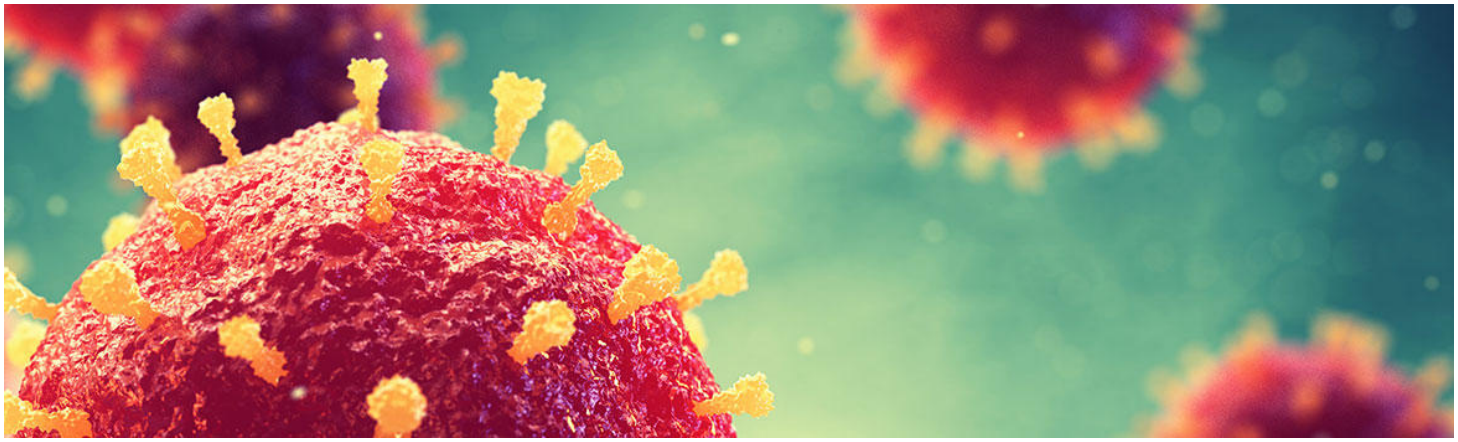


Analyzing the epidemiological outbreak of COVID-19

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
```

Step 1: Reading Data

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).

This data is daily-updated, so we can keep our project up-to-date just by loading this data again.

Let's load the data and quickly analyze it's columns and values:

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)

print(covid_confirmed.shape)

covid_confirmed.head()
```

```
(463, 59)
```

Out[2]:

	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

In [3]:

```
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)

print(covid_confirmed.shape)

covid_deaths.head()
```

```
(463, 59)
```

Out[3]:

	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	0	0	0	0	0
1	NaN	Japan	36.000	138.000	0	0	0	0	0
2	NaN	Singapore	1.283	103.833	0	0	0	0	0
3	NaN	Nepal	28.167	84.250	0	0	0	0	0
4	NaN	Malaysia	2.500	112.500	0	0	0	0	0

In [4]:

```
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)

print(covid_recovered.shape)

covid_recovered.head()
```

```
(463, 59)
```

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	0	0	0	0	2
1	NaN	Japan	36.000	138.000	0	0	0	0	1
2	NaN	Singapore	1.283	103.833	0	0	0	0	0
3	NaN	Nepal	28.167	84.250	0	0	0	0	0
4	NaN	Malaysia	2.500	112.500	0	0	0	0	0

> You can learn how to read other type of files using Pandas on our ***Reading Data with Pandas and Python*** course (https://rmotr.com/reading-data-with-python-and-pandas/)!

We are using `DataFrame` s to store our data. A pandas `DataFrame` is two-dimensional size-mutable, potentially heterogeneous tabular data structure with labeled axes (rows and columns).

So far we have all our datasets loaded, let's analyze them!

Step 2: Cleaning our data

Another important step before diving into data analysis is cleaning the data.

As the data is already really clean, we'll just replace `Mainland china` with just `China` , and fill some missing values.

> You can learn more on data cleaning on our ***Data Cleaning with pandas*** course (https://rmotr.com/data-cleaning-with-pandas/)!

In [5]:

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

In [6]:

```
covid_confirmed[['Province/State']] = covid_confirmed[['Province/State']].fillna('')
covid_confirmed.fillna(0, inplace=True)

covid_deaths[['Province/State']] = covid_deaths[['Province/State']].fillna('')
covid_deaths.fillna(0, inplace=True)

covid_recovered[['Province/State']] = covid_recovered[['Province/State']].fillna('')
covid_recovered.fillna(0, inplace=True)
```

Final checks:

In [7]:

```
covid_confirmed.isna().sum().sum()
```

Out[7]:

0

In [8]:

```
covid_deaths.isna().sum().sum()
```

Out[8]:

0

In [9]:

```
covid_recovered.isna().sum().sum()
```

Out[9]:

0

Step 3 & 4: Analysis (worldwide impact) and Data Wrangling

With the data loaded, we will start by aggregating all the cases so we can quickly see what's going on in the world.

To do that we'll use the pandas Python library.

> **pandas** is the most popular Python library for Data Science. You can learn data analysis fundamentals using pandas on our ***Intro to Pandas for Data Analysis*** course (https://rmotr.com/intro-to-pandas-for-data-analysis/)!

In [10]:

```
covid_confirmed_count = covid_confirmed.iloc[:, 4:].sum().max()

covid_confirmed_count
```

Out[10]:

181546

In [11]:

```
covid_deaths_count = covid_deaths.iloc[:, 4:].sum().max()

covid_deaths_count
```

Out[11]:

```
7126
```

In [12]:

```
covid_recovered_count = covid_recovered.iloc[:, 4:].sum().max()

covid_recovered_count
```

Out[12]:

```
78088
```

Store that values on a `DataFrame` , and calculate a new `active` cases value with the following formula: $$ Active = Confirmed - Deaths - Recovered $$

In [13]:

```
world_df = pd.DataFrame({
    'confirmed': [covid_confirmed_count],
    'deaths': [covid_deaths_count],
    'recovered': [covid_recovered_count],
    'active': [covid_confirmed_count - covid_deaths_count - covid_recovered_coun
t]
})

world_df
```

Out[13]:

	confirmed	deaths	recovered	active
0	181546	7126	78088	96332

In [14]:

```
world_long_df = world_df.melt(value_vars=['active', 'deaths', 'recovered'],
                              var_name="status",
                              value_name="count")

world_long_df['upper'] = 'confirmed'

world_long_df
```

Out[14]:

	status	count	upper
0	active	96332	confirmed
1	deaths	7126	confirmed
2	recovered	78088	confirmed

In [15]:

```
fig = px.treemap(world_long_df, path=["upper", "status"], values="count",
                 color_discrete_sequence=['#3498db', '#2ecc71', '#e74c3c'],
                 template='plotly_dark')

fig.show()
```

We see that almost half of the cases are still active!

Worldwide over the time evolution analysis

Let's make a more convenient plot showing how these cases increased day by day.

As we want to analyze daily worldwide aggregated values, let's remove unused columns (`Province/State` , `Country/Region` , `Lat` , `Long`) and aggregate the columns we need (all the other columns):

In [16]:

```
covid_worldwide_confirmed = covid_confirmed.iloc[:, 4:].sum(axis=0)

covid_worldwide_confirmed.head()
```

Out[16]:

```
1/22/20     555
1/23/20     653
1/24/20     941
1/25/20    1434
1/26/20    2118
dtype: int64
```

In [17]:

```
covid_worldwide_deaths = covid_deaths.iloc[:, 4:].sum(axis=0)

covid_worldwide_deaths.head()
```

Out[17]:

```
1/22/20    17
1/23/20    18
1/24/20    26
1/25/20    42
1/26/20    56
dtype: int64
```

In [18]:

```
covid_worldwide_recovered = covid_recovered.iloc[:, 4:].sum(axis=0)

covid_worldwide_recovered.head()
```

Out[18]:

```
1/22/20    28
1/23/20    30
1/24/20    36
1/25/20    39
1/26/20    52
dtype: int64
```

Also, we can calculate `active` cases again:

In [19]:

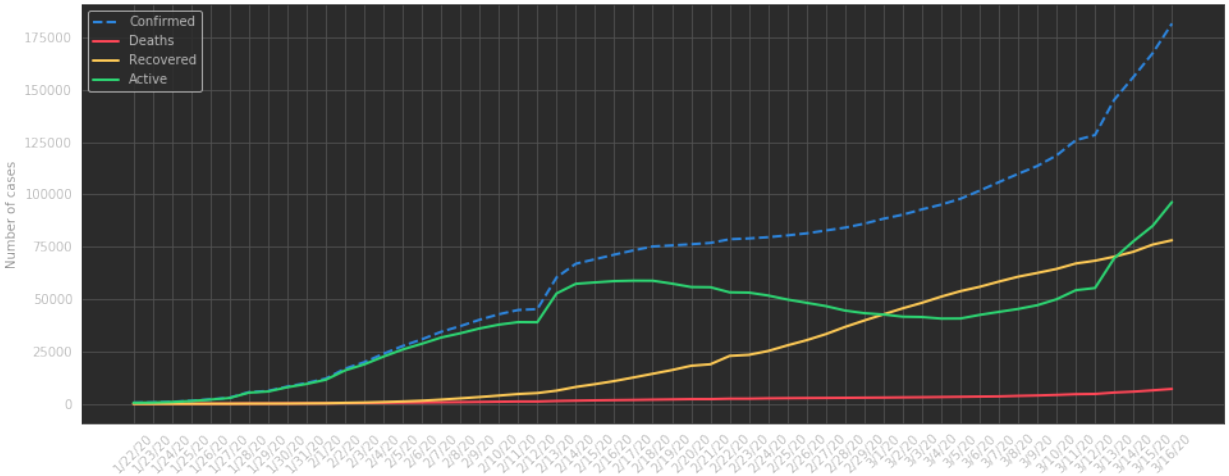
```
covid_worldwide_active = covid_worldwide_confirmed - covid_worldwide_deaths - co
vid_worldwide_recovered

covid_worldwide_active.head()
```

Out[19]:

```
1/22/20     510
1/23/20     605
1/24/20     879
1/25/20    1353
1/26/20    2010
dtype: int64
```

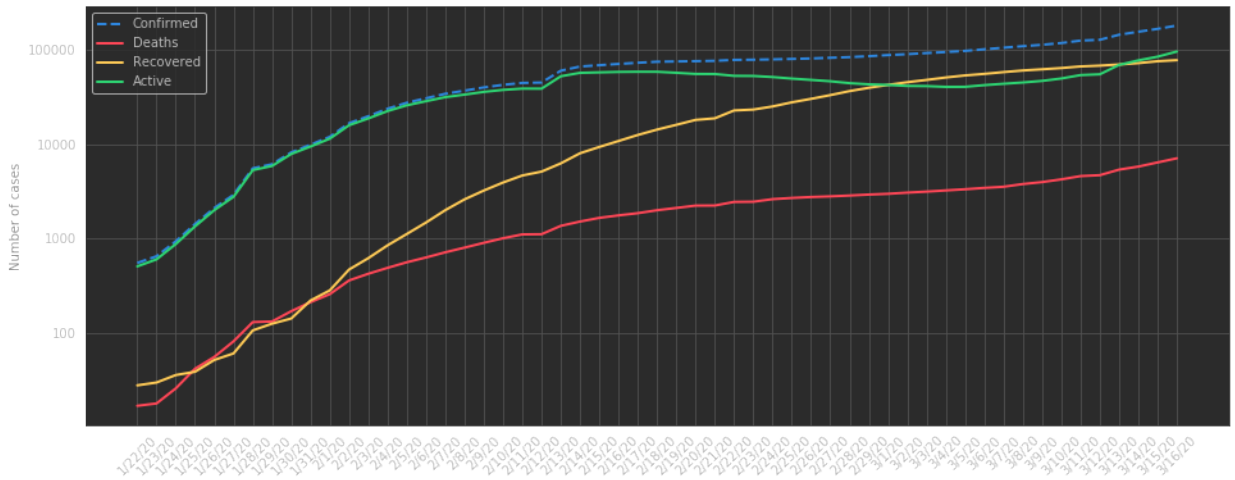
In [20]:

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

sns.lineplot(x=covid_worldwide_confirmed.index, y=covid_worldwide_confirmed, sor
t=False, linewidth=2)
sns.lineplot(x=covid_worldwide_deaths.index, y=covid_worldwide_deaths, sort=Fals
e, linewidth=2)
sns.lineplot(x=covid_worldwide_recovered.index, y=covid_worldwide_recovered, sor
t=False, linewidth=2)
sns.lineplot(x=covid_worldwide_active.index, y=covid_worldwide_active, sort=Fals
e, linewidth=2)

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

plt.suptitle("COVID-19 worldwide cases over the time evolution", fontsize=16, fo
ntweight='bold', color='white')

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

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

plt.show()
```



In [21]:

```
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_worldwide_confirmed.index, y=covid_worldwide_confirmed, sor
t=False, linewidth=2)
sns.lineplot(x=covid_worldwide_deaths.index, y=covid_worldwide_deaths, sort=Fals
e, linewidth=2)
sns.lineplot(x=covid_worldwide_recovered.index, y=covid_worldwide_recovered, sor
t=False, linewidth=2)
sns.lineplot(x=covid_worldwide_active.index, y=covid_worldwide_active, sort=Fals
e, linewidth=2)

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

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

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

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

plt.show()
```



Recovery and mortality rate over time

In [22]:

```
world_rate_df = pd.DataFrame({
    'confirmed': covid_worldwide_confirmed,
    'deaths': covid_worldwide_deaths,
    'recovered': covid_worldwide_recovered,
    'active': covid_worldwide_active
}, index=covid_worldwide_confirmed.index)

world_rate_df.tail()
```

Out[22]:

	confirmed	deaths	recovered	active
3/12/20	128343	4720	68324	55299
3/13/20	145193	5404	70251	69538
3/14/20	156097	5819	72624	77654
3/15/20	167449	6440	76034	84975
3/16/20	181546	7126	78088	96332

In [23]:

```
world_rate_df['recovered / 100 confirmed'] = world_rate_df['recovered'] / world_
rate_df['confirmed'] * 100

world_rate_df['deaths / 100 confirmed'] = world_rate_df['deaths'] / world_rate_d
f['confirmed'] * 100

world_rate_df['date'] = world_rate_df.index

world_rate_df.tail()
```

Out[23]:

	confirmed	deaths	recovered	active	recovered / 100 confirmed	deaths / 100 confirmed	date
3/12/20	128343	4720	68324	55299	53.235	3.678	3/12/20
3/13/20	145193	5404	70251	69538	48.385	3.722	3/13/20
3/14/20	156097	5819	72624	77654	46.525	3.728	3/14/20
3/15/20	167449	6440	76034	84975	45.407	3.846	3/15/20
3/16/20	181546	7126	78088	96332	43.013	3.925	3/16/20

In [24]:

```
world_rate_long_df = world_rate_df.melt(id_vars="date",
                                        value_vars=['recovered / 100 confirmed',
'deaths / 100 confirmed'],
                                        var_name="status",
                                        value_name="ratio")

world_rate_long_df
```

Out[24]:

	date	status	ratio
0	1/22/20	recovered / 100 confirmed	5.045
1	1/23/20	recovered / 100 confirmed	4.594
2	1/24/20	recovered / 100 confirmed	3.826
3	1/25/20	recovered / 100 confirmed	2.720
4	1/26/20	recovered / 100 confirmed	2.455
...	...	...	...
105	3/12/20	deaths / 100 confirmed	3.678
106	3/13/20	deaths / 100 confirmed	3.722
107	3/14/20	deaths / 100 confirmed	3.728
108	3/15/20	deaths / 100 confirmed	3.846
109	3/16/20	deaths / 100 confirmed	3.925

110 rows × 3 columns

In [25]:

```
fig = px.line(world_rate_long_df, x="date", y="ratio", color='status', log_y=True,
              title='Recovery and Mortality rate over the time',
              color_discrete_sequence=['#2ecc71', '#e74c3c'],
              template='plotly_dark')

fig.show()
```

Visualizing worldwide COVID-19 cases in a map

We'll now create a small animation showing COVID-19 confirmed cases through the days.

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

Hands on! Let's group rows with the same value at the `Country/Region` column, so we can aggregate all the values from each country in a single aggregated value. We'll use the `sum()` method to count all the values from the same country.

In [26]:

```
covid_confirmed_agg = covid_confirmed.groupby('Country/Region').sum().reset_index()
```

As there could be many Provinces/States within the same country, we'll calculate the `mean` latitude and longitude for each country.

In [27]:

```
covid_confirmed_agg.loc[:, ['Lat', 'Long']] = covid_confirmed.groupby('Country/Region').mean().reset_index().loc[:, ['Lat', 'Long']]
```

In [28]:

```
covid_confirmed_agg
```

Out[28]:

	Country/Region	Lat	Long	1/22/20	1/23/20	1/24/20	1/25/20	1/26/20	1/27/20	1/
0	Afghanistan	33.000	65.000	0	0	0	0	0	0	
1	Albania	41.153	20.168	0	0	0	0	0	0	
2	Algeria	28.034	1.660	0	0	0	0	0	0	
3	Andorra	42.506	1.522	0	0	0	0	0	0	
4	Antigua and Barbuda	17.061	-61.796	0	0	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	
151	Uruguay	-32.523	-55.766	0	0	0	0	0	0	
152	Uzbekistan	41.377	64.585	0	0	0	0	0	0	
153	Venezuela	6.424	-66.590	0	0	0	0	0	0	
154	Vietnam	16.000	108.000	0	2	2	2	2	2	
155	occupied Palestinian territory	31.952	35.233	0	0	0	0	0	0	

156 rows × 58 columns

Now we'll do is filtering countries with more than a `MIN_CASES` value, in this case we'll use `100` cases.

In [29]:

```
MIN_CASES = 100

covid_confirmed_agg = covid_confirmed_agg[covid_confirmed_agg.iloc[:, 3:].max(ax
is=1) > MIN_CASES]
```

Our data is now ready, but in a **wrong format**, so we'll need to transform our data **from wide to long format**, to do that we'll use the `melt()` pandas method.

In [30]:

```
print(covid_confirmed_agg.shape)

covid_confirmed_agg.head()
```

```
(48, 58)
```

Out[30]:

	Country/Region	Lat	Long	1/22/20	1/23/20	1/24/20	1/25/20	1/26/20	1/27/20	1/2
8	Australia	-24.503	141.056	0	0	0	0	4	5	
9	Austria	47.516	14.550	0	0	0	0	0	0	
11	Bahrain	26.027	50.550	0	0	0	0	0	0	
14	Belgium	50.833	4.000	0	0	0	0	0	0	
19	Brazil	-14.235	-51.925	0	0	0	0	0	0	

In [31]:

```
covid_confirmed_agg_long = pd.melt(covid_confirmed_agg,
                                   id_vars=covid_confirmed_agg.iloc[:, :3],
                                   var_name='date',
                                   value_vars=covid_confirmed_agg.iloc[:, 3:],
                                   value_name='date_confirmed_cases')
```

In [32]:

```
print(covid_confirmed_agg_long.shape)

covid_confirmed_agg_long.head()
```

```
(2640, 5)
```

Out[32]:

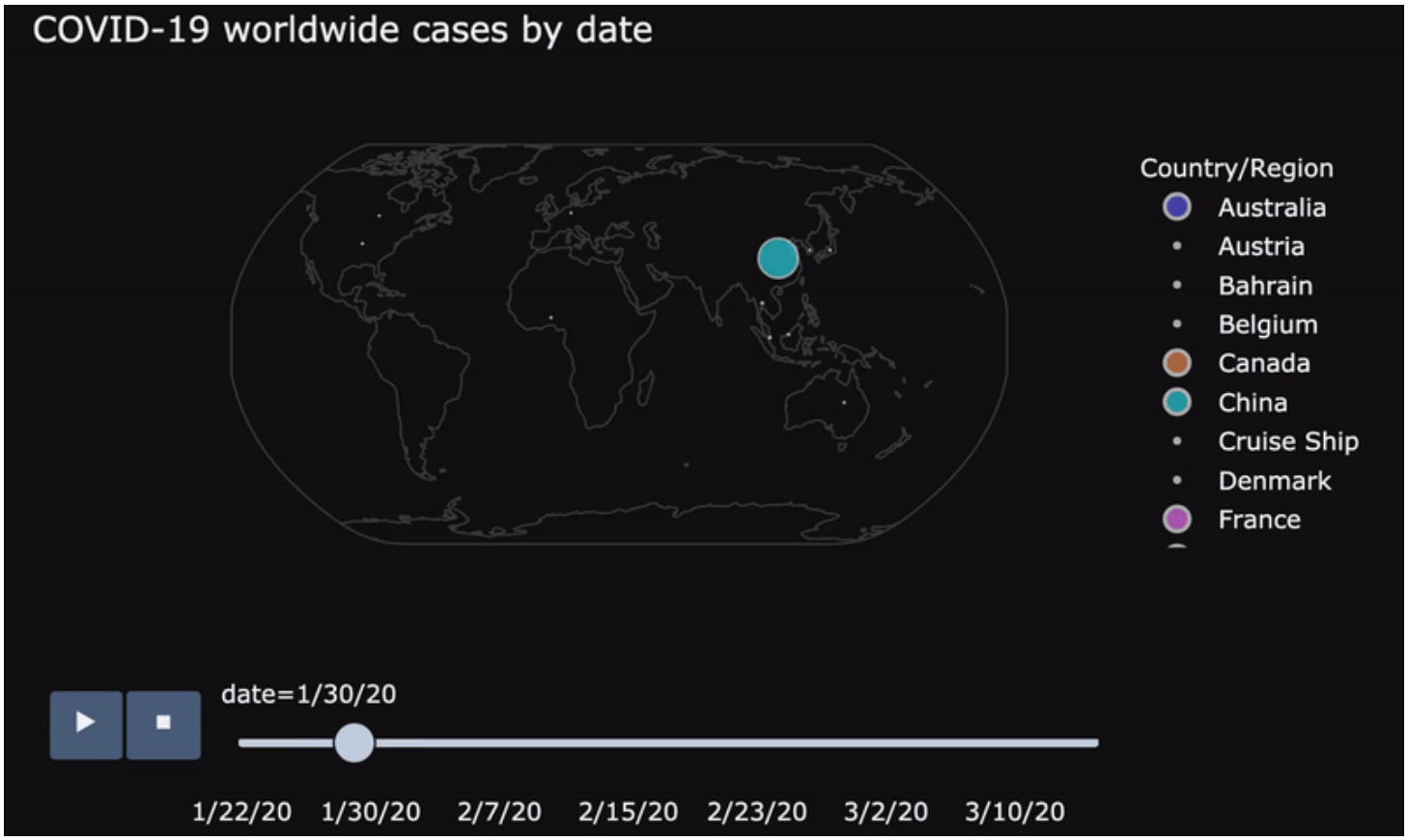
	Country/Region	Lat	Long	date	date_confirmed_cases
0	Australia	-24.503	141.056	1/22/20	0
1	Austria	47.516	14.550	1/22/20	0
2	Bahrain	26.027	50.550	1/22/20	0
3	Belgium	50.833	4.000	1/22/20	0
4	Brazil	-14.235	-51.925	1/22/20	0

Finally, let's use Plotly (https://plot.ly/python/) to create a worldwide visualization.

(this could take a few seconds...)

In [33]:

```
fig = px.scatter_geo(covid_confirmed_agg_long,
                     lat="Lat", lon="Long", color="Country/Region",
                     hover_name="Country/Region", size="date_confirmed_cases",
                     size_max=50, animation_frame="date",
                     template='plotly_dark', projection="natural earth",
                     title="COVID-19 worldwide confirmed cases over time")

fig.show()
```


COVID-19 worldwide cases by date
Country/Region
Australia
Austria
Bahrain
Belgium
Canada
China
Cruise Ship
Denmark
France
date=1/30/20
1/22/20
1/30/20
2/7/20
2/15/20
2/23/20
3/2/20
3/10/20