)
plt.ylabel('Confirmed cases')

ax.legend(['China', 'Italy', 'Germany', 'Spain', 'Argentina', 'World except Chin
a'])

plt.show()
```



Custom country analyzer

Finally we'll create our custom `get_country_covid_info(country, log)` function to analyze countries. This function will receive `country` and `log` parameters to filter desired country and apply -or not- logarithmic scale.

That function will return some cool plots showing the country COVID-19 cases. To make these plots, we'll create another helper functions: `plot_country_global_info()` , `plot_country_cases_over_time()` and `plot_province_cases()` .

In [37]:

```
def plot_country_global_info(country):
    country_info = covid_countries_df[covid_countries_df['Country/Region'] == country]

    country_info_long = country_info.melt(value_vars=['active', 'deaths', 'recovered'],
                                          var_name="status",
                                          value_name="count")

    country_info_long['upper'] = 'Confirmed cases'

    fig = px.treemap(country_info_long, path=["upper", "status"], values="count",
                     title=f"Total COVID-19 confirmed cases in {country}",
                     color_discrete_sequence=['#3498db', '#2ecc71', '#e74c3c'],
                     template='plotly_dark')

    fig.data[0].textinfo = 'label+text+value'

    fig.show()
```

In [38]:

```
def plot_country_cases_over_time(country, log):
    country_date_info = covid_countries_date_df[covid_countries_date_df['Country/Region'] == country]

    fig, ax = plt.subplots(figsize=(16, 6))

    if log:
        ax.set(yscale="log")
        ax.yaxis.set_major_formatter(ticker.FuncFormatter(lambda y, _: '{:g}'.format(y)))
        plt.title("(logarithmic scale)", color='white')

    sns.lineplot(x=country_date_info['date'], y=country_date_info['confirmed'], sort=False, linewidth=2)
    sns.lineplot(x=country_date_info['date'], y=country_date_info['deaths'], sort=False, linewidth=2)
    sns.lineplot(x=country_date_info['date'], y=country_date_info['recovered'], sort=False, linewidth=2)
    sns.lineplot(x=country_date_info['date'], y=country_date_info['active'], sort=False, linewidth=2)

    ax.lines[0].set_linestyle("--")

    plt.suptitle(f"COVID-19 cases in {country} over the time", fontsize=16, fontweight='bold', color='white')

    plt.xticks(rotation=45)
    plt.ylabel('Number of cases')

    ax.legend(['Confirmed', 'Deaths', 'Recovered', 'Active'])

    plt.show()
```

In [39]:

```
def plot_province_cases(country):
    covid_provinces_df = covid_df.groupby(['Province/State', 'Country/Region']).max().reset_index()

    country_provinces_info = covid_provinces_df[covid_provinces_df['Country/Region'] == country]

    has_provinces = country_provinces_info.shape[0] > 1

    if (has_provinces):
        country_info_long = country_provinces_info.melt(id_vars=['Province/State'],
                                                        value_vars=['active', 'deaths', 'recovered'],
                                                        var_name="status",
                                                        value_name="count")

        country_info_long['upper'] = 'Confirmed cases'

        fig = px.treemap(country_info_long, path=['upper', "Province/State", "status"],
                         values="count",
                         title=f"Number of COVID-19 confirmed cases per Province/State in {country}",
                         template='plotly_dark')

        fig.data[0].textinfo = 'label+text+value'

        fig.show()
```

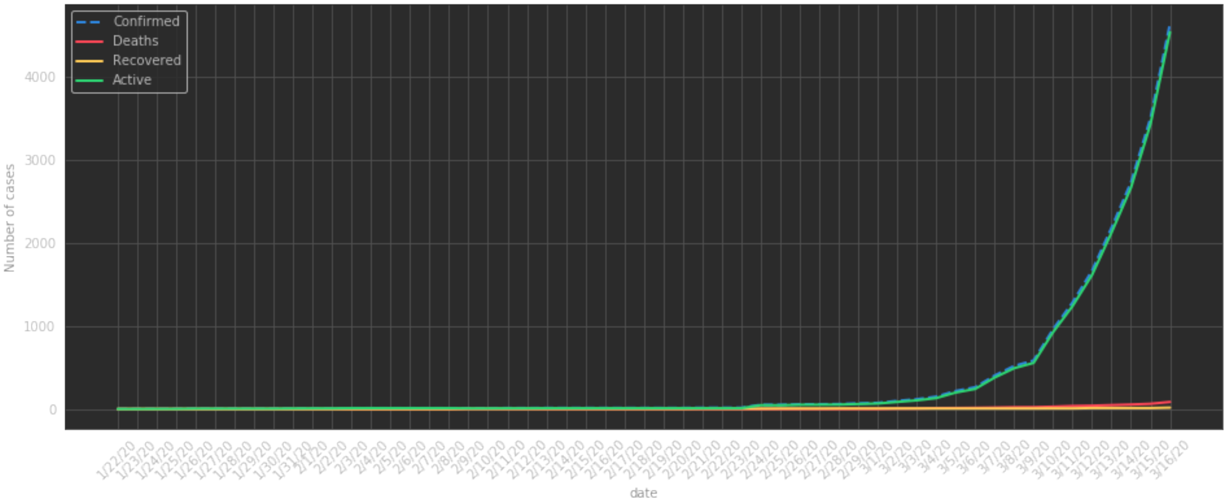
In [40]:

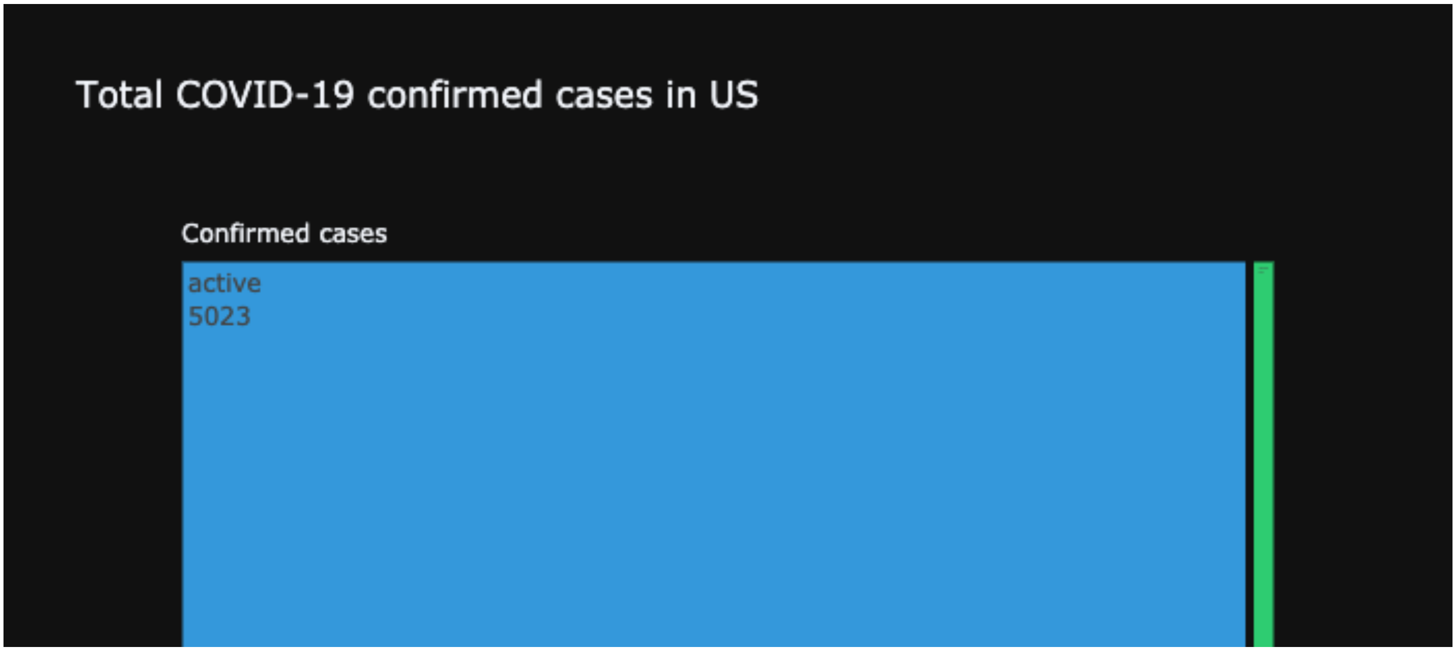
```
def get_country_covid_info(country, log=False):
    plot_country_global_info(country)

    plot_country_cases_over_time(country, log)

    plot_province_cases(country)
```

Finally, let's make some calls to our function:

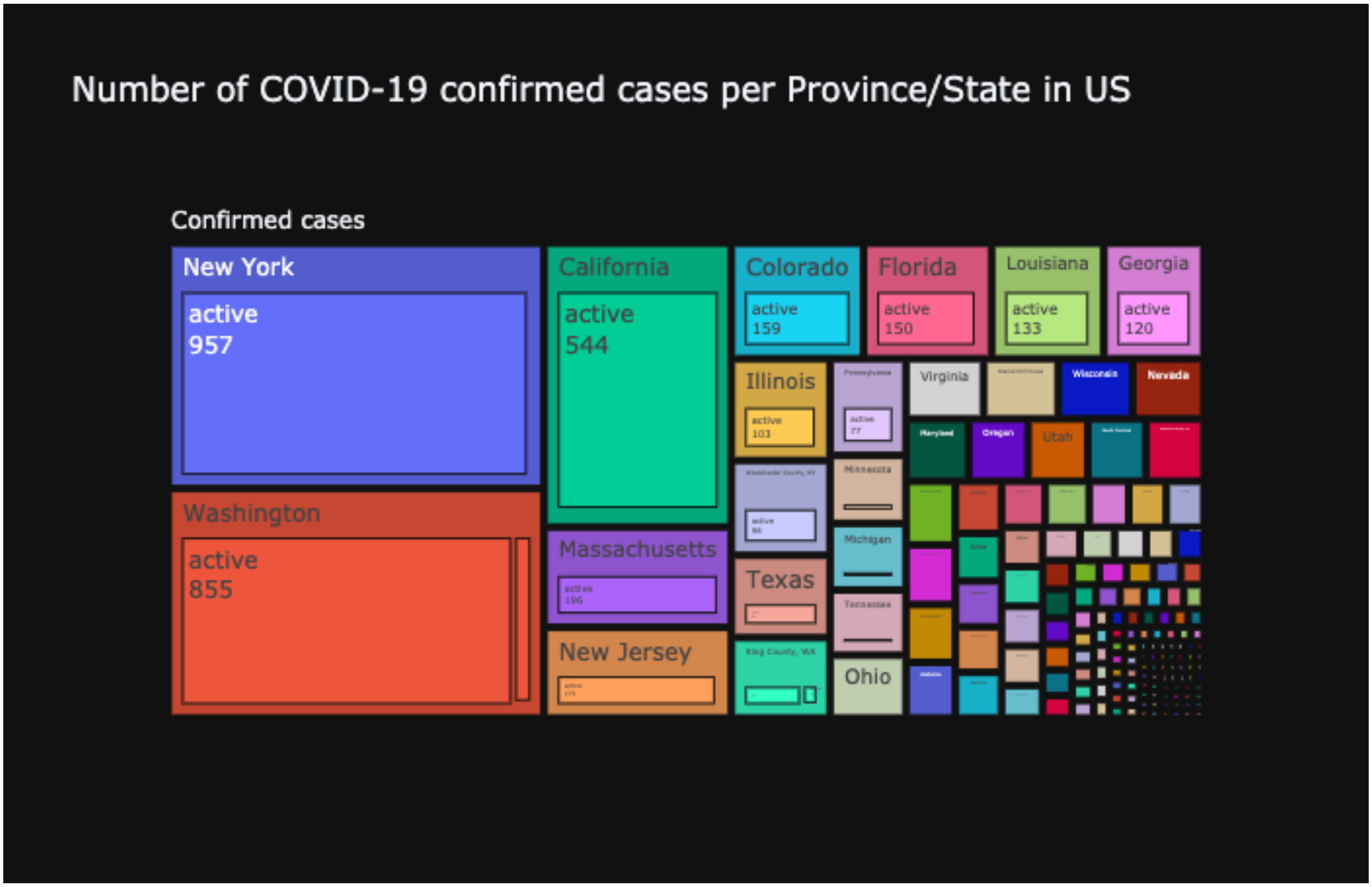
In [41]:

```
get_country_covid_info('US')
```

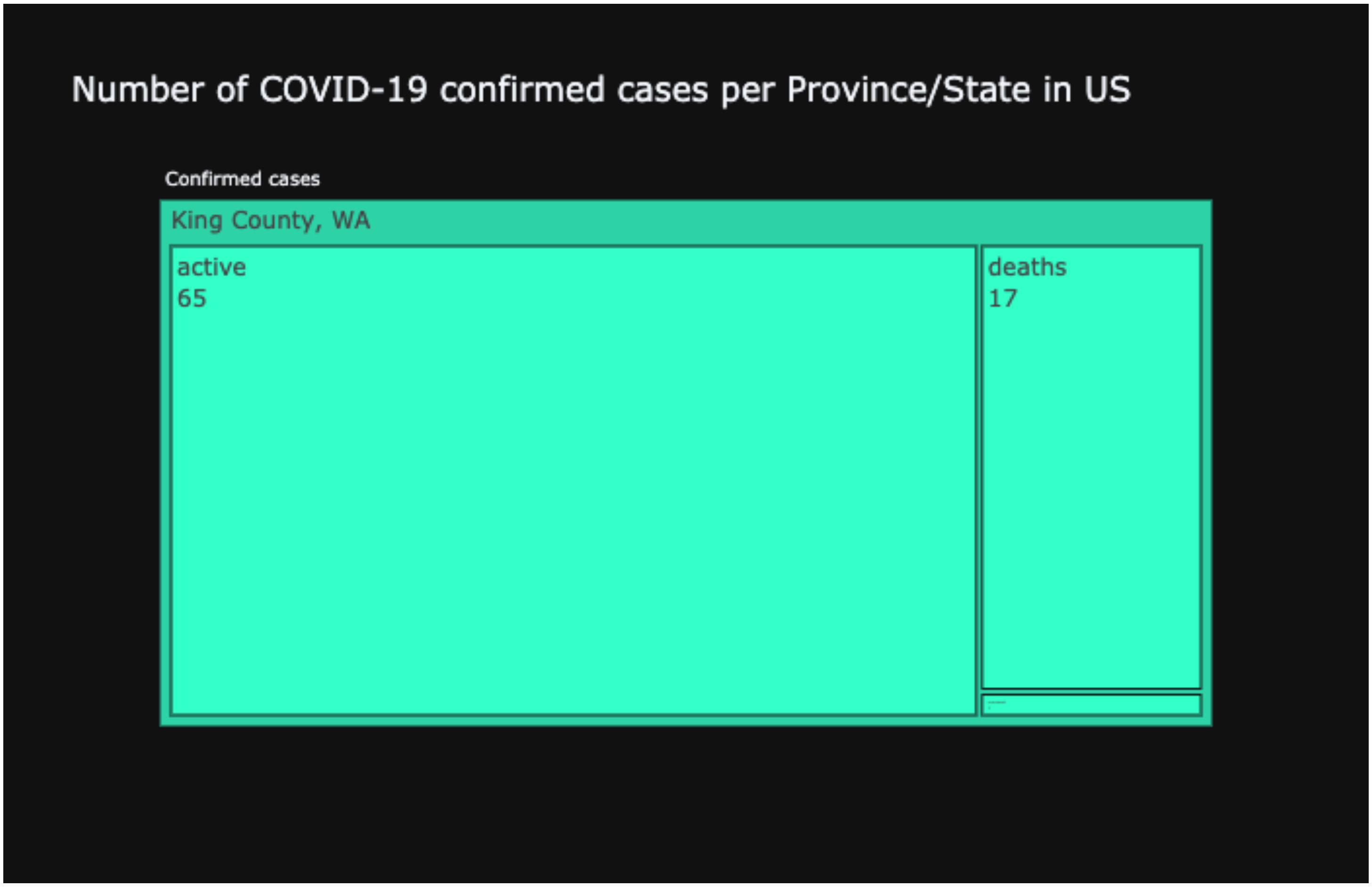

Confirmed
Deaths
Recovered
Active
4000
3000
2000
1000
0
Number of cases
1/22/20
1/23/20
1/24/20
1/25/20
1/26/20
1/27/20
1/28/20
1/29/20
1/30/20
1/31/20
2/1/20
2/2/20
2/3/20
2/4/20
2/5/20
2/6/20
2/7/20
2/8/20
2/9/20
2/10/20
2/11/20
2/12/20
2/13/20
2/14/20
2/15/20
2/16/20
2/17/20
2/18/20
2/19/20
2/20/20
2/21/20
2/22/20
2/23/20
2/24/20
2/25/20
2/26/20
2/27/20
2/28/20
2/29/20
3/1/20
3/2/20
3/3/20
3/4/20
3/5/20
3/6/20
3/7/20
3/8/20
3/9/20
3/10/20
3/11/20
3/12/20
3/13/20
3/14/20
3/15/20
3/16/20
date



Total COVID-19 confirmed cases in US

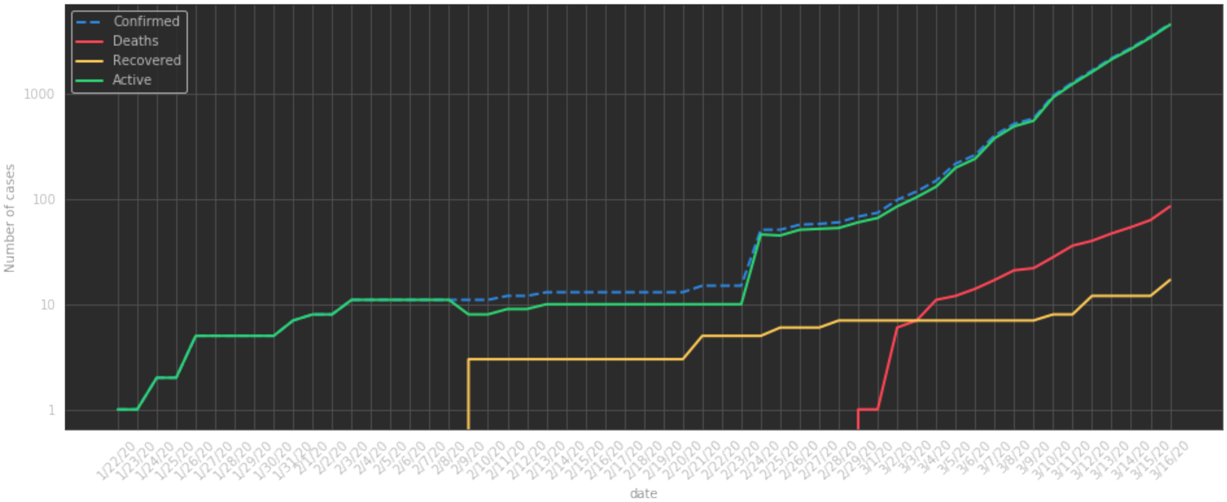


Number of COVID-19 confirmed cases per Province/State in US



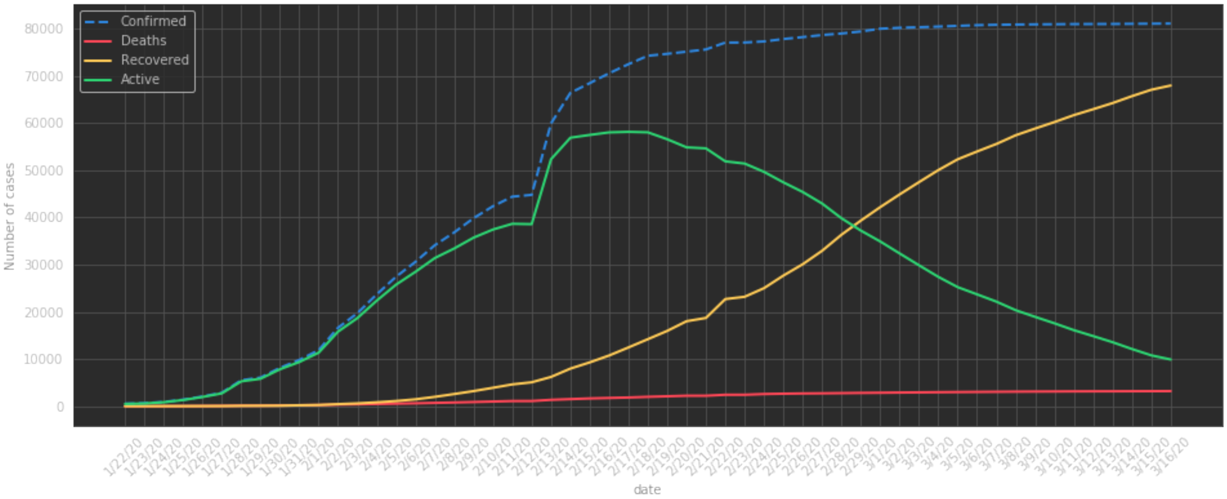
In [42]:

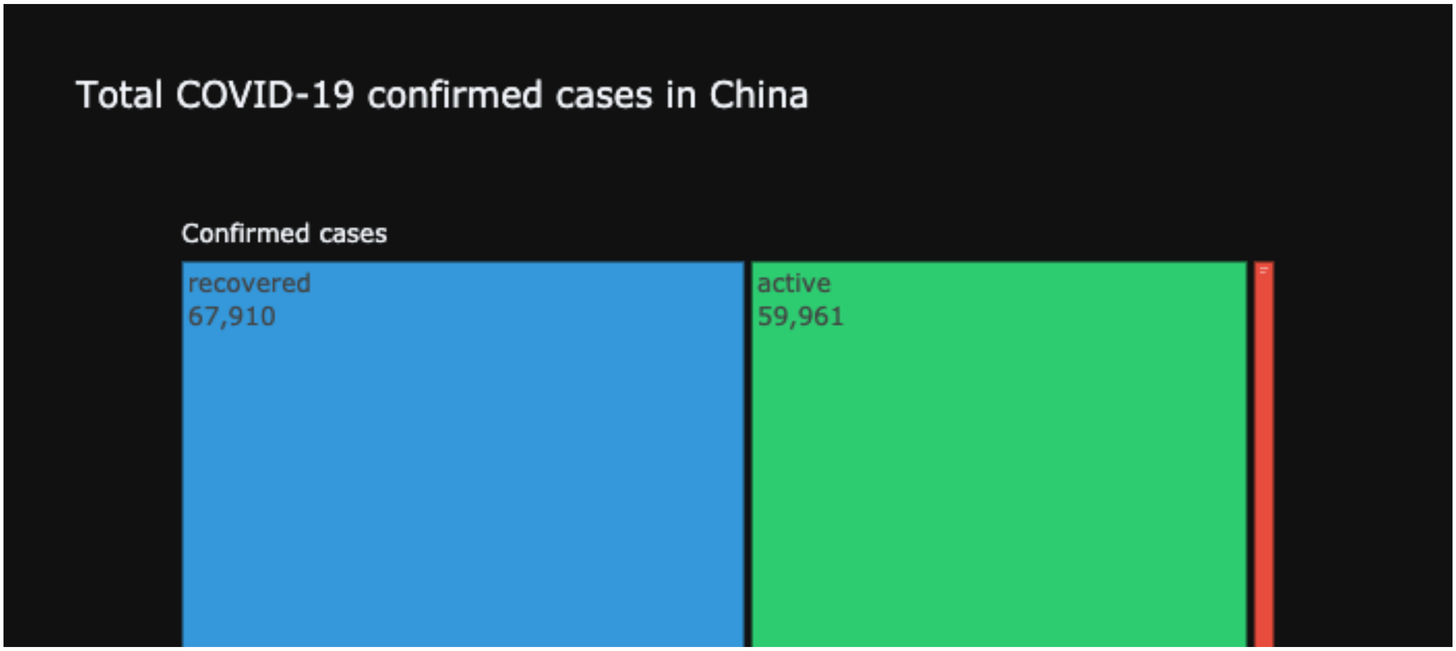
```
get_country_covid_info('US', log=True)
```

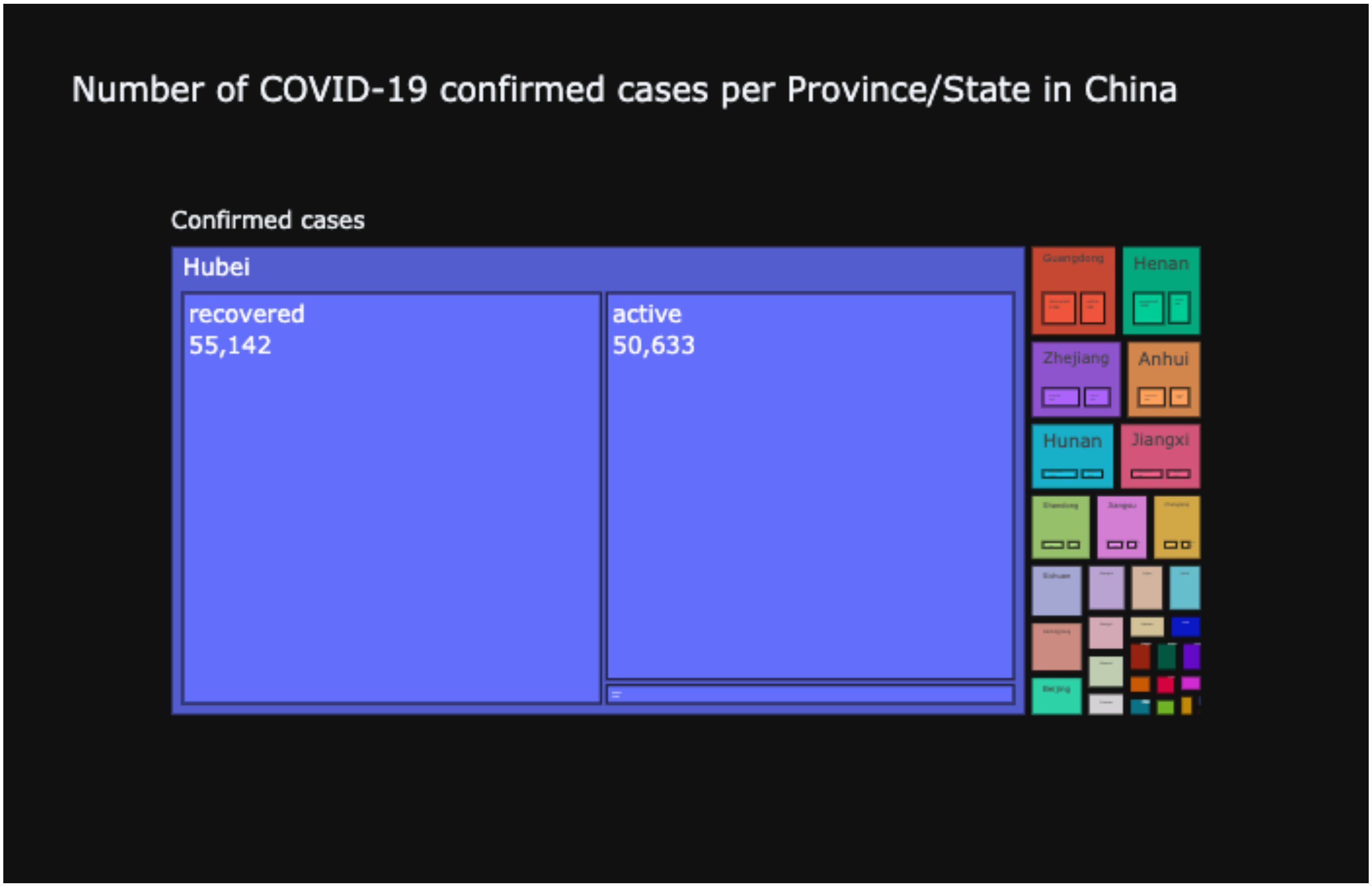

Confirmed
Deaths
Recovered
Active
Number of cases
1000
100
10
1
1/22/20
1/23/20
1/24/20
1/25/20
1/26/20
1/27/20
1/28/20
1/29/20
1/30/20
1/31/20
2/1/20
2/2/20
2/3/20
2/4/20
2/5/20
2/6/20
2/7/20
2/8/20
2/9/20
2/10/20
2/11/20
2/12/20
2/13/20
2/14/20
2/15/20
2/16/20
2/17/20
2/18/20
2/19/20
2/20/20
2/21/20
2/22/20
2/23/20
2/24/20
2/25/20
2/26/20
2/27/20
2/28/20
2/29/20
3/1/20
3/2/20
3/3/20
3/4/20
3/5/20
3/6/20
3/7/20
3/8/20
3/9/20
3/10/20
3/11/20
3/12/20
3/13/20
3/14/20
3/15/20
3/16/20
date

In [43]:

```
get_country_covid_info('China')
```


Confirmed
Deaths
Recovered
Active
80000
70000
60000
50000
40000
30000
20000
10000
0
Number of cases
1/22/20
1/23/20
1/24/20
1/25/20
1/26/20
1/27/20
1/28/20
1/29/20
1/30/20
1/31/20
2/1/20
2/2/20
2/3/20
2/4/20
2/5/20
2/6/20
2/7/20
2/8/20
2/9/20
2/10/20
2/11/20
2/12/20
2/13/20
2/14/20
2/15/20
2/16/20
2/17/20
2/18/20
2/19/20
2/20/20
2/21/20
2/22/20
2/23/20
2/24/20
2/25/20
2/26/20
2/27/20
2/28/20
2/29/20
3/1/20
3/2/20
3/3/20
3/4/20
3/5/20
3/6/20
3/7/20
3/8/20
3/9/20
3/10/20
3/11/20
3/12/20
3/13/20
3/14/20
3/15/20
3/16/20
date


Total COVID-19 confirmed cases in China
Confirmed cases
recovered
67,910
active
59,961



Go ahead and try other countries!

In [ ]:

```
get_country_covid_info('OTHER_COUNTRY')
```