Pandemic COVID-19 Analysis and Visualization

Journal: International Journal of Infectious Diseases

Md. Abrar Jahin

Email: abrar.jahin.2652@gmail.com

2nd year, Industrial Engineering and Management

Abstract

The 2019-nCoV is a contagious coronavirus that hailed from Wuhan, China. This new strain of virus has striked fear in many countries as cities are quarantined and hospitals are overcrowded. This dataset will help us understand how 2019-nCoV is spread aroud the world. In the following dataset global death rate, new cases, new recovered, confirmed, active, change in 1 week and %increment, total population, timestamps etc. have been included and considered. Useful Python data analysis & visualization libraries numpy, pandas, matplotlib, seaborn etc. have been used to answer

Motivation

This analysis and visualization is the key factor of survival for 2020 while the USA is facing continuous death increase, economic fall and losing equilibrium of the system in spite of having such great power and developed technology. I tried to visualize the chain multiplication of the virus overtime and compare the death, active, recovered cases within a certain country and globally as well. The map showing the Red Spots indicates Corona affected zones and epidemic span, comparison with SARS, Ebola, H1N1 etc. also have been shown to relate the insights to reduce the spread, find a pattern. Less affected and fast recovered countries and their reasons have been analyzed.

Dataset(s)

The datasets have been collected from:

1. https://cgdv.github.io/challenges/COVID-19/datasource/
2. https://github.com/CSSEGISandData/COVID-19
3. https://github.com/imdevskp/covid_19_jhu_data_web_scrap_and_cleaning

There are confirmed positive, death, recovered, active cases mentioned for each countries and the data regarding spread and death rate in other epidemics have been enlisted. The datasets are authentic, reliable and clean enough compared to other sites.

Data Preparation and Cleaning

At first the pandemic cases are totally out of control and pattern-less which caused outliers for a few times. Data cleaning including finding null values, filtering a certain number of cases as well as merging data resulted in our final outputs. The inner join, group-by etc. have been done based on the countries most of the time. It is the lengthiest process among all steps and takes a lot of patience and trials. Fortunately there were no null values which didn’t cause panic while visualizing graphs. Still the death, recover, confirmed cases are changing continuously which compels the data sets to be upgraded from time to time.

Research Question(s)

- What is the impact of Covid 19 on marginalized populations?
- Based on US/Western populations across LMICs, where are these populations most at risk based on the current rate of spread of Covid-19?
- Based on current situations and physical distancing norms where are the next ‘hot zones’ going to develop around the world?
- Given levels of physical distancing – will this be sufficient to flatten the curve?
- What will be the impact of no-go zones - marginalized areas in cities - on the ability to stop the spread of the virus
- Are countries that have active citizen engagement mitigating the spread of Covid 19 more effectively?
- What is the nature of the changing relationship between China and Taiwan, Hong Kong, the US, Germany, Italy, France, the UK and others?
- What would be the impact of religious distress due to COVID 19 on particular community in India?
- What has been the economic impact of Covid 19 on the US, China, Russia, and Germany?

Methods

Using Python numpy, pandas, matplotlib, seaborn, datetime, math libraries the whole analysis has been done including graphs, dataframes, charts, maps etc. The whole methods have been attached with the existing pdf file.

Findings

jupyter covid-19-analysis-visualization-comparisons Last Checkpoint: an hour ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

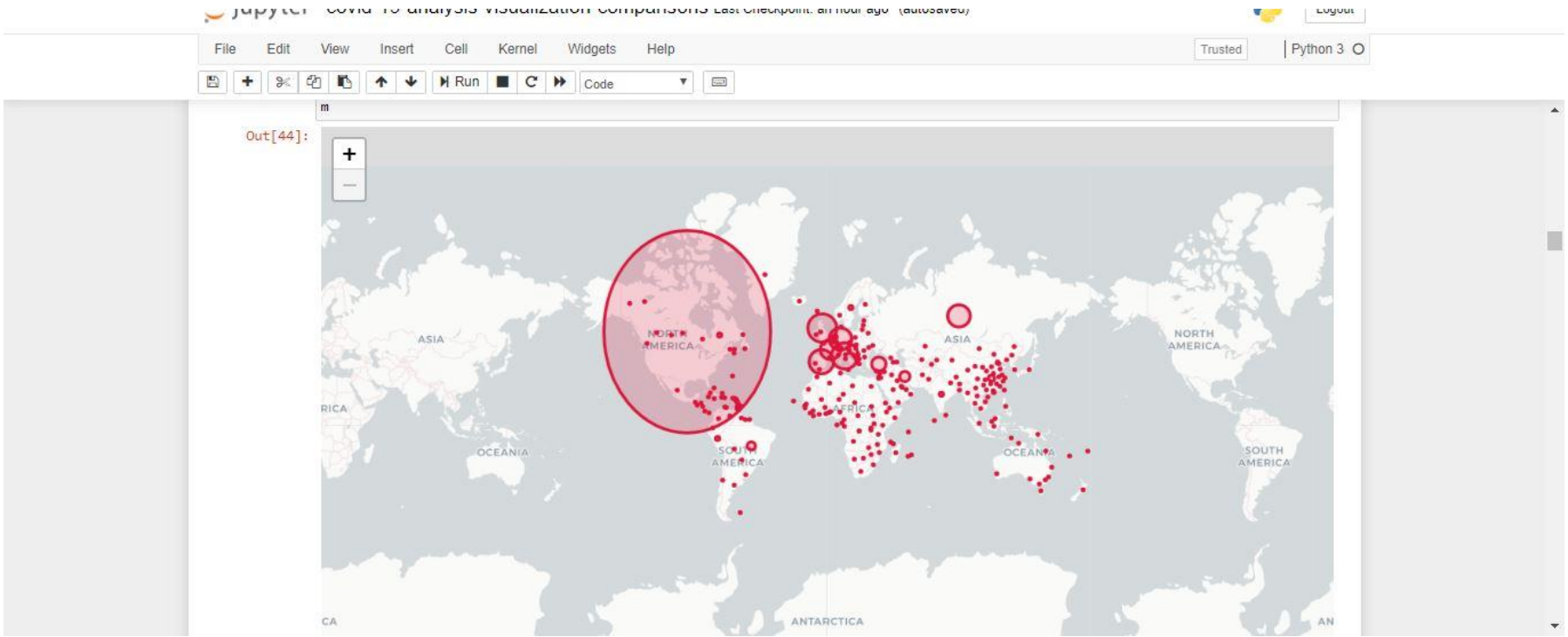
Run Code

Across the world

In [44]:
```
# World wide

temp = full_table[full_table['Date'] == max(full_table['Date'])]

m = folium.Map(location=[0, 0], tiles='cartodbpositron',
               min_zoom=1, max_zoom=4, zoom_start=1)

for i in range(0, len(temp)):
    folium.Circle(
        location=[temp.iloc[i]['Lat'], temp.iloc[i]['Long']],
        color='crimson', fill='crimson',
        tooltip =   '<li><bold>Country : '+str(temp.iloc[i]['Country/Region'])+
                    '<li><bold>Province : '+str(temp.iloc[i]['Province/State'])+
                    '<li><bold>Confirmed : '+str(temp.iloc[i]['Confirmed'])+
                    '<li><bold>Deaths : '+str(temp.iloc[i]['Deaths']),
        radius=int(temp.iloc[i]['Confirmed'])**1.1).add_to(m)
m
```

Out[44]:

Findings


File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Trusted
Python 3
Run
Code
Out[44]:
ASIA
NORTH AMERICA
OCEANIA
SOUTH AMERICA
ANTARCTICA

Findings

The USA has been suffering from COVID-19 mostly in spite of having developed medical equipments and technologies which is a remarkable threat to the other developing as well as developed countries too. Italy has almost been damaged by it though they possess the best medical facilities. The economic condition is breaking in these developed European countries as well as South East Asian countries including India, Bangladesh, Pakistan etc. just because of having an innumerable and unorganized population it became so tough to keep them quarantined for a few months. The people are blind to religious cultures and taboos like basking in the sun will keep them safe from the virus. The virus didn’t spread within a day rather it took a few months which was ignored by the administrators and resulted in uncontrolled situation. The only fear is that there is

Findings

No vaccine for the virus still invented which can control that chain multiplication. Moreover, the government of developed countries are providing services and foods to the nation while the developing and under developed countries are deprived of that which compels the govt. to keep some of the industries open just to keep the country mobile.
Social distancing can definitely flatten the curve and lengthen the contamination and the medical team will get time to treat the patients otherwise a huge portion of the nation will be dead if quarantine rule is not being followed. The active citizens i.e. military, police, doctors, food delivery men are contributing the highest in this situation and if the proper regulations are not being followed, it can

Findings

Even cause more danger to the general people. The jobless people are in tension regarding managing food for themselves where simple sympathy and help to each other can make the society more worthy for living and pandemic is temporary but that love towards each other is permanent.

Acknowledgements

https://github.com/CSSEGISandData/COVID-19 (https://github.com/CSSEGISandData/COVID-19)

Collection methodology

https://github.com/imdevskp/covid_19_jhu_data_web_scrap_and_cleaning (https://github.com/imdevskp/covid_19_jhu_data_web_scrap_and_cleaning)

Disclaimer

- The data is scrapped from JHU github repository. Any variation in the data will also reflect in this notebook.

In [1]:
```
from IPython.core.display import HTML
```

COVID-19

Libraries

In [2]:
```
# install calmap
# ==============

!pip install calmap
!pip install plotly
!pip install folium
```

```
Requirement already satisfied: calmap in h:\anaconda_python_3.7\lib\site-packages (0.0.7)
Requirement already satisfied: numpy in h:\anaconda_python_3.7\lib\site-packages (from calmap) (1.18.1)
Requirement already satisfied: pandas in h:\anaconda_python_3.7\lib\site-packages (from calmap) (1.0.1)
Requirement already satisfied: matplotlib in h:\anaconda_python_3.7\lib\site-packages (from calmap) (3.1.3)
Requirement already satisfied: pytz>=2017.2 in h:\anaconda_python_3.7\lib\site-packages (from pandas->calmap) (2019.3)
Requirement already satisfied: python-dateutil>=2.6.1 in h:\anaconda_python_3.7\lib\site-packages (from pandas->calmap) (2.8.1)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in h:\anaconda_python_3.7\lib\site-packages (from matplotlib->calmap) (2.4.6)
Requirement already satisfied: kiwisolver>=1.0.1 in h:\anaconda_python_3.7\lib\site-packages (from matplotlib->calmap) (1.1.0)
Requirement already satisfied: cycler>=0.10 in h:\anaconda_python_3.7\lib\site-packages (from matplotlib->calmap) (0.10.0)
Requirement already satisfied: six>=1.5 in h:\anaconda_python_3.7\lib\site-packages (from python-dateutil>=2.6.1->pandas->calmap) (1.14.0)
Requirement already satisfied: setuptools in h:\anaconda_python_3.7\lib\site-packages (from kiwisolver>=1.0.1->matplotlib->calmap) (45.2.0.post20200210)
Requirement already satisfied: plotly in h:\anaconda_python_3.7\lib\site-packages (4.6.0)
Requirement already satisfied: six in h:\anaconda_python_3.7\lib\site-packages (from plotly) (1.14.0)
Requirement already satisfied: retrying>=1.3.3 in h:\anaconda_python_3.7\lib\site-packages (from plotly) (1.3.3)
Requirement already satisfied: folium in h:\anaconda_python_3.7\lib\site-packages (0.10.1)
Requirement already satisfied: requests in h:\anaconda_python_3.7\lib\site-packages (from folium) (2.22.0)
Requirement already satisfied: branca>=0.3.0 in h:\anaconda_python_3.7\lib\site-packages (from folium) (0.4.0)
Requirement already satisfied: numpy in h:\anaconda_python_3.7\lib\site-packages (from folium) (1.18.1)
Requirement already satisfied: jinja2>=2.9 in h:\anaconda_python_3.7\lib\site-packages (from folium) (2.11.1)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in h:\anaconda_python_3.7\lib\site-packages (from requests->folium) (1.25.8)
Requirement already satisfied: idna<2.9,>=2.5 in h:\anaconda_python_3.7\lib\site-packages (from requests->folium) (2.8)
Requirement already satisfied: chardet<3.1.0,>=3.0.2 in h:\anaconda_python_3.7\lib\site-packages (from requests->folium) (3.0.4)
Requirement already satisfied: certifi>=2017.4.17 in h:\anaconda_python_3.7\lib\site-packages (from requests->folium) (2019.11.28)
Requirement already satisfied: six in h:\anaconda_python_3.7\lib\site-packages (from branca>=0.3.0->folium) (1.14.0)
Requirement already satisfied: MarkupSafe>=0.23 in h:\anaconda_python_3.7\lib\site-packages (from jinja2>=2.9->folium) (1.1.1)
```

In [3]:
```
# Import
# ======

# essential libraries
import math
import random
from datetime import timedelta

# storing and anaysis
import numpy as np
import pandas as pd

# visualization
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import plotly.graph_objs as go
import plotly.figure_factory as ff
from plotly.subplots import make_subplots
import calmap
import folium

# color pallette
cnf, dth, rec, act = '#393e46', '#ff2e63', '#21bf73', '#fe9801'

# converter
from pandas.plotting import register_matplotlib_converters
register_matplotlib_converters()

# hide warnings
import warnings
warnings.filterwarnings('ignore')
```

In [4]:
```
# for offline ploting
# ===================
from plotly.offline import plot, iplot, init_notebook_mode
init_notebook_mode(connected=True)
```

Dataset

In [5]:
```
# list files
# ==========

# !ls ./corona-virus-report
```

In [6]:
```
# importing datasets
# ==================

full_table = pd.read_csv('./corona-virus-report/Covid/covid_19_clean_complete.csv',
                         parse_dates=['Date'])
full_table.sample(6)
```

Out[6]:

	Province/State	Country/Region	Lat	Long	Date	Confirmed	Deaths	Recovered
15106	Hebei	China	39.5490	116.1306	2020-03-19	318	6	310
14885	NaN	Eritrea	15.1794	39.7823	2020-03-18	0	0	0
21023	Curacao	Netherlands	12.1696	-68.9900	2020-04-10	14	1	7
19525	NaN	Burundi	-3.3731	29.9189	2020-04-04	3	0	0
2892	NaN	Botswana	-22.3285	24.6849	2020-02-01	0	0	0
12821	NaN	Liechtenstein	47.1400	9.5500	2020-03-10	1	0	0

In [7]:
```
# dataframe info
# full_table.info()
```

In [8]:
```
# checking for missing value
# full_table.isna().sum()
```

Preprocessing

In [9]:
```
# Ship
# ====

# ship rows
ship_rows = full_table['Province/State'].str.contains('Grand Princess') | full_table['Province/State'].str.contains('Diamond Princess') | full_table['Country/Region'].str.contain
s('Diamond Princess') | full_table['Country/Region'].str.contains('MS Zaandam')

# ship
ship = full_table[ship_rows]

# full table
full_table = full_table[~(ship_rows)]

# Latest cases from the ships
ship_latest = ship[ship['Date']==max(ship['Date'])]

# ship_latest.style.background_gradient(cmap='Pastel1_r')
```

In [10]:
```
# Cleaning data
# =============

# Active Case = confirmed - deaths - recovered
full_table['Active'] = full_table['Confirmed'] - full_table['Deaths'] - full_table['Recovered']

# replacing Mainland china with just China
full_table['Country/Region'] = full_table['Country/Region'].replace('Mainland China', 'China')

# filling missing values
full_table[['Province/State']] = full_table[['Province/State']].fillna('')
full_table[['Confirmed', 'Deaths', 'Recovered', 'Active']] = full_table[['Confirmed', 'Deaths', 'Recovered', 'Active']].fillna(0)

# fixing datatypes
full_table['Recovered'] = full_table['Recovered'].astype(int)

full_table.sample(6)
```

Out[10]:

	Province/State	Country/Region	Lat	Long	Date	Confirmed	Deaths	Recovered	Active
15631	Guangxi	China	23.8298	108.7881	2020-03-21	254	2	250	2
5560		Austria	47.5162	14.5501	2020-02-12	0	0	0	0
26474	Shanghai	China	31.2020	121.4491	2020-05-01	652	7	0	645
5177		Mongolia	46.8625	103.8467	2020-02-10	0	0	0	0
12093		Turkey	38.9637	35.2433	2020-03-07	0	0	0	0
9342		Estonia	58.5953	25.0136	2020-02-26	0	0	0	0

In [11]:
```
# Grouped by day, country
# =======================

full_grouped = full_table.groupby(['Date', 'Country/Region'])['Confirmed', 'Deaths', 'Recovered', 'Active'].sum().reset_index()

# new cases ======================================================
temp = full_grouped.groupby(['Country/Region', 'Date', ])['Confirmed', 'Deaths', 'Recovered']
temp = temp.sum().diff().reset_index()

mask = temp['Country/Region'] != temp['Country/Region'].shift(1)

temp.loc[mask, 'Confirmed'] = np.nan
temp.loc[mask, 'Deaths'] = np.nan
temp.loc[mask, 'Recovered'] = np.nan

# renaming columns
temp.columns = ['Country/Region', 'Date', 'New cases', 'New deaths', 'New recovered']
# ================================================================

# merging new values
full_grouped = pd.merge(full_grouped, temp, on=['Country/Region', 'Date'])

# filling na with 0
full_grouped = full_grouped.fillna(0)

# fixing data types
cols = ['New cases', 'New deaths', 'New recovered']
full_grouped[cols] = full_grouped[cols].astype('int')

full_grouped['New cases'] = full_grouped['New cases'].apply(lambda x: 0 if x<0 else x)

full_grouped.head()
```

Out[11]:

	Date	Country/Region	Confirmed	Deaths	Recovered	Active	New cases	New deaths	New recovered
0	2020-01-22	Afghanistan	0	0	0	0	0	0	0
1	2020-01-22	Albania	0	0	0	0	0	0	0
2	2020-01-22	Algeria	0	0	0	0	0	0	0
3	2020-01-22	Andorra	0	0	0	0	0	0	0
4	2020-01-22	Angola	0	0	0	0	0	0	0

In [12]:
```
# Day wise
# ========

# table
day_wise = full_grouped.groupby('Date')['Confirmed', 'Deaths', 'Recovered', 'Active', 'New cases'].sum().reset_index()

# number cases per 100 cases
day_wise['Deaths / 100 Cases'] = round((day_wise['Deaths']/day_wise['Confirmed'])*100, 2)
day_wise['Recovered / 100 Cases'] = round((day_wise['Recovered']/day_wise['Confirmed'])*100, 2)
day_wise['Deaths / 100 Recovered'] = round((day_wise['Deaths']/day_wise['Recovered'])*100, 2)

# no. of countries
day_wise['No. of countries'] = full_grouped[full_grouped['Confirmed']!=0].groupby('Date')['Country/Region'].unique().apply(len).values

# fillna by 0
cols = ['Deaths / 100 Cases', 'Recovered / 100 Cases', 'Deaths / 100 Recovered']
day_wise[cols] = day_wise[cols].fillna(0)

day_wise.head()
```

Out[12]:

	Date	Confirmed	Deaths	Recovered	Active	New cases	Deaths / 100 Cases	Recovered / 100 Cases	Deaths / 100 Recovered	No. of countries
0	2020-01-22	555	17	28	510	0	3.06	5.05	60.71	6
1	2020-01-23	654	18	30	606	99	2.75	4.59	60.00	8
2	2020-01-24	941	26	35	880	287	2.76	3.72	74.29	9
3	2020-01-25	1434	42	38	1354	493	2.93	2.65	110.53	11
4	2020-01-26	2118	56	51	2011	684	2.64	2.41	109.80	13

In [13]:
```
# Country wise
# ============

# getting latest values
country_wise = full_grouped[full_grouped['Date']==max(full_grouped['Date'])].reset_index(drop=True).drop('Date', axis=1)

# group by country
country_wise = country_wise.groupby('Country/Region')['Confirmed', 'Deaths', 'Recovered', 'Active', 'New cases'].sum().reset_index()

# per 100 cases
country_wise['Deaths / 100 Cases'] = round((country_wise['Deaths']/country_wise['Confirmed'])*100, 2)
country_wise['Recovered / 100 Cases'] = round((country_wise['Recovered']/country_wise['Confirmed'])*100, 2)
country_wise['Deaths / 100 Recovered'] = round((country_wise['Deaths']/country_wise['Recovered'])*100, 2)

cols = ['Deaths / 100 Cases', 'Recovered / 100 Cases', 'Deaths / 100 Recovered']
country_wise[cols] = country_wise[cols].fillna(0)

country_wise.head()
```

Out[13]:

	Country/Region	Confirmed	Deaths	Recovered	Active	New cases	Deaths / 100 Cases	Recovered / 100 Cases	Deaths / 100 Recovered
0	Afghanistan	2704	85	345	2274	235	3.14	12.76	24.64
1	Albania	795	31	531	233	6	3.90	66.79	5.84
2	Algeria	4474	463	1936	2075	179	10.35	43.27	23.92
3	Andorra	748	45	493	210	1	6.02	65.91	9.13
4	Angola	35	2	11	22	0	5.71	31.43	18.18

In [14]:
```
# load population dataset
pop = pd.read_csv("./corona-virus-report/Covid/population_by_country_2020.csv")

# select only population
pop = pop.iloc[:, :2]

# rename column names
pop.columns = ['Country/Region', 'Population']

# merged data
country_wise = pd.merge(country_wise, pop, on='Country/Region', how='left')

# update population
cols = ['Burma', 'Congo (Brazzaville)', 'Congo (Kinshasa)', "Cote d'Ivoire", 'Czechia',
        'Kosovo', 'Saint Kitts and Nevis', 'Saint Vincent and the Grenadines',
        'Taiwan*', 'US', 'West Bank and Gaza']
pops = [54409800, 89561403, 5518087, 26378274, 10708981, 1793000,
        53109, 110854, 23806638, 330541757, 4543126]
for c, p in zip(cols, pops):
    country_wise.loc[country_wise['Country/Region']== c, 'Population'] = p

# missing values
# country_wise.isna().sum()
# country_wise[country_wise['Population'].isna()]['Country/Region'].tolist()

# Cases per population
country_wise['Cases / Million People'] = round((country_wise['Confirmed'] / country_wise['Population']) * 1000000)

country_wise.head()
```

Out[14]:

	Country/Region	Confirmed	Deaths	Recovered	Active	New cases	Deaths / 100 Cases	Recovered / 100 Cases	Deaths / 100 Recovered	Population	Cases / Million People
0	Afghanistan	2704	85	345	2274	235	3.14	12.76	24.64	38742911.0	70.0
1	Albania	795	31	531	233	6	3.90	66.79	5.84	2878420.0	276.0
2	Algeria	4474	463	1936	2075	179	10.35	43.27	23.92	43685618.0	102.0
3	Andorra	748	45	493	210	1	6.02	65.91	9.13	77240.0	9684.0
4	Angola	35	2	11	22	0	5.71	31.43	18.18	32644783.0	1.0

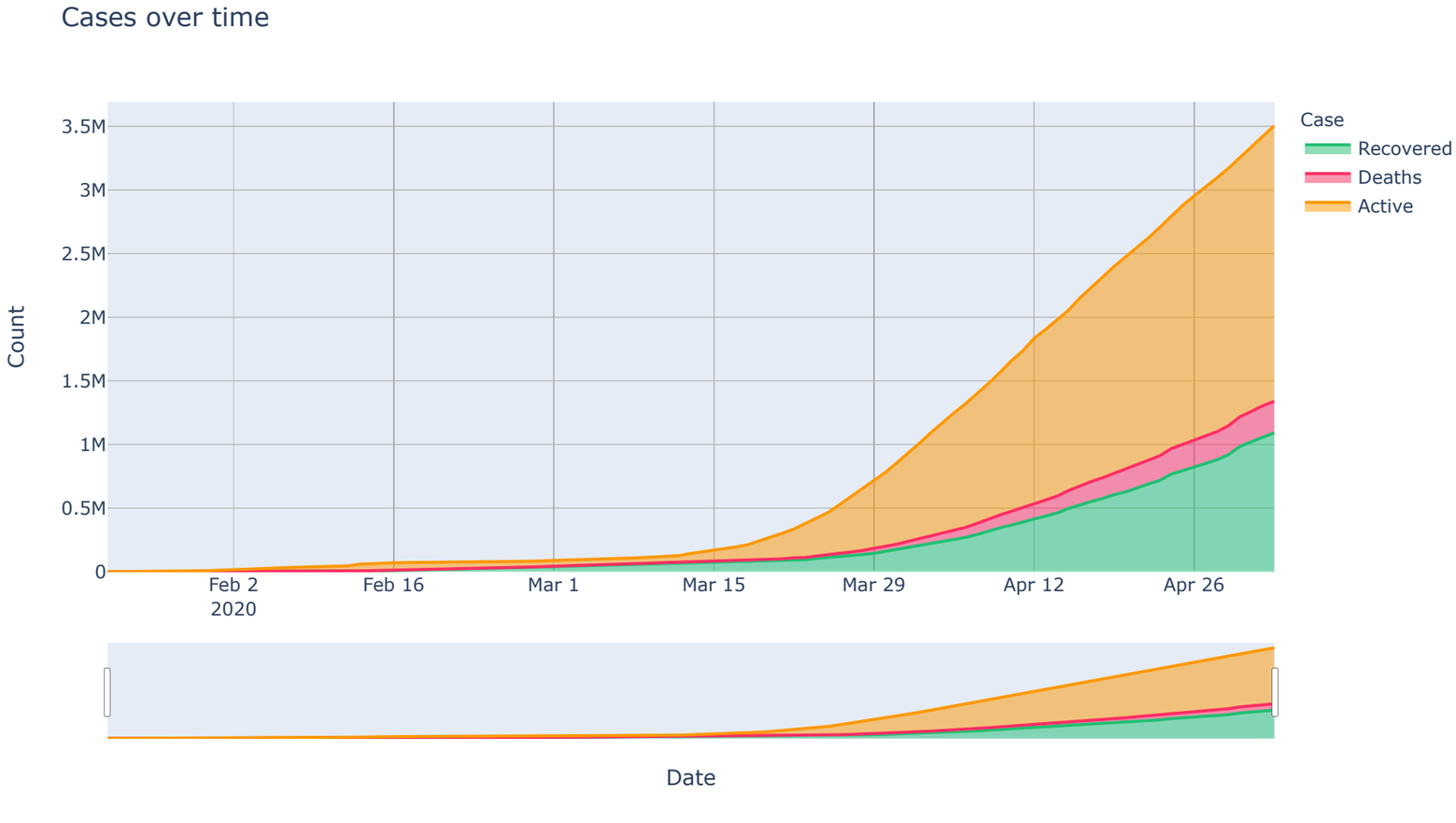
In [15]:
```
today = full_grouped[full_grouped['Date']==max(full_grouped['Date'])].reset_index(drop=True).drop('Date', axis=1)[['Country/Region', 'Confirmed']]
last_week = full_grouped[full_grouped['Date']==max(full_grouped['Date'])-timedelta(days=7)].reset_index(drop=True).drop('Date', axis=1)[['Country/Region', 'Confirmed']]

temp = pd.merge(today, last_week, on='Country/Region', suffixes=(' today', ' last week'))

# temp = temp[['Country/Region', 'Confirmed last week']]
temp['1 week change'] = temp['Confirmed today'] - temp['Confirmed last week']

temp = temp[['Country/Region', 'Confirmed last week', '1 week change']]

country_wise = pd.merge(country_wise, temp, on='Country/Region')

country_wise['1 week % increase'] = round(country_wise['1 week change']/country_wise['Confirmed last week']*100, 2)

country_wise.head()
```

Out[15]:

	Country/Region	Confirmed	Deaths	Recovered	Active	New cases	Deaths / 100 Cases	Recovered / 100 Cases	Deaths / 100 Recovered	Population	Cases / Million People	Confirmed last week	1 week change	1 week % increase
0	Afghanistan	2704	85	345	2274	235	3.14	12.76	24.64	38742911.0	70.0	1531	1173	76.62
1	Albania	795	31	531	233	6	3.90	66.79	5.84	2878420.0	276.0	726	69	9.50
2	Algeria	4474	463	1936	2075	179	10.35	43.27	23.92	43685618.0	102.0	3382	1092	32.29
3	Andorra	748	45	493	210	1	6.02	65.91	9.13	77240.0	9684.0	738	10	1.36
4	Angola	35	2	11	22	0	5.71	31.43	18.18	32644783.0	1.0	26	9	34.62

In [16]:
```
temp = full_table.groupby('Date')['Confirmed', 'Deaths', 'Recovered', 'Active'].sum().reset_index()
temp = temp[temp['Date']==max(temp['Date'])].reset_index(drop=True)

tm = temp.melt(id_vars="Date", value_vars=['Active', 'Deaths', 'Recovered'])
fig = px.treemap(tm, path=["variable"], values="value", height=225, width=1200,
                 color_discrete_sequence=[act, rec, dth])
fig.data[0].textinfo = 'label+text+value'
fig.show()
```

In [17]:
```
temp = full_table.groupby('Date')['Recovered', 'Deaths', 'Active'].sum().reset_index()
temp = temp.melt(id_vars="Date", value_vars=['Recovered', 'Deaths', 'Active'],
                 var_name='Case', value_name='Count')
temp.head()

fig = px.area(temp, x="Date", y="Count", color='Case', height=600,
              title='Cases over time', color_discrete_sequence = [rec, dth, act])
fig.update_layout(xaxis_rangeslider_visible=True)
fig.show()
```



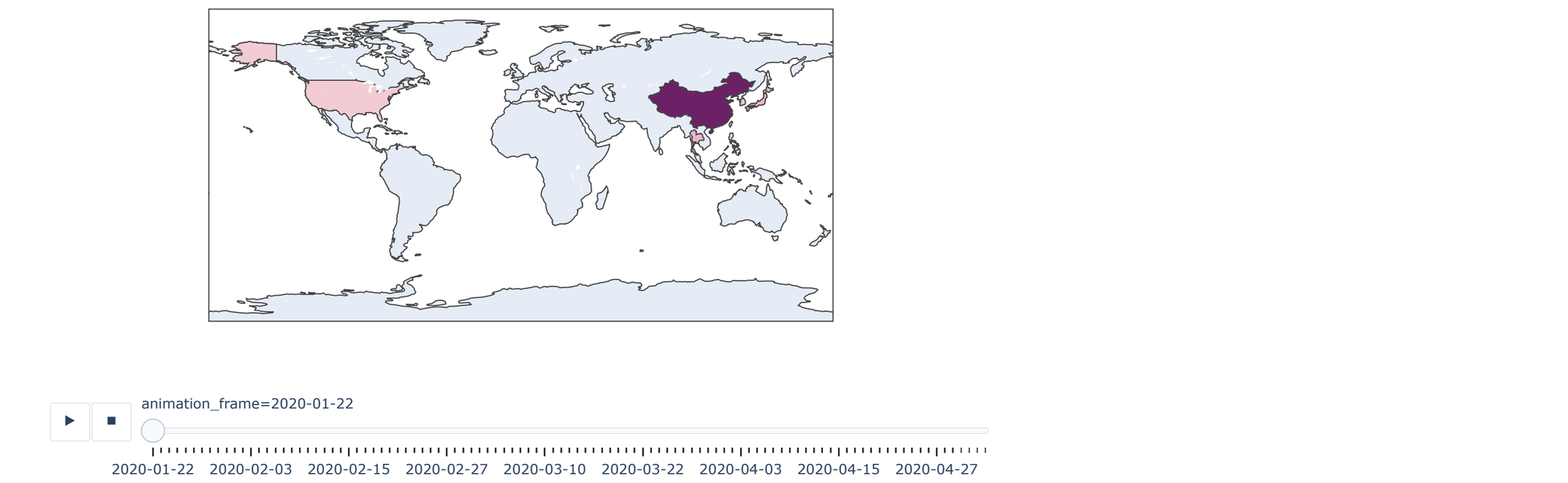
Maps

Across the world

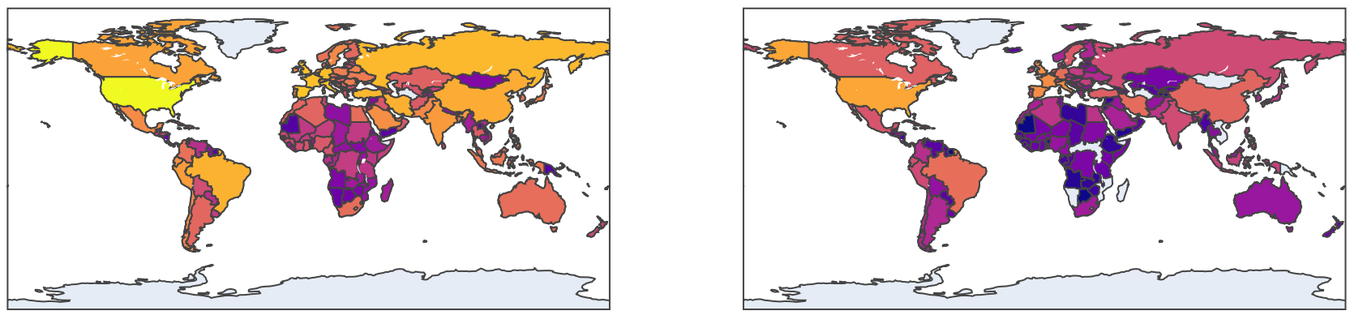
In [18]:
```
# World wide

temp = full_table[full_table['Date'] == max(full_table['Date'])]

m = folium.Map(location=[0, 0], tiles='cartodbpositron',
               min_zoom=1, max_zoom=4, zoom_start=1)

for i in range(0, len(temp)):
    folium.Circle(
        location=[temp.iloc[i]['Lat'], temp.iloc[i]['Long']],
        color='crimson', fill='crimson',
        tooltip =   '<li><bold>Country : '+str(temp.iloc[i]['Country/Region'])+
                    '<li><bold>Province : '+str(temp.iloc[i]['Province/State'])+
                    '<li><bold>Confirmed : '+str(temp.iloc[i]['Confirmed'])+
                    '<li><bold>Deaths : '+str(temp.iloc[i]['Deaths']),
        radius=int(temp.iloc[i]['Confirmed'])**1.1).add_to(m)
m
```

Out[18]:

In [19]:
```
# Over the time

fig = px.choropleth(full_grouped, locations="Country/Region", locationmode='country names', color=np.log(full_grouped["Confirmed"]),
                    hover_name="Country/Region", animation_frame=full_grouped["Date"].dt.strftime('%Y-%m-%d'),
                    title='Cases over time', color_continuous_scale=px.colors.sequential.Magenta)
fig.update(layout_coloraxis_showscale=False)
fig.show()
```



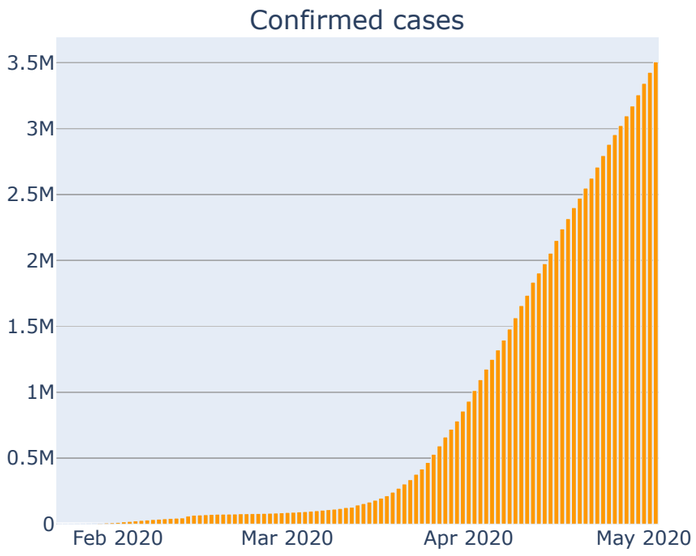
In [20]:
```
# Confirmed
fig_c = px.choropleth(country_wise, locations="Country/Region", locationmode='country names',
                      color=np.log(country_wise["Confirmed"]), hover_name="Country/Region", hover_data=['Confirmed'])

# Deaths
temp = country_wise[country_wise['Deaths']>0]
fig_d = px.choropleth(temp, locations="Country/Region", locationmode='country names',
                      color=np.log(temp["Deaths"]), hover_name="Country/Region", hover_data=['Deaths'])

# Plot
fig = make_subplots(rows=1, cols=2, subplot_titles = ['Confirmed', 'Deaths'],
                    specs=[[{"type": "choropleth"}, {"type": "choropleth"}]])

fig.add_trace(fig_c['data'][0], row=1, col=1)
fig.add_trace(fig_d['data'][0], row=1, col=2)

fig.update(layout_coloraxis_showscale=False)

fig.show()
```

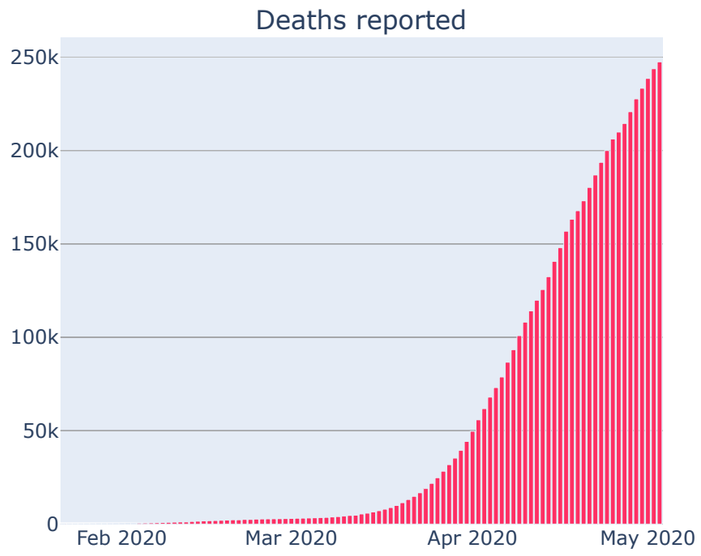


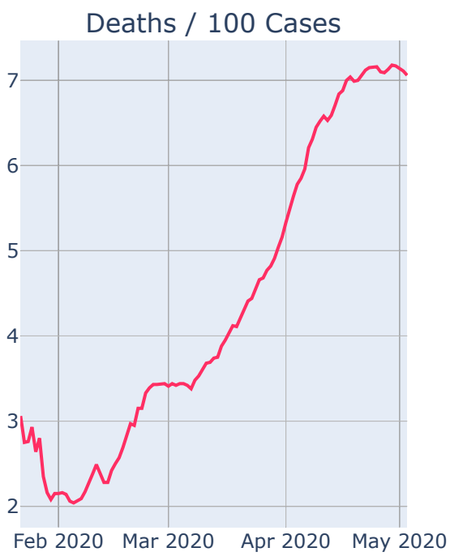
Cases over the time

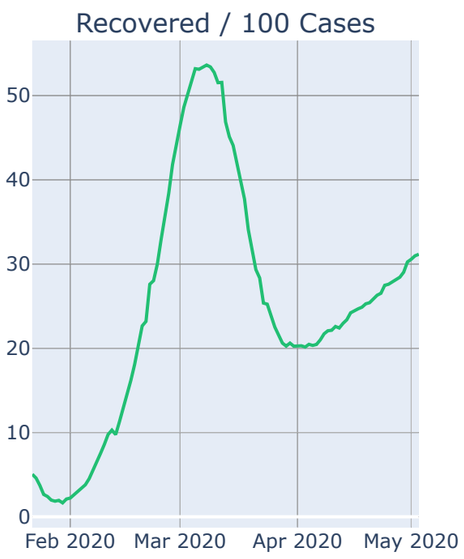
In [21]:
```
fig_c = px.bar(day_wise, x="Date", y="Confirmed", color_discrete_sequence = [act])
fig_d = px.bar(day_wise, x="Date", y="Deaths", color_discrete_sequence = [dth])

fig = make_subplots(rows=1, cols=2, shared_xaxes=False, horizontal_spacing=0.1,
                    subplot_titles=('Confirmed cases', 'Deaths reported'))

fig.add_trace(fig_c['data'][0], row=1, col=1)
fig.add_trace(fig_d['data'][0], row=1, col=2)

fig.update_layout(height=480)
fig.show()

# ================================

fig_1 = px.line(day_wise, x="Date", y="Deaths / 100 Cases", color_discrete_sequence = [dth])
fig_2 = px.line(day_wise, x="Date", y="Recovered / 100 Cases", color_discrete_sequence = [rec])
fig_3 = px.line(day_wise, x="Date", y="Deaths / 100 Recovered", color_discrete_sequence = ['#333333'])

fig = make_subplots(rows=1, cols=3, shared_xaxes=False,
                    subplot_titles=('Deaths / 100 Cases', 'Recovered / 100 Cases', 'Deaths / 100 Recovered'))

fig.add_trace(fig_1['data'][0], row=1, col=1)
fig.add_trace(fig_2['data'][0], row=1, col=2)
fig.add_trace(fig_3['data'][0], row=1, col=3)

fig.update_layout(height=480)
fig.show()

# ==================================
```

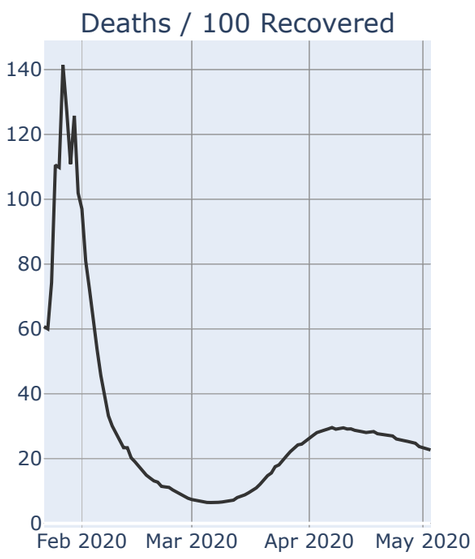
```
fig_c = px.bar(day_wise, x="Date", y="New cases", color_discrete_sequence = [act])
fig_d = px.bar(day_wise, x="Date", y="No. of countries", color_discrete_sequence = [dth])

fig = make_subplots(rows=1, cols=2, shared_xaxes=False, horizontal_spacing=0.1,
                    subplot_titles=('No. of new cases everyday', 'No. of countries'))

fig.add_trace(fig_c['data'][0], row=1, col=1)
fig.add_trace(fig_d['data'][0], row=1, col=2)

fig.update_layout(height=480)
fig.show()
```

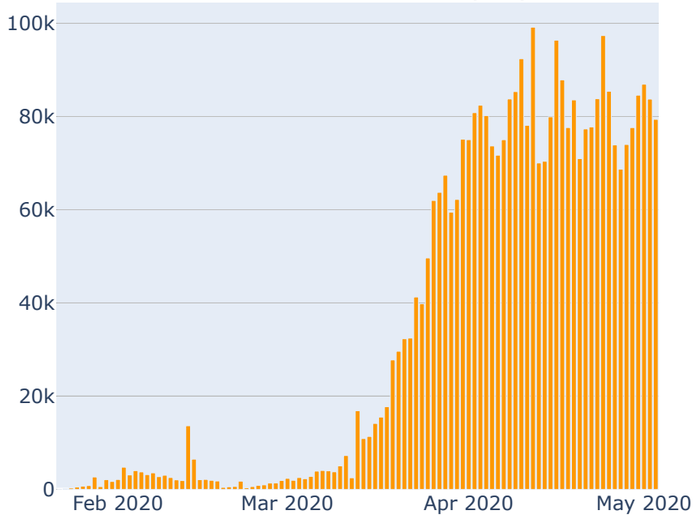


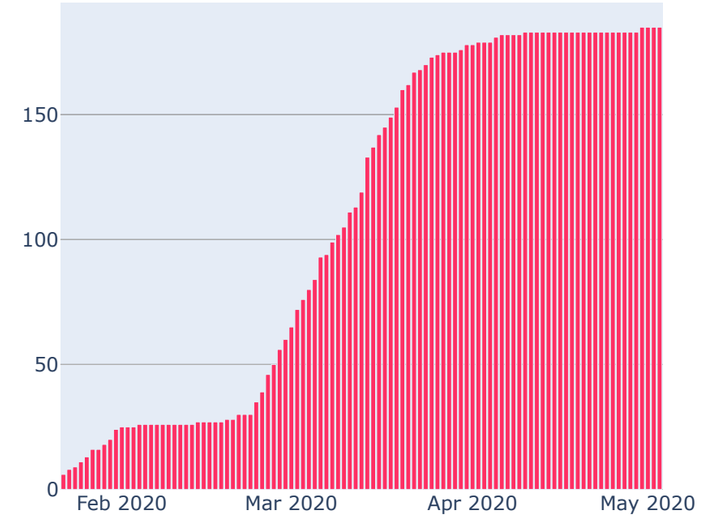






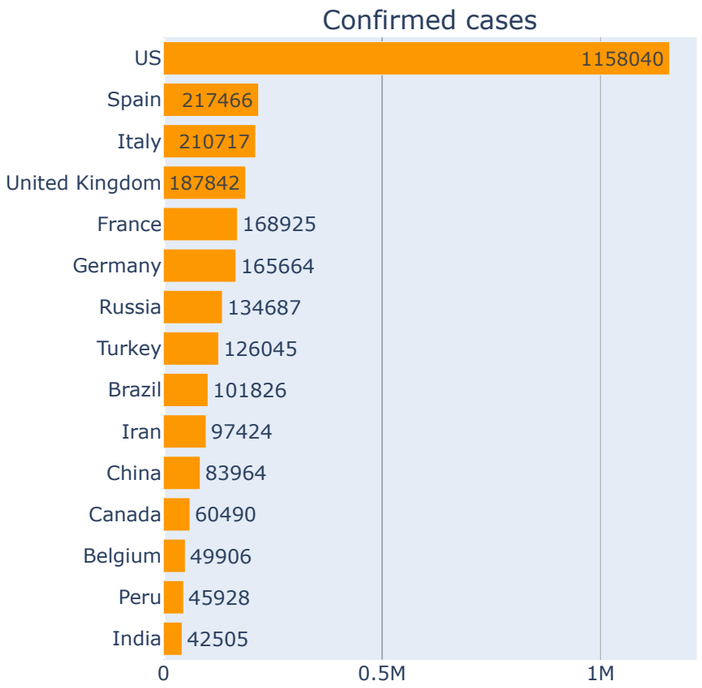


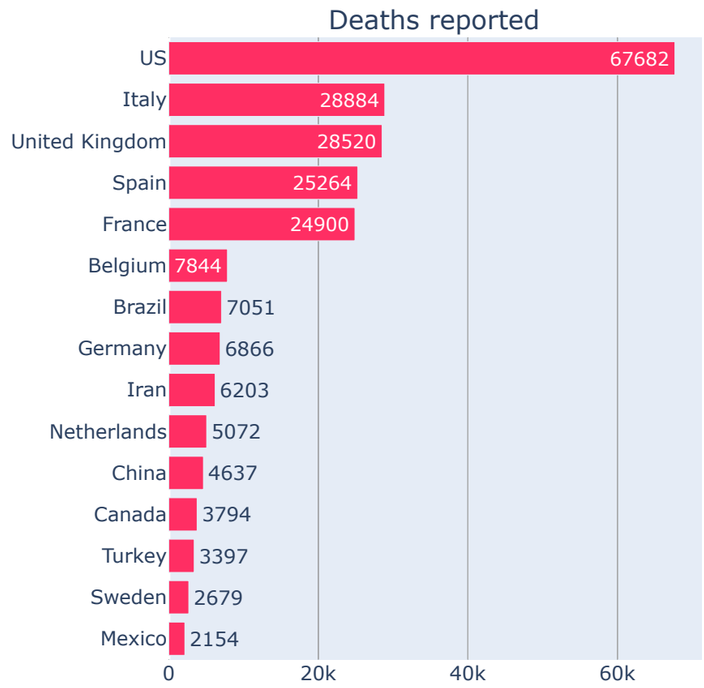


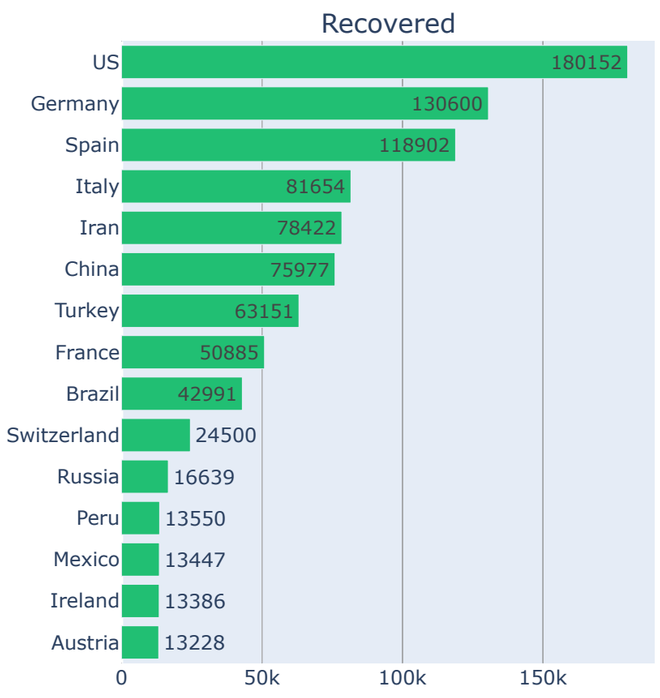


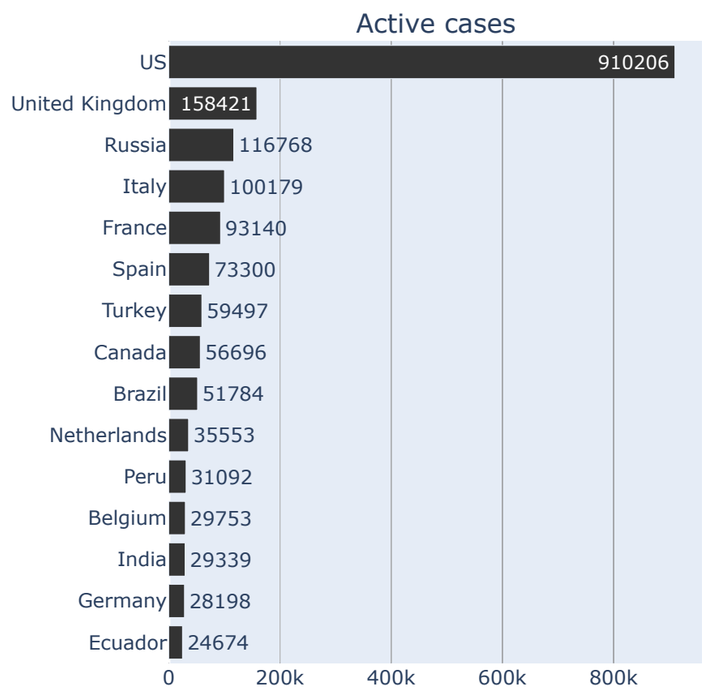
Top 20 Countries

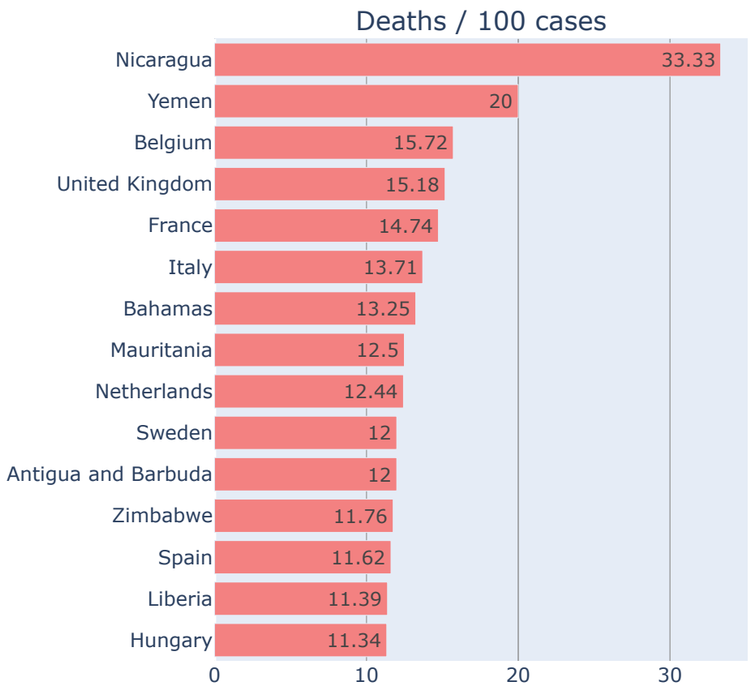
In [22]:
```
# confirmed - deaths
fig_c = px.bar(country_wise.sort_values('Confirmed').tail(15), x="Confirmed", y="Country/Region",
               text='Confirmed', orientation='h', color_discrete_sequence = [act])
fig_d = px.bar(country_wise.sort_values('Deaths').tail(15), x="Deaths", y="Country/Region",
               text='Deaths', orientation='h', color_discrete_sequence = [dth])

# recovered - active
fig_r = px.bar(country_wise.sort_values('Recovered').tail(15), x="Recovered", y="Country/Region",
               text='Recovered', orientation='h', color_discrete_sequence = [rec])
fig_a = px.bar(country_wise.sort_values('Active').tail(15), x="Active", y="Country/Region",
               text='Active', orientation='h', color_discrete_sequence = ['#333333'])

# death - recoverd / 100 cases
fig_dc = px.bar(country_wise.sort_values('Deaths / 100 Cases').tail(15), x="Deaths / 100 Cases", y="Country/Region",
               text='Deaths / 100 Cases', orientation='h', color_discrete_sequence = ['#f38181'])
fig_rc = px.bar(country_wise.sort_values('Recovered / 100 Cases').tail(15), x="Recovered / 100 Cases", y="Country/Region",
               text='Recovered / 100 Cases', orientation='h', color_discrete_sequence = ['#a3de83'])

# new cases - cases per million people
fig_nc = px.bar(country_wise.sort_values('New cases').tail(15), x="New cases", y="Country/Region",
               text='New cases', orientation='h', color_discrete_sequence = ['#c61951'])
temp = country_wise[country_wise['Population']>1000000]
fig_p = px.bar(temp.sort_values('Cases / Million People').tail(15), x="Cases / Million People", y="Country/Region",
               text='Cases / Million People', orientation='h', color_discrete_sequence = ['#741938'])

# week change, percent increase
fig_wc = px.bar(country_wise.sort_values('1 week change').tail(15), x="1 week change", y="Country/Region",
               text='1 week change', orientation='h', color_discrete_sequence = ['#004a7c'])
temp = country_wise[country_wise['Confirmed']>100]
fig_pi = px.bar(temp.sort_values('1 week % increase').tail(15), x="1 week % increase", y="Country/Region",
               text='1 week % increase', orientation='h', color_discrete_sequence = ['#005691'],
                hover_data=['Confirmed last week', 'Confirmed'])


# plot
fig = make_subplots(rows=5, cols=2, shared_xaxes=False, horizontal_spacing=0.14, vertical_spacing=0.08,
                    subplot_titles=('Confirmed cases', 'Deaths reported', 'Recovered', 'Active cases',
                                    'Deaths / 100 cases', 'Recovered / 100 cases', 'New cases',
                                    'Cases / Million People', '1 week increase', '1 week % increase'))

fig.add_trace(fig_c['data'][0], row=1, col=1)
fig.add_trace(fig_d['data'][0], row=1, col=2)
fig.add_trace(fig_r['data'][0], row=2, col=1)
fig.add_trace(fig_a['data'][0], row=2, col=2)

fig.add_trace(fig_dc['data'][0], row=3, col=1)
fig.add_trace(fig_rc['data'][0], row=3, col=2)
fig.add_trace(fig_nc['data'][0], row=4, col=1)
fig.add_trace(fig_p['data'][0], row=4, col=2)

fig.add_trace(fig_wc['data'][0], row=5, col=1)
fig.add_trace(fig_pi['data'][0], row=5, col=2)


fig.update_layout(height=3000)
```

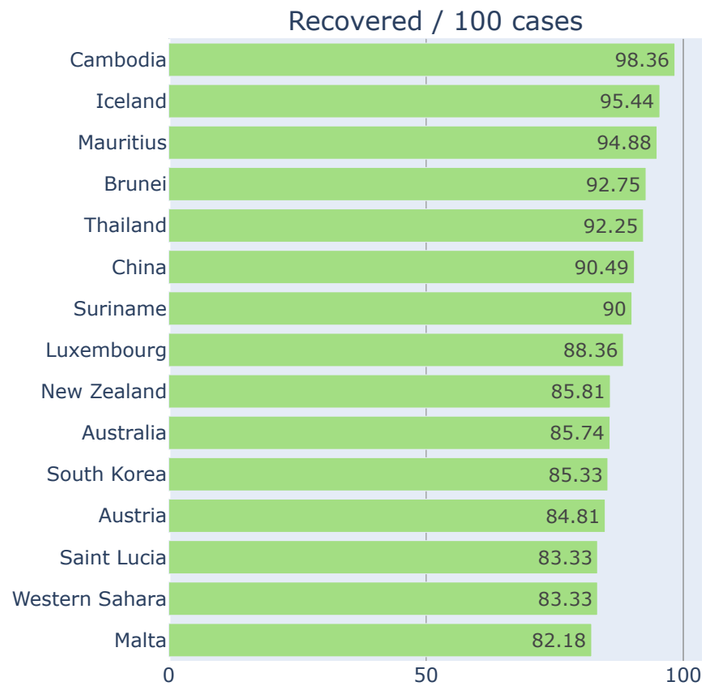

Confirmed cases
US 1158040
Spain 217466
Italy 210717
United Kingdom 187842
France 168925
Germany 165664
Russia 134687
Turkey 126045
Brazil 101826
Iran 97424
China 83964
Canada 60490
Belgium 49906
Peru 45928
India 42505
0
0.5M
1M

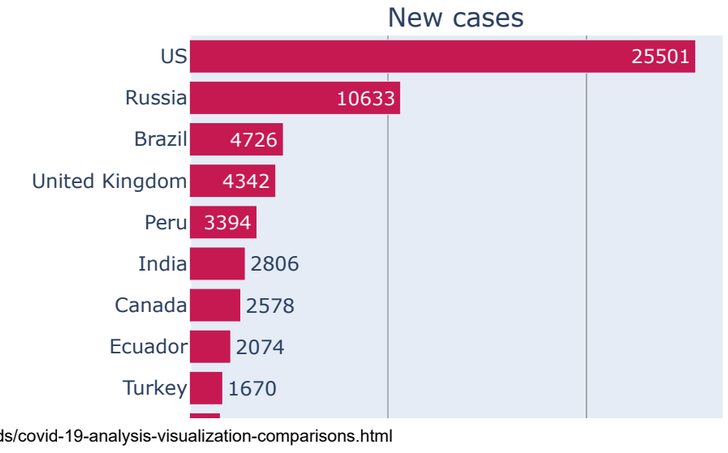

Deaths reported
US 67682
Italy 28884
United Kingdom 28520
Spain 25264
France 24900
Belgium 7844
Brazil 7051
Germany 6866
Iran 6203
Netherlands 5072
China 4637
Canada 3794
Turkey 3397
Sweden 2679
Mexico 2154
0
20k
40k
60k

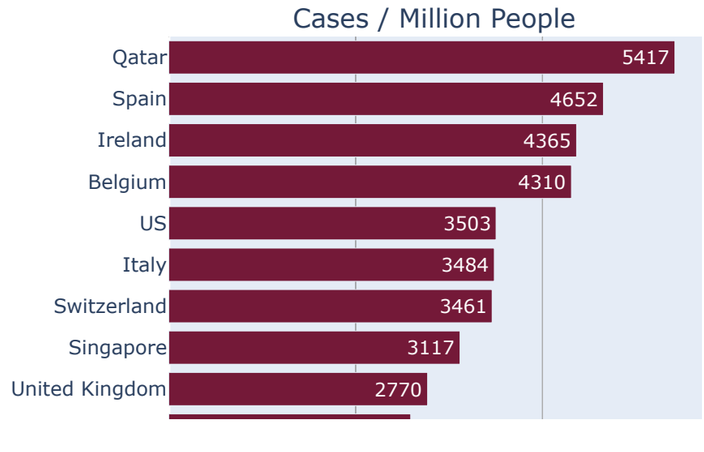

Recovered
US 180152
Germany 130600
Spain 118902
Italy 81654
Iran 78422
China 75977
Turkey 63151
France 50885
Brazil 42991
Switzerland 24500
Russia 16639
Peru 13550
Mexico 13447
Ireland 13386
Austria 13228
0
50k
100k
150k

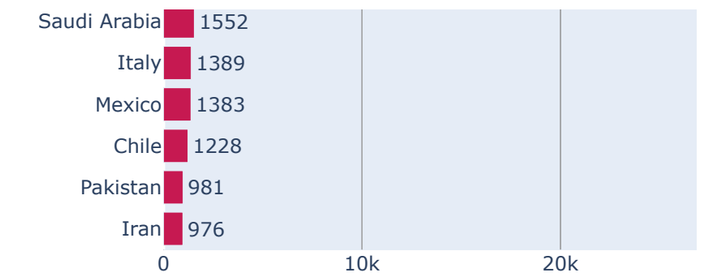

Active cases
US 910206
United Kingdom 158421
Russia 116768
Italy 100179
France 93140
Spain 73300
Turkey 59497
Canada 56696
Brazil 51784
Netherlands 35553
Peru 31092
Belgium 29753
India 29339
Germany 28198
Ecuador 24674
0
200k
400k
600k
800k

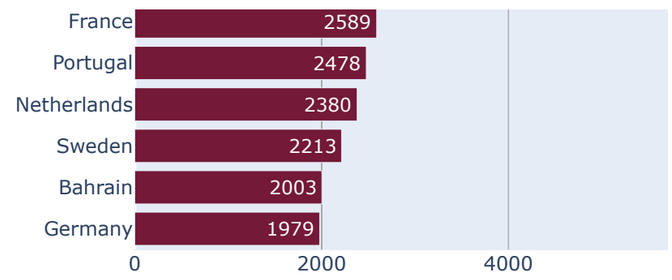

Deaths / 100 cases
Nicaragua 33.33
Yemen 20
Belgium 15.72
United Kingdom 15.18
France 14.74
Italy 13.71
Bahamas 13.25
Mauritania 12.5
Netherlands 12.44
Sweden 12
Antigua and Barbuda 12
Zimbabwe 11.76
Spain 11.62
Liberia 11.39
Hungary 11.34
0
10
20
30

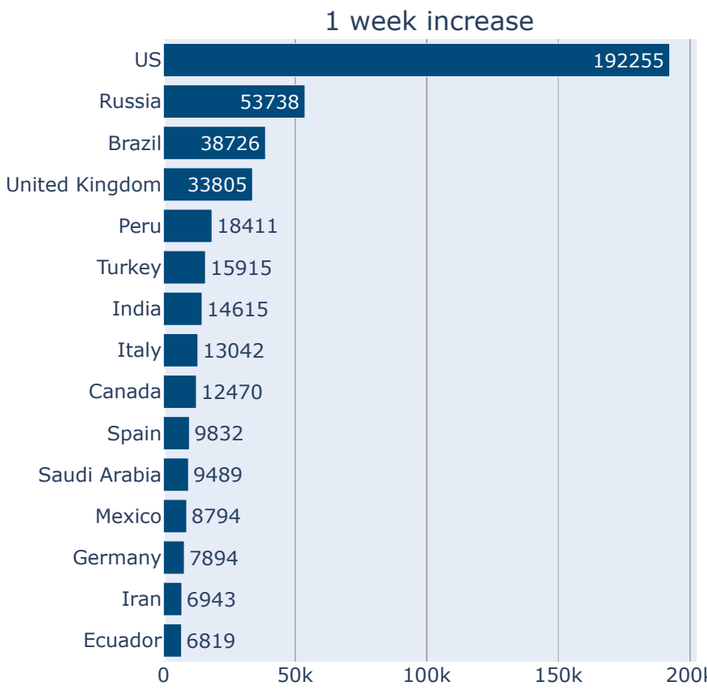

Recovered / 100 cases
Cambodia 98.36
Iceland 95.44
Mauritius 94.88
Brunei 92.75
Thailand 92.25
China 90.49
Suriname 90
Luxembourg 88.36
New Zealand 85.81
Australia 85.74
South Korea 85.33
Austria 84.81
Saint Lucia 83.33
Western Sahara 83.33
Malta 82.18
0
50
100

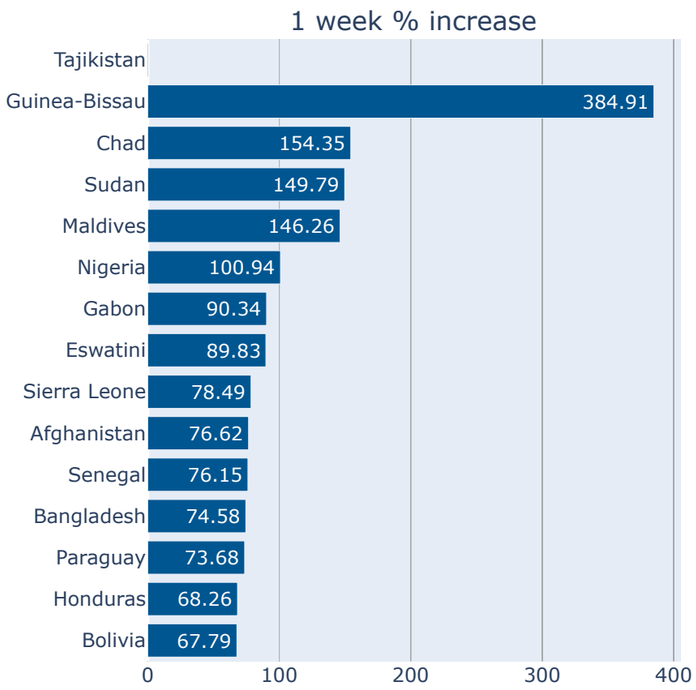

New cases
US 25501
Russia 10633
Brazil 4726
United Kingdom 4342
Peru 3394
India 2806
Canada 2578
Ecuador 2074
Turkey 1670


Cases / Million People
Qatar 5417
Spain 4652
Ireland 4365
Belgium 4310
US 3503
Italy 3484
Switzerland 3461
Singapore 3117
United Kingdom 2770

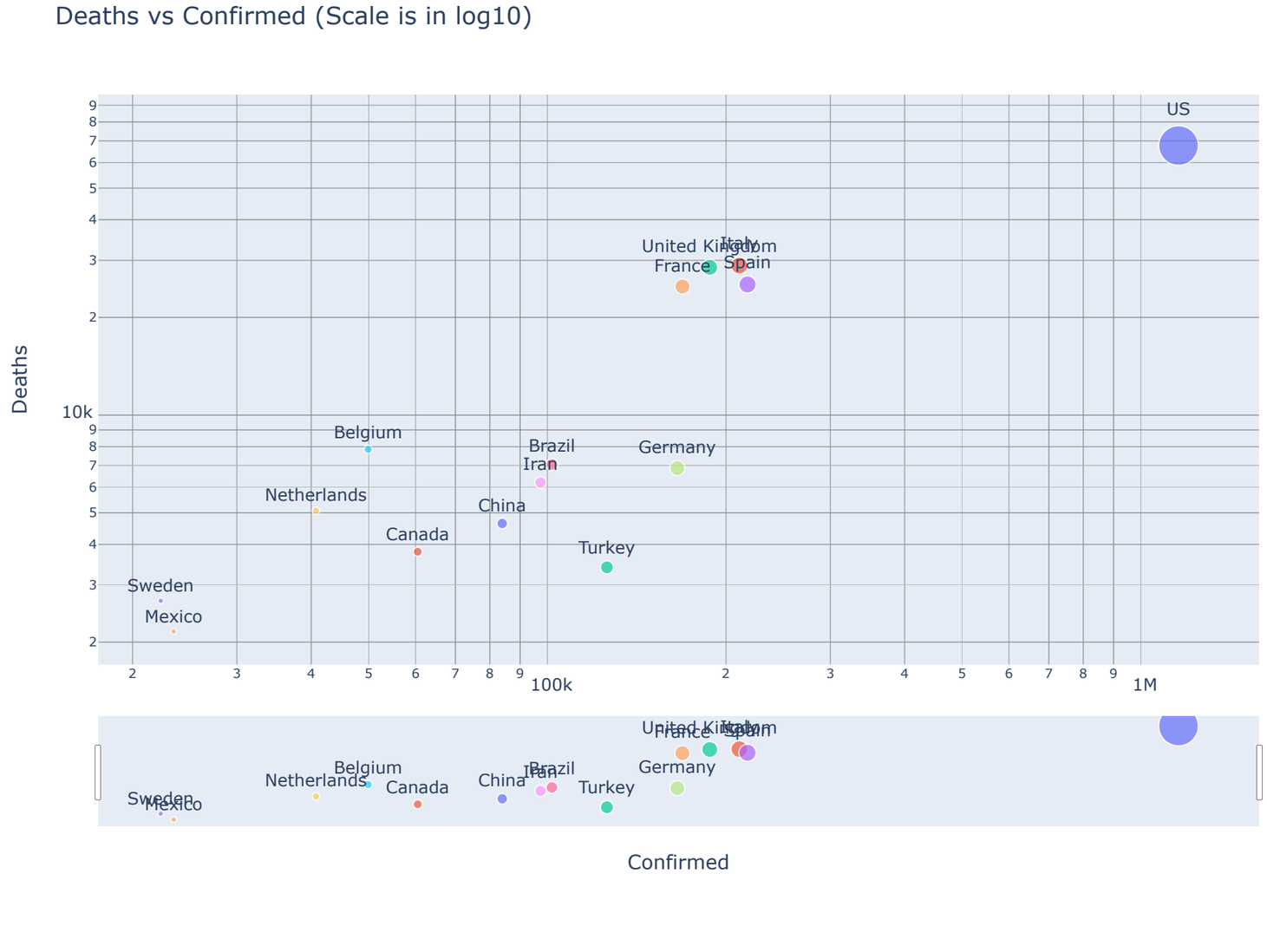






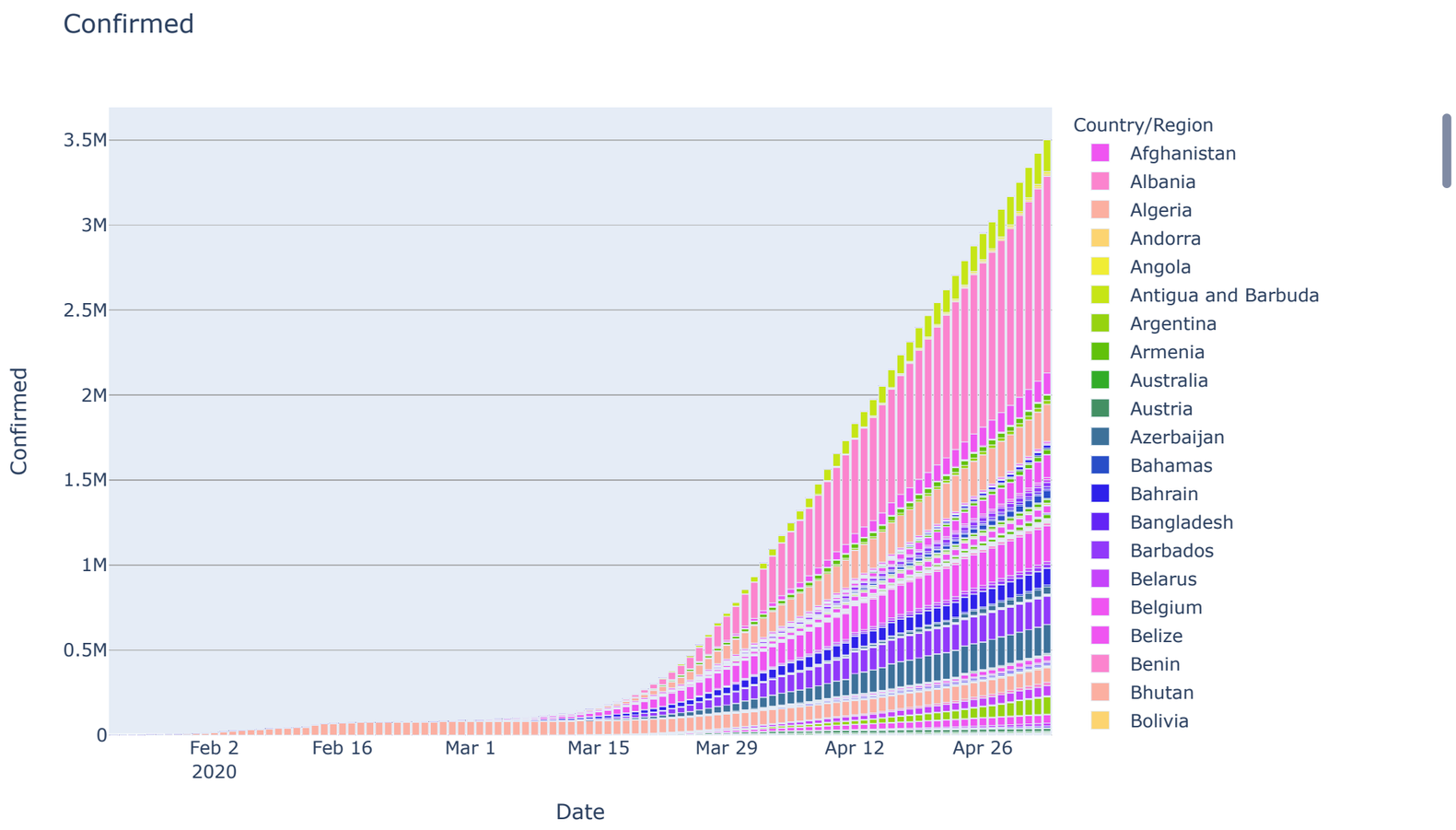


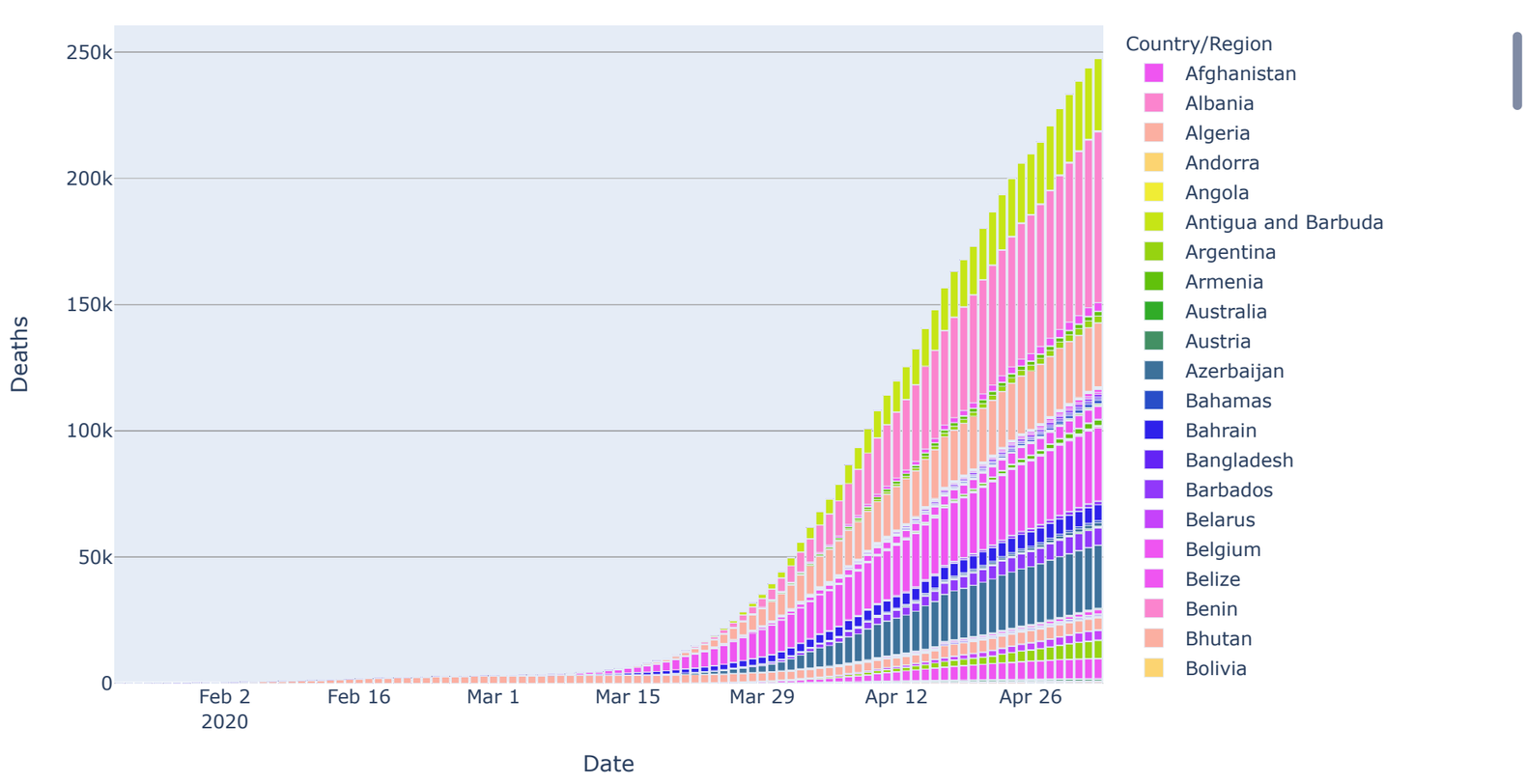
In [23]:
```
fig = px.scatter(country_wise.sort_values('Deaths', ascending=False).iloc[:15, :],
                 x='Confirmed', y='Deaths', color='Country/Region', size='Confirmed', height=700,
                 text='Country/Region', log_x=True, log_y=True, title='Deaths vs Confirmed (Scale is in log10)')
fig.update_traces(textposition='top center')
fig.update_layout(showlegend=False)
fig.update_layout(xaxis_rangeslider_visible=True)
fig.show()
```

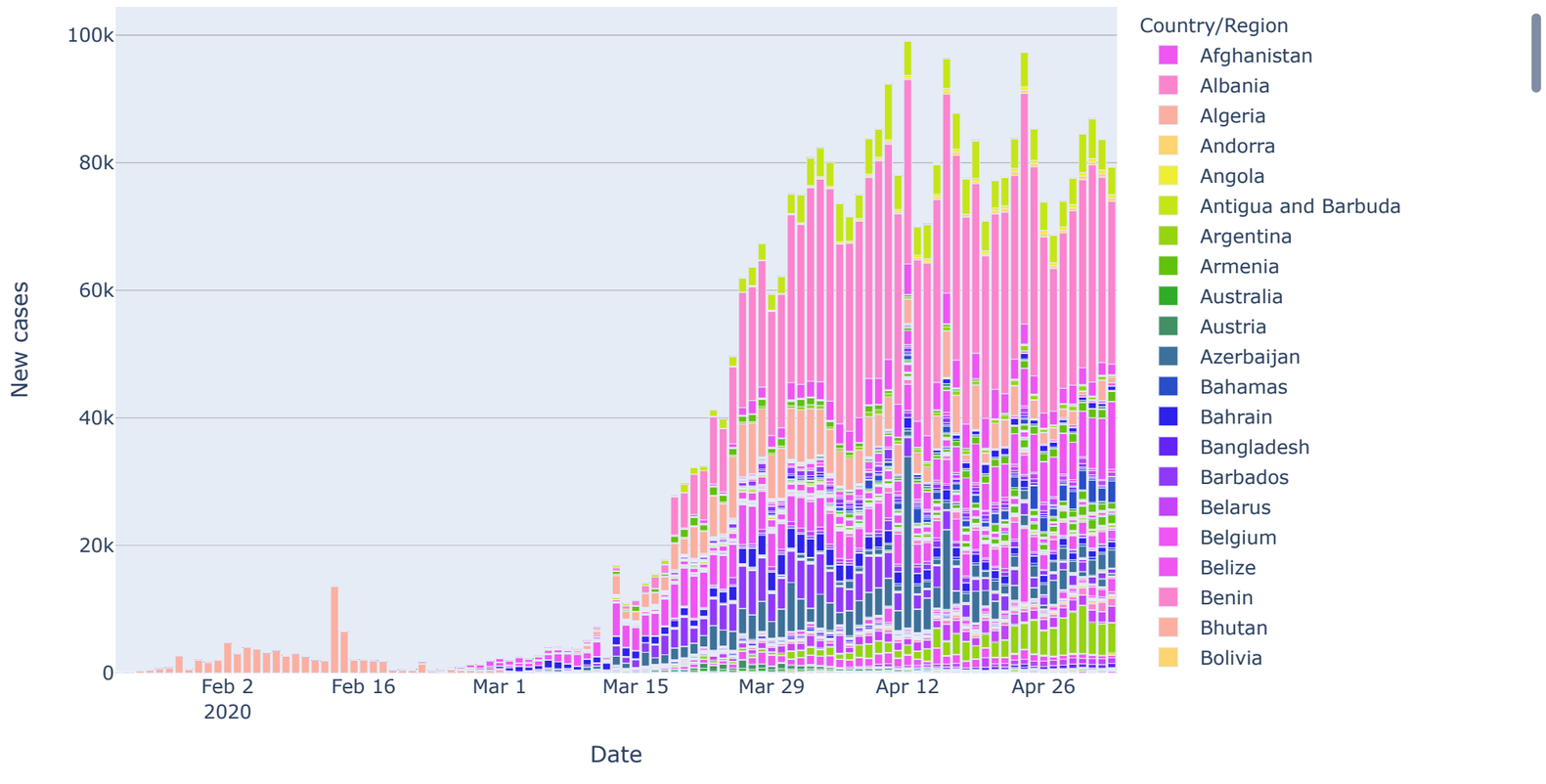


Date vs

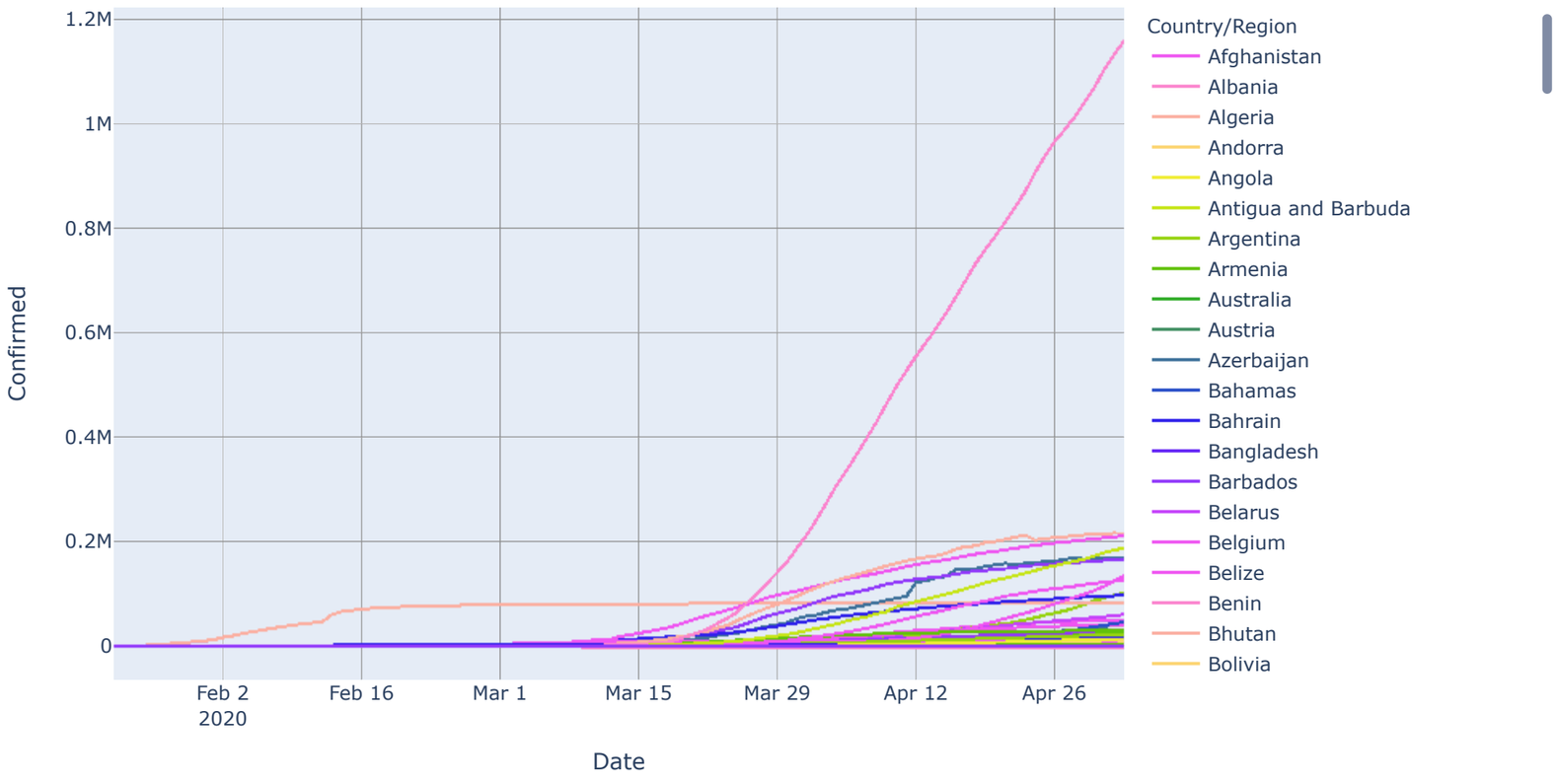
In [24]:
```
fig = px.bar(full_grouped, x="Date", y="Confirmed", color='Country/Region', height=600,
             title='Confirmed', color_discrete_sequence = px.colors.cyclical.mygbm)
fig.show()

# ========================================

fig = px.bar(full_grouped, x="Date", y="Deaths", color='Country/Region', height=600,
             title='Deaths', color_discrete_sequence = px.colors.cyclical.mygbm)
fig.show()

# ========================================

fig = px.bar(full_grouped, x="Date", y="New cases", color='Country/Region', height=600,
             title='New cases', color_discrete_sequence = px.colors.cyclical.mygbm)
fig.show()
```

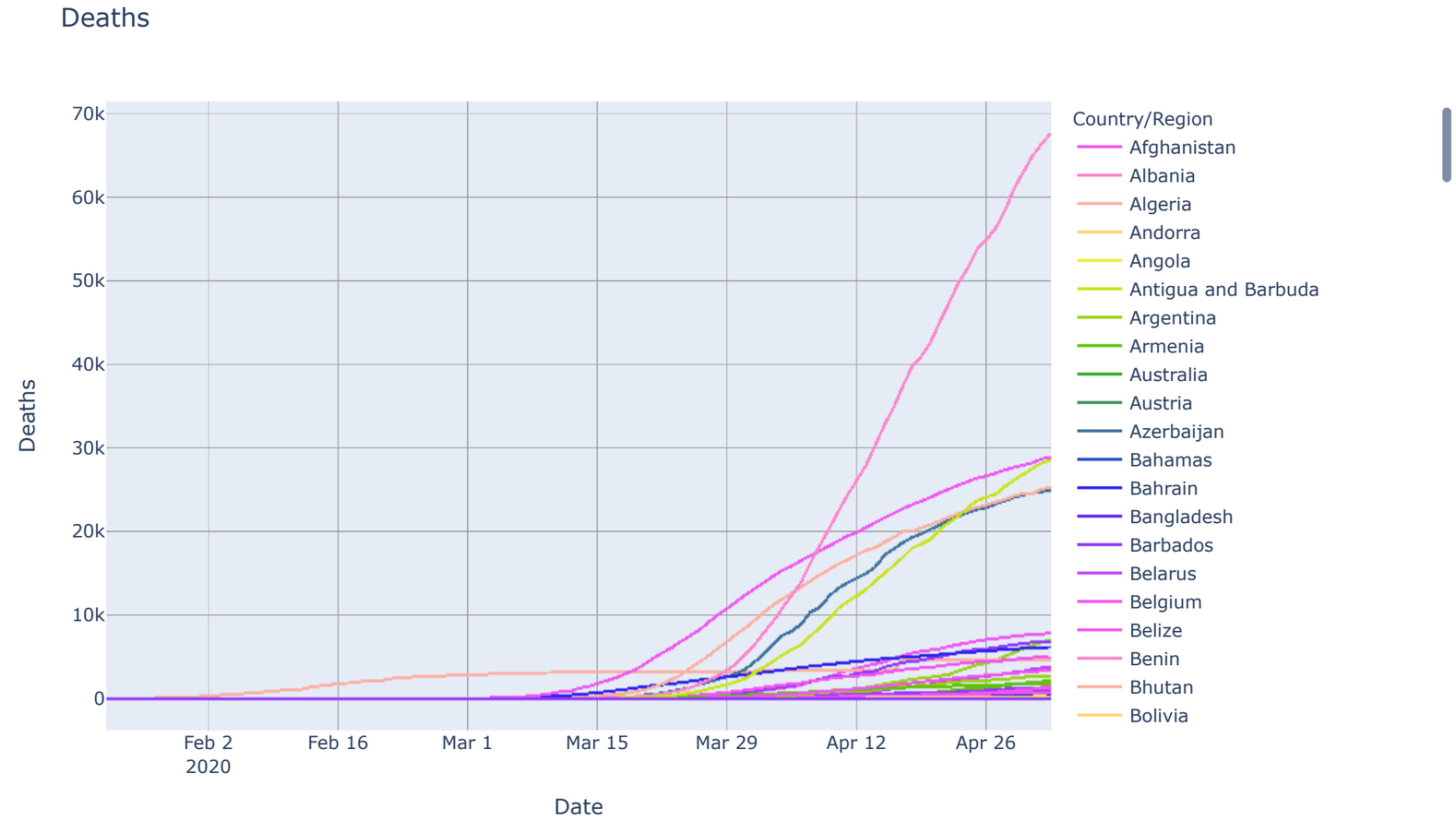


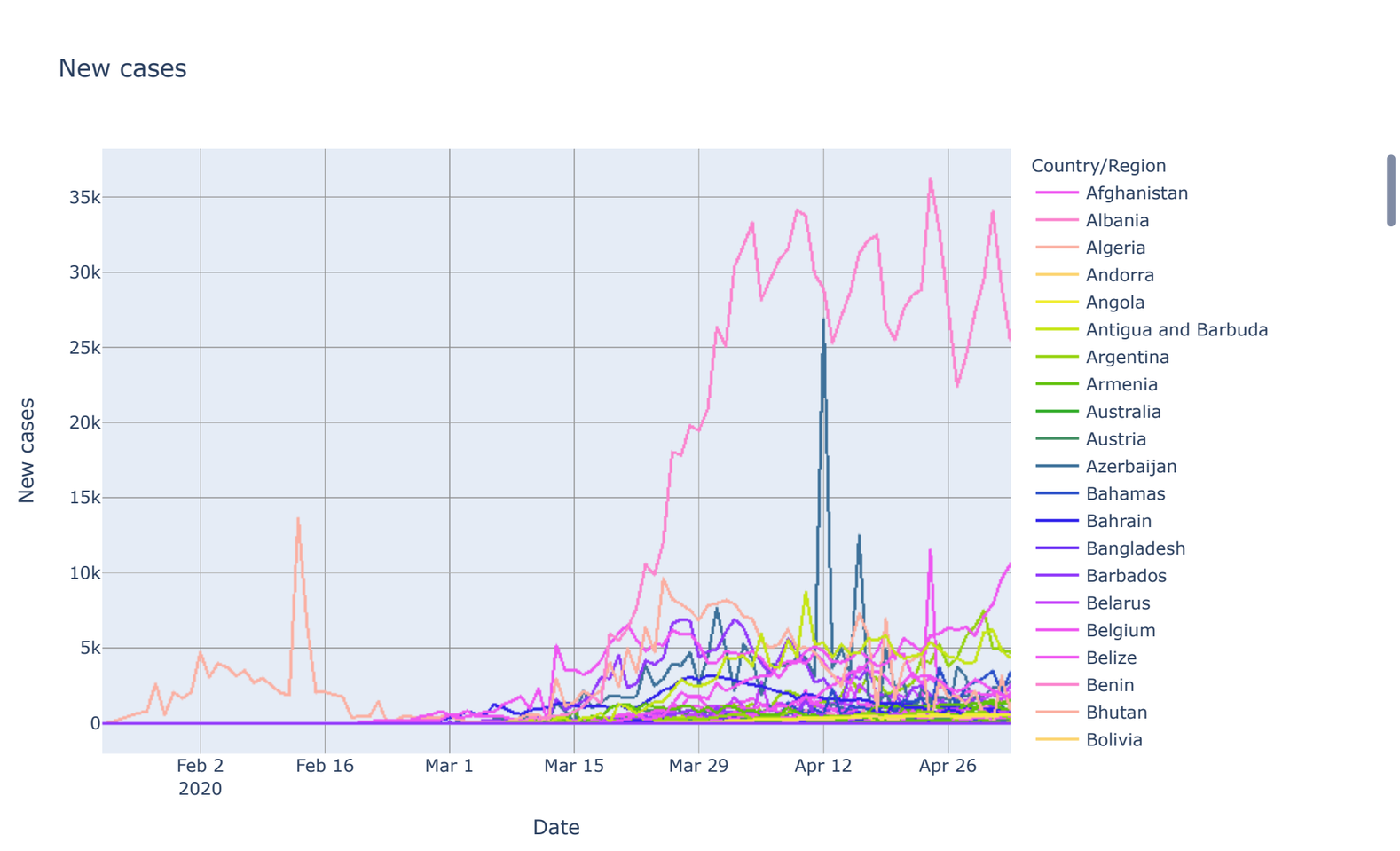




In [25]:
```
fig = px.line(full_grouped, x="Date", y="Confirmed", color='Country/Region', height=600,
              title='Confirmed', color_discrete_sequence = px.colors.cyclical.mygbm)
fig.show()

# ========================================

fig = px.line(full_grouped, x="Date", y="Deaths", color='Country/Region', height=600,
              title='Deaths', color_discrete_sequence = px.colors.cyclical.mygbm)
fig.show()

# ========================================

fig = px.line(full_grouped, x="Date", y="New cases", color='Country/Region', height=600,
              title='New cases', color_discrete_sequence = px.colors.cyclical.mygbm)
fig.show()
```



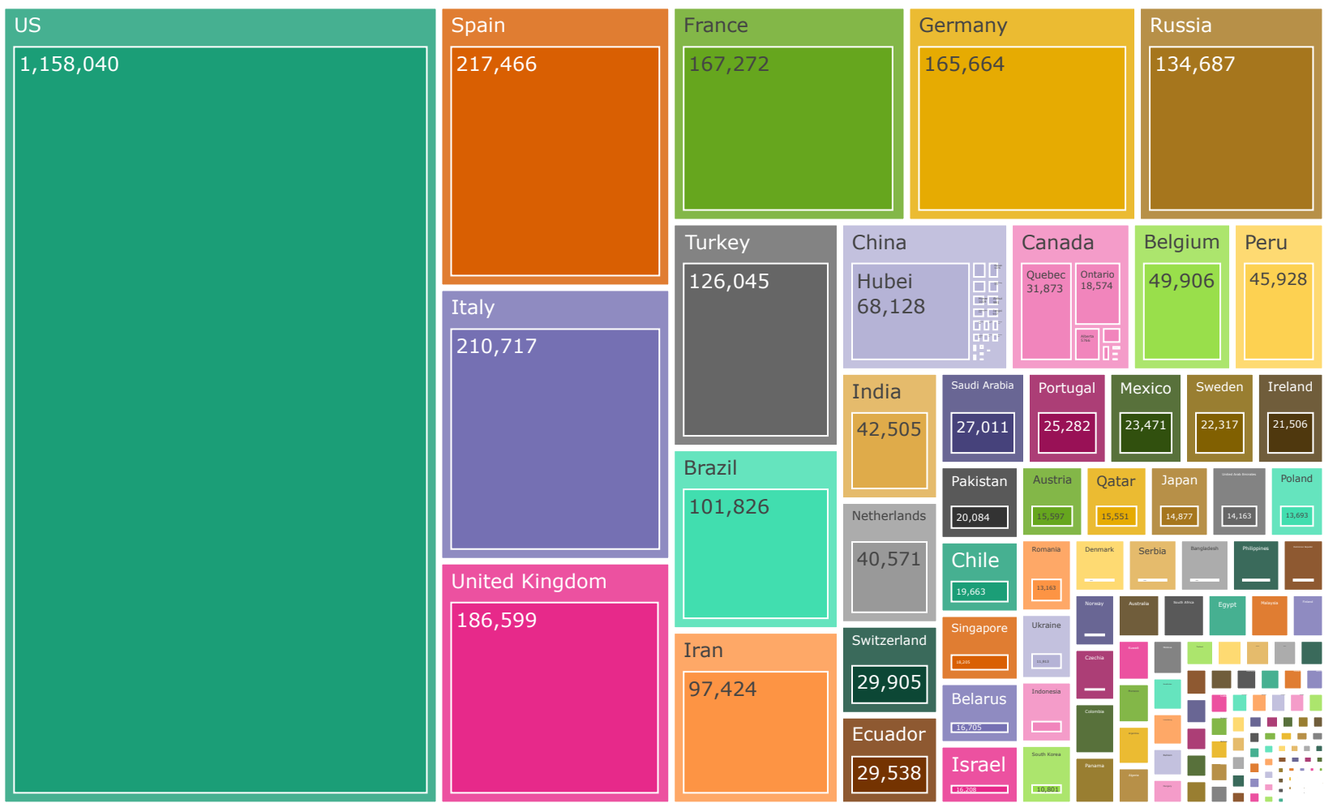


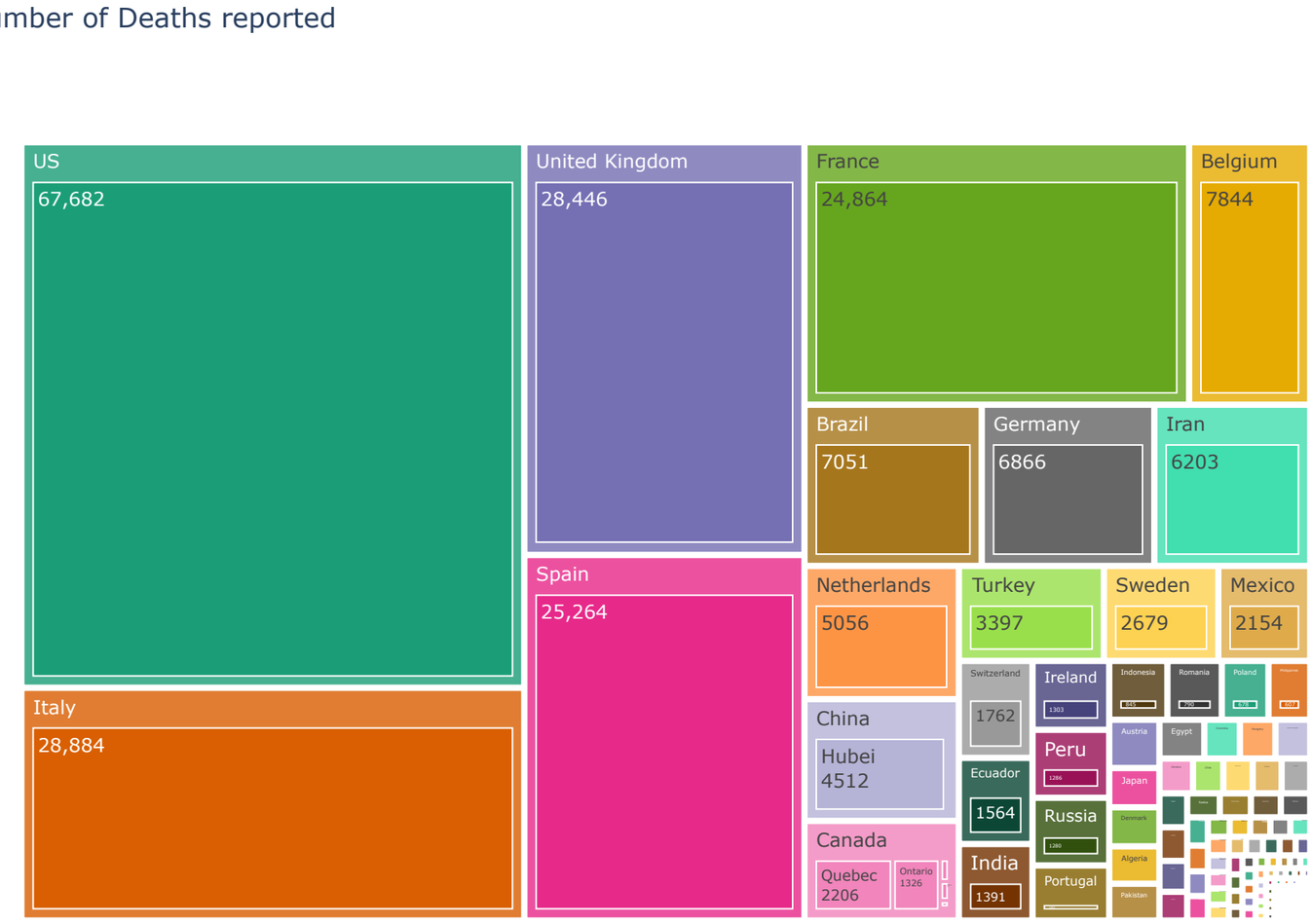


In [26]:
```
gt_100 = full_grouped[full_grouped['Confirmed']>100]['Country/Region'].unique()
temp = full_table[full_table['Country/Region'].isin(gt_100)]
temp = temp.groupby(['Country/Region', 'Date'])['Confirmed'].sum().reset_index()
temp = temp[temp['Confirmed']>100]
# print(temp.head())

min_date = temp.groupby('Country/Region')['Date'].min().reset_index()
min_date.columns = ['Country/Region', 'Min Date']
# print(min_date.head())

from_100th_case = pd.merge(temp, min_date, on='Country/Region')
from_100th_case['N days'] = (from_100th_case['Date'] - from_100th_case['Min Date']).dt.days
# print(from_100th_case.head())

fig = px.line(from_100th_case, x='N days', y='Confirmed', color='Country/Region', title='N days from 100 case', height=600)
fig.show()

# ============================================================================

gt_1000 = full_grouped[full_grouped['Confirmed']>1000]['Country/Region'].unique()
temp = full_table[full_table['Country/Region'].isin(gt_1000)]
temp = temp.groupby(['Country/Region', 'Date'])['Confirmed'].sum().reset_index()
temp = temp[temp['Confirmed']>1000]
# print(temp.head())

min_date = temp.groupby('Country/Region')['Date'].min().reset_index()
min_date.columns = ['Country/Region', 'Min Date']
# print(min_date.head())

from_1000th_case = pd.merge(temp, min_date, on='Country/Region')
from_1000th_case['N days'] = (from_1000th_case['Date'] - from_1000th_case['Min Date']).dt.days
# print(from_1000th_case.head())

fig = px.line(from_1000th_case, x='N days', y='Confirmed', color='Country/Region', title='N days from 1000 case', height=600)
fig.show()

# ============================================================================

gt_10000 = full_grouped[full_grouped['Confirmed']>10000]['Country/Region'].unique()
temp = full_table[full_table['Country/Region'].isin(gt_10000)]
temp = temp.groupby(['Country/Region', 'Date'])['Confirmed'].sum().reset_index()
temp = temp[temp['Confirmed']>10000]
# print(temp.head())

min_date = temp.groupby('Country/Region')['Date'].min().reset_index()
min_date.columns = ['Country/Region', 'Min Date']
# print(min_date.head())

from_10000th_case = pd.merge(temp, min_date, on='Country/Region')
from_10000th_case['N days'] = (from_10000th_case['Date'] - from_10000th_case['Min Date']).dt.days
# print(from_10000th_case.head())full_grouped

fig = px.line(from_10000th_case, x='N days', y='Confirmed', color='Country/Region', title='N days from 10000 case', height=600)
fig.show()
```

Composition of Cases

In [27]:
```
full_latest = full_table[full_table['Date'] == max(full_table['Date'])]

fig = px.treemap(full_latest.sort_values(by='Confirmed', ascending=False).reset_index(drop=True),
                 path=["Country/Region", "Province/State"], values="Confirmed", height=700,
                 title='Number of Confirmed Cases',
                 color_discrete_sequence = px.colors.qualitative.Dark2)
fig.data[0].textinfo = 'label+text+value'
fig.show()

fig = px.treemap(full_latest.sort_values(by='Deaths', ascending=False).reset_index(drop=True),
                 path=["Country/Region", "Province/State"], values="Deaths", height=700,
                 title='Number of Deaths reported',
                 color_discrete_sequence = px.colors.qualitative.Dark2)
fig.data[0].textinfo = 'label+text+value'
fig.show()
```





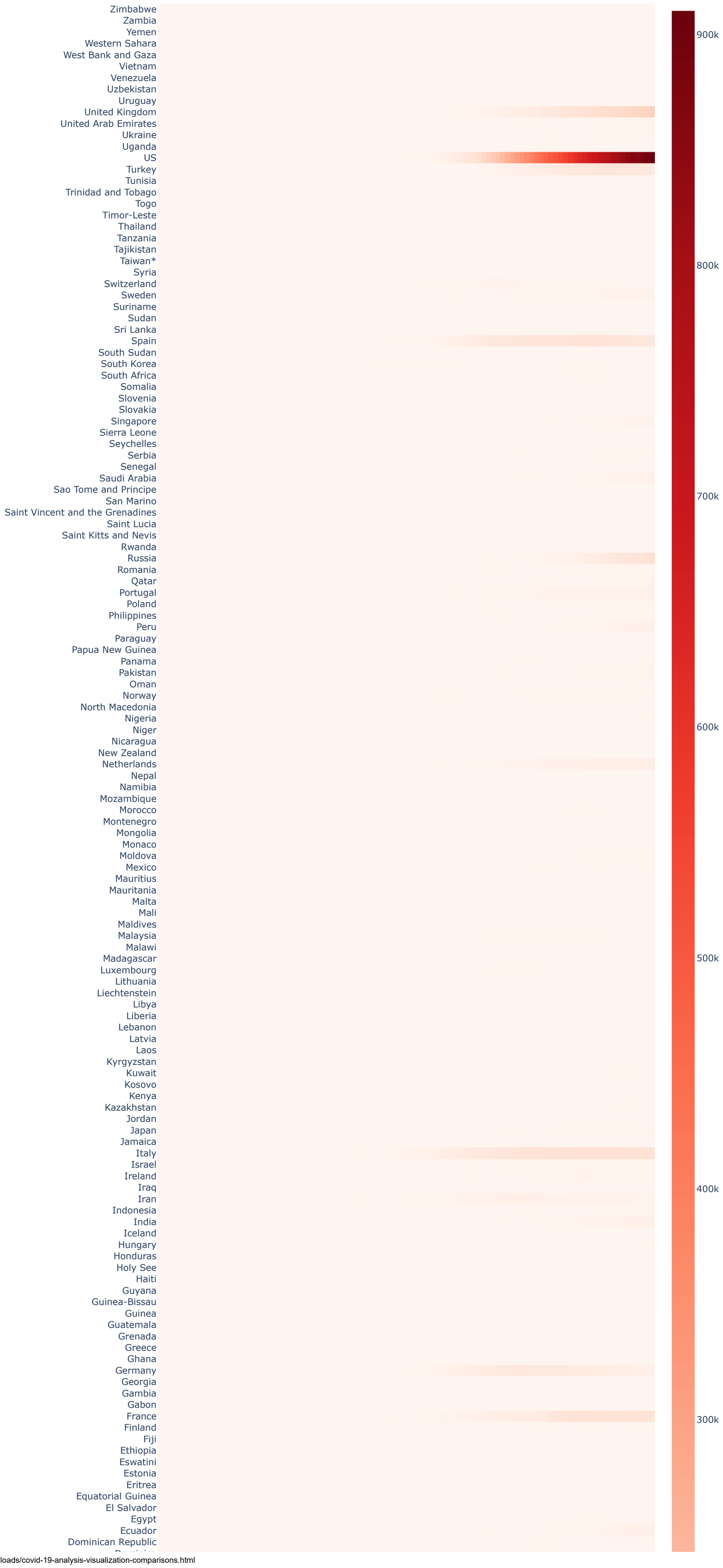
New cases

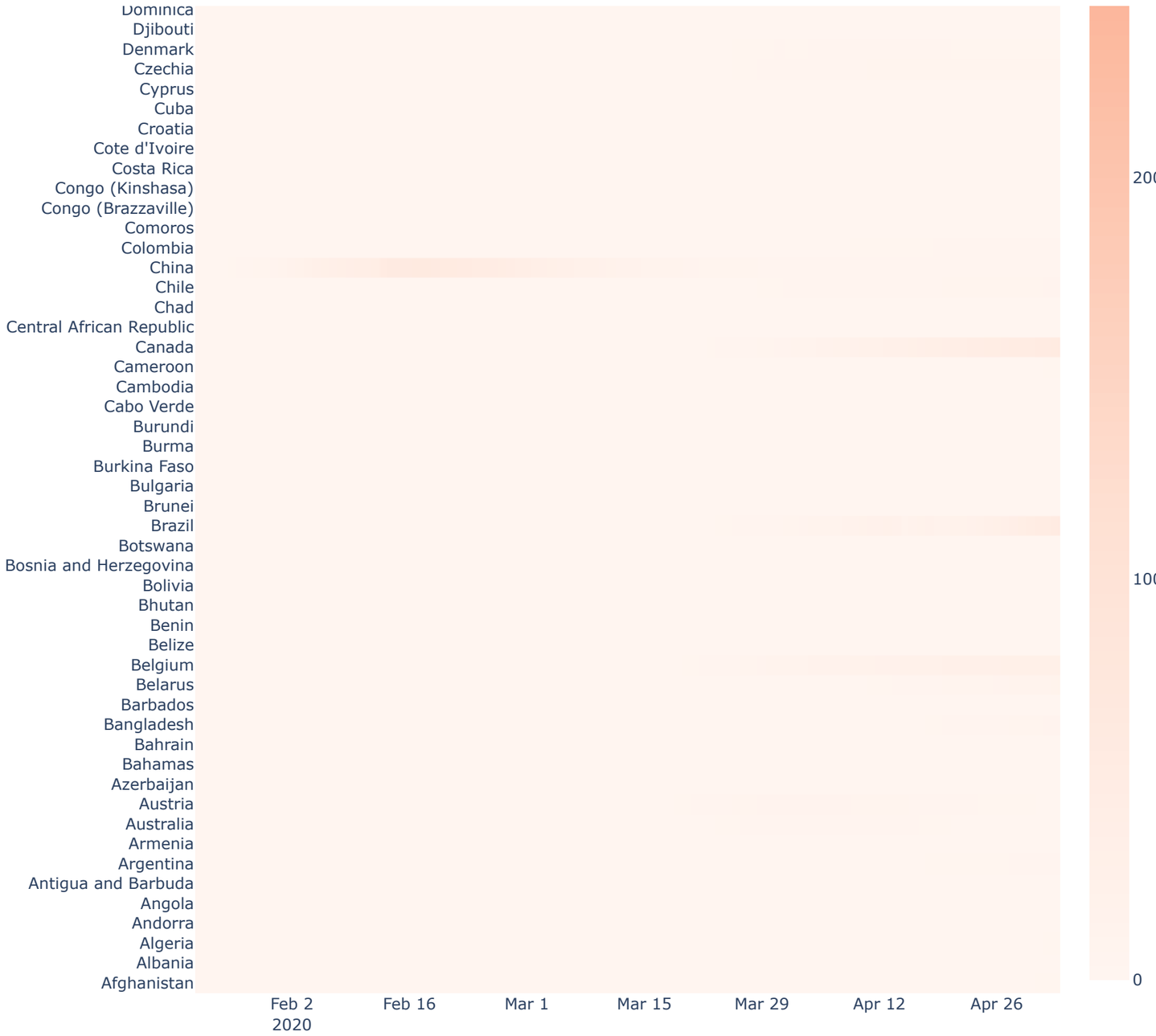
In [28]:
```
temp = full_grouped[full_grouped['New cases']>0].sort_values('Country/Region', ascending=False)
fig = px.scatter(temp, x='Date', y='Country/Region', size='New cases', color='New cases', height=3000,
           color_continuous_scale=px.colors.sequential.Viridis)
fig.update_layout(yaxis = dict(dtick = 1))
fig.update(layout_coloraxis_showscale=False)
fig.show()
```


Country/Region
Afghanistan
Albania
Algeria
Andorra
Angola
Antigua and Barbuda
Argentina
Armenia
Australia
Austria
Azerbaijan
Bahamas
Bahrain
Bangladesh
Barbados
Belarus
Belgium
Belize
Benin
Bhutan
Bolivia
Bosnia and Herzegovina
Botswana
Brazil
Brunei
Bulgaria
Burkina Faso
Burma
Burundi
Cabo Verde
Cambodia
Cameroon
Canada
Central African Republic
Chad
Chile
China
Colombia
Comoros
Congo (Brazzaville)
Congo (Kinshasa)
Costa Rica
Cote d'Ivoire
Croatia
Cuba
Cyprus
Czechia
Denmark
Djibouti
Dominica
Dominican Republic
Ecuador
Egypt
El Salvador
Equatorial Guinea
Eritrea
Estonia
Eswatini
Ethiopia
Fiji
Finland
France
Gabon
Gambia
Georgia
Germany
Ghana
Greece
Grenada
Guatemala
Guinea
Guinea-Bissau
Guyana
Haiti
Holy See
Honduras
Hungary
Iceland
India
Indonesia
Iran
Iraq
Ireland
Israel
Italy
Jamaica
Japan
Jordan
Kazakhstan
Kenya
Kosovo
Kuwait
Kyrgyzstan
Laos
Latvia
Lebanon
Liberia
Libya
Liechtenstein
Lithuania
Luxembourg
Madagascar
Malawi
Malaysia
Maldives
Mali
Malta
Mauritania
Mauritius
Mexico
Moldova
Monaco
Mongolia
Montenegro
Morocco
Mozambique
Namibia
Nepal
Netherlands
New Zealand
Nicaragua
Niger
Nigeria
North Macedonia
Norway
Oman
Pakistan
Panama
Papua New Guinea
Paraguay
Peru
Philippines
Poland
Portugal
Qatar
Romania
Russia
Rwanda
Saint Kitts and Nevis
Saint Lucia
Saint Vincent and the Grenadines

Active cases

In [29]:
```
fig = go.Figure(data=go.Heatmap(
        z=full_grouped['Active'],
        x=full_grouped['Date'],
        y=full_grouped['Country/Region'],
        colorscale='Reds',
        showlegend=False,
        text=full_grouped['Active']))

fig.update_layout(yaxis = dict(dtick = 1))
fig.update_layout(height=3000)
fig.show()
```


Zimbabwe
Zambia
Yemen
Western Sahara
West Bank and Gaza
Vietnam
Venezuela
Uzbekistan
Uruguay
United Kingdom
United Arab Emirates
Ukraine
Uganda
US
Turkey
Tunisia
Trinidad and Tobago
Togo
Timor-Leste
Thailand
Tanzania
Tajikistan
Taiwan*
Syria
Switzerland
Sweden
Suriname
Sudan
Sri Lanka
Spain
South Sudan
South Korea
South Africa
Somalia
Slovenia
Slovakia
Singapore
Sierra Leone
Seychelles
Serbia
Senegal
Saudi Arabia
Sao Tome and Principe
San Marino
Saint Vincent and the Grenadines
Saint Lucia
Saint Kitts and Nevis
Rwanda
Russia
Romania
Qatar
Portugal
Poland
Philippines
Peru
Paraguay
Papua New Guinea
Panama
Pakistan
Oman
Norway
North Macedonia
Nigeria
Niger
Nicaragua
New Zealand
Netherlands
Nepal
Namibia
Mozambique
Morocco
Montenegro
Mongolia
Monaco
Moldova
Mexico
Mauritius
Mauritania
Malta
Mali
Maldives
Malaysia
Malawi
Madagascar
Luxembourg
Lithuania
Liechtenstein
Libya
Liberia
Lebanon
Latvia
Laos
Kyrgyzstan
Kuwait
Kosovo
Kenya
Kazakhstan
Jordan
Japan
Jamaica
Italy
Israel
Ireland
Iraq
Iran
Indonesia
India
Iceland
Hungary
Honduras
Holy See
Haiti
Guyana
Guinea-Bissau
Guinea
Guatemala
Grenada
Greece
Ghana
Germany
Georgia
Gambia
Gabon
France
Finland
Fiji
Ethiopia
Eswatini
Estonia
Eritrea
Equatorial Guinea
El Salvador
Egypt
Ecuador
Dominican Republic
900k
800k
700k
600k
500k
400k
300k



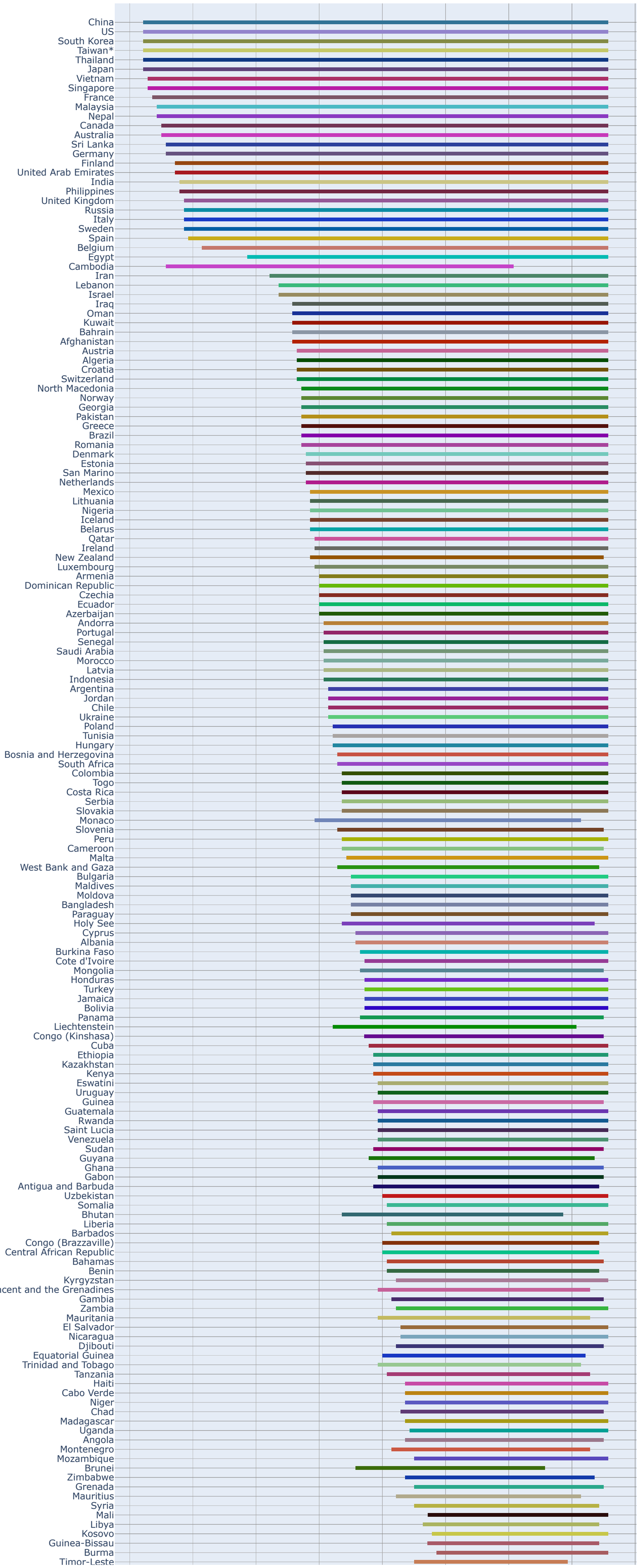
Epidemic Span

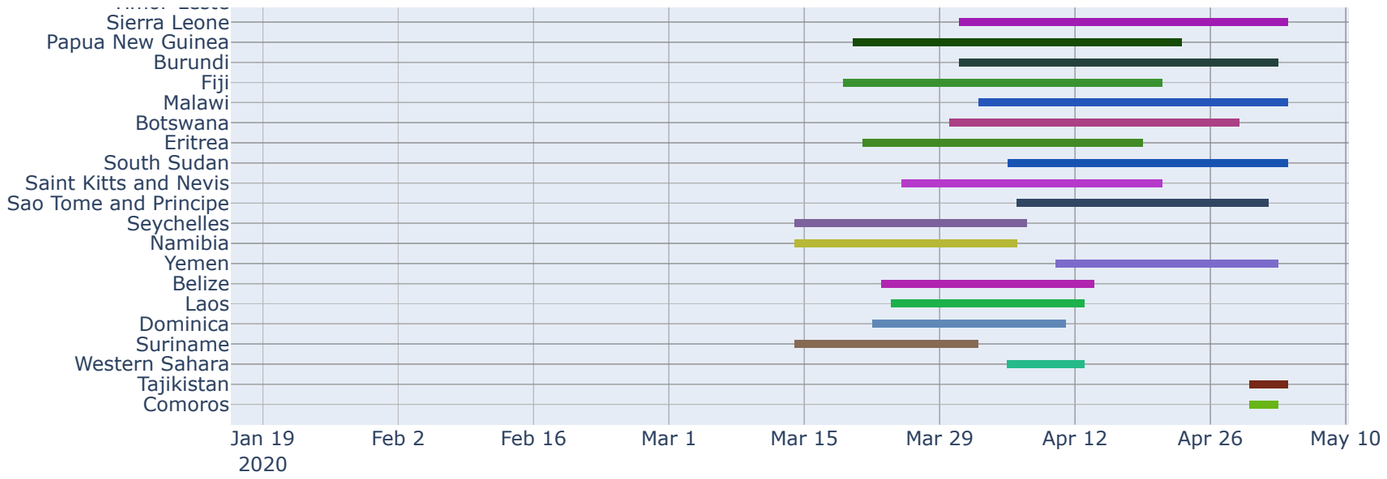
Note : In the graph, last day is shown as one day after the last time a new confirmed cases reported in the Country / Region

In [30]:
```
# first date
# ==========
first_date = full_table[full_table['Confirmed']>0]
first_date = first_date.groupby('Country/Region')['Date'].agg(['min']).reset_index()
# first_date.head()

# last date
# =========
last_date = full_table.groupby(['Country/Region', 'Date', ])['Confirmed', 'Deaths', 'Recovered']
last_date = last_date.sum().diff().reset_index()

mask = last_date['Country/Region'] != last_date['Country/Region'].shift(1)
last_date.loc[mask, 'Confirmed'] = np.nan
last_date.loc[mask, 'Deaths'] = np.nan
last_date.loc[mask, 'Recovered'] = np.nan

last_date = last_date[last_date['Confirmed']>0]
last_date = last_date.groupby('Country/Region')['Date'].agg(['max']).reset_index()
# last_date.head()

# first_last
# ==========
first_last = pd.concat([first_date, last_date[['max']]], axis=1)

# added 1 more day, which will show the next day as the day on which last case appeared
first_last['max'] = first_last['max'] + timedelta(days=1)

# no. of days
first_last['Days'] = first_last['max'] - first_last['min']

# task column as country
first_last['Task'] = first_last['Country/Region']

# rename columns
first_last.columns = ['Country/Region', 'Start', 'Finish', 'Days', 'Task']

# sort by no. of days
first_last = first_last.sort_values('Days')
# first_last.head()

# visualization
# =============

# produce random colors
clr = ["#"+''.join([random.choice('0123456789ABC') for j in range(6)]) for i in range(len(first_last))]

# plot
fig = ff.create_gantt(first_last, index_col='Country/Region', colors=clr, show_colorbar=False,
                      bar_width=0.2, showgrid_x=True, showgrid_y=True, height=2500)
fig.show()
```

Gantt Chart

1w 1m 6m YTD 1y all

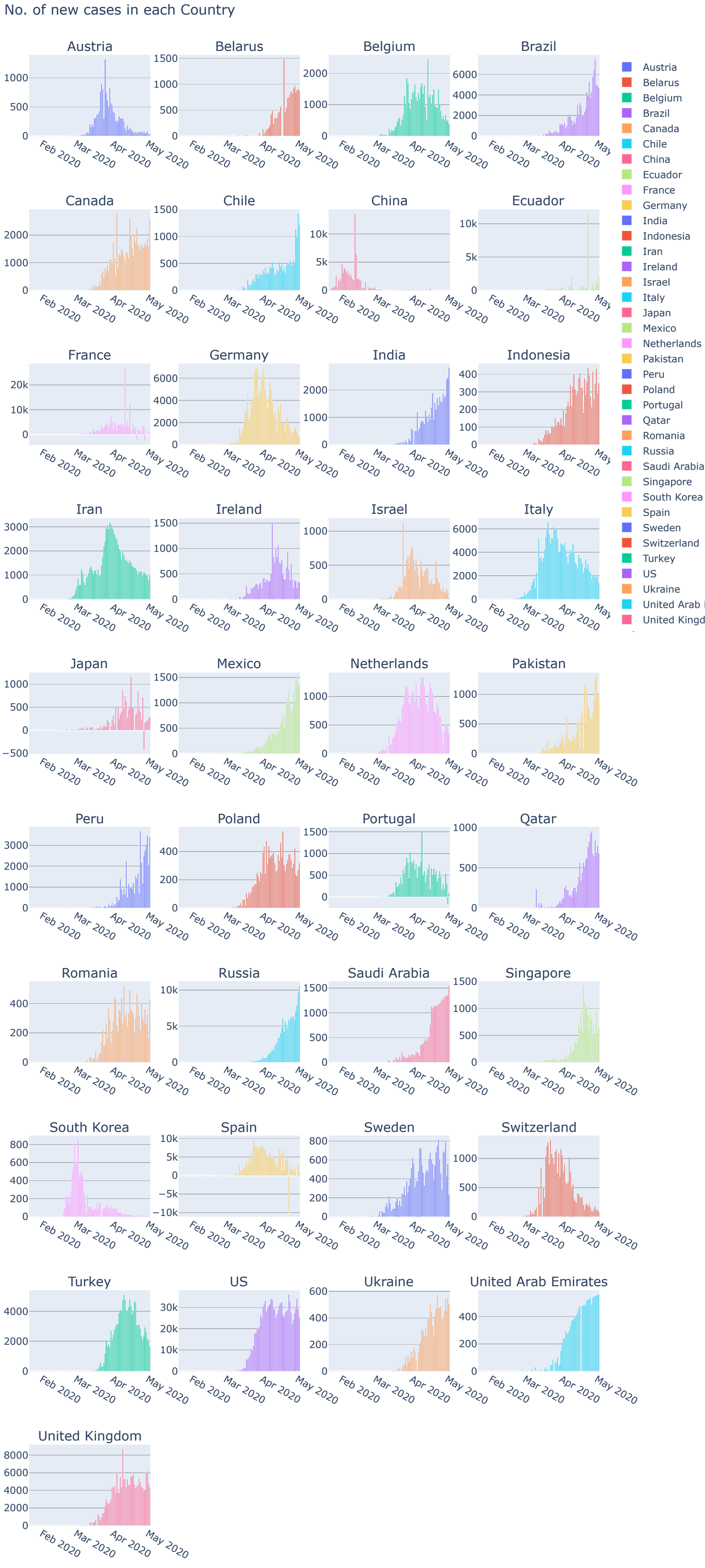




https://app.flourish.studio/visualisation/1571387/edit (https://app.flourish.studio/visualisation/1571387/edit)

Country Wise

In [32]:
```
temp = full_table.groupby(['Country/Region', 'Date', ])['Confirmed', 'Deaths']
temp = temp.sum().diff().reset_index()

mask = temp['Country/Region'] != temp['Country/Region'].shift(1)

temp.loc[mask, 'Confirmed'] = np.nan
temp.loc[mask, 'Deaths'] = np.nan

temp = temp[temp['Country/Region'].isin(gt_10000)]

# countries = ['China', 'Iran', 'South Korea', 'Italy', 'France', 'Germany', 'Italy', 'Spain', 'US']
countries = temp['Country/Region'].unique()

n_cols = 4
n_rows = math.ceil(len(countries)/n_cols)

fig = make_subplots(rows=n_rows, cols=n_cols, shared_xaxes=False, subplot_titles=countries)

for ind, country in enumerate(countries):
    row = int((ind/n_cols)+1)
    col = int((ind%n_cols)+1)
    fig.add_trace(go.Bar(x=temp['Date'], y=temp.loc[temp['Country/Region']==country, 'Confirmed'], name=country), row=row, col=col)

fig.update_layout(height=2000, title_text="No. of new cases in each Country")
fig.show()
```



Calander map

Number of new cases every day

In [33]:
```
temp = full_table.groupby('Date')['Confirmed'].sum()
temp = temp.diff()

plt.figure(figsize=(20, 5))
ax = calmap.yearplot(temp, fillcolor='white', cmap='Reds', linewidth=0.5)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-33-374279ef5d5e> in <module>
      3
      4 plt.figure(figsize=(20, 5))
----> 5 ax = calmap.yearplot(temp, fillcolor='white', cmap='Reds', linewidth=0.5)

H:\Anaconda_Python_3.7\lib\site-packages\calmap\__init__.py in yearplot(data, year, how, vmin, vmax, cmap, fillcolor, linewidth, linecolor, daylabels, dayticks, monthlabels, mont
hticks, ax, **kwargs)
    219     ax.set_xlabel('')
    220     ax.set_xticks([by_day.ix[datetime.date(year, i + 1, 15)].week
--> 221                    for i in monthticks])
    222     ax.set_xticklabels([monthlabels[i] for i in monthticks], ha='center')
    223

H:\Anaconda_Python_3.7\lib\site-packages\calmap\__init__.py in <listcomp>(.0)
    219     ax.set_xlabel('')
    220     ax.set_xticks([by_day.ix[datetime.date(year, i + 1, 15)].week
--> 221                    for i in monthticks])
    222     ax.set_xticklabels([monthlabels[i] for i in monthticks], ha='center')
    223

H:\Anaconda_Python_3.7\lib\site-packages\pandas\core\generic.py in __getattr__(self, name)
   5272             if self._info_axis._can_hold_identifiers_and_holds_name(name):
   5273                 return self[name]
-> 5274             return object.__getattribute__(self, name)
   5275
   5276     def __setattr__(self, name: str, value) -> None:

AttributeError: 'DataFrame' object has no attribute 'ix'
```

Number of new countries every day

In [34]:
```
spread = full_table[full_table['Confirmed']!=0].groupby('Date')
spread = spread['Country/Region'].unique().apply(len).diff()

plt.figure(figsize=(20, 5))
ax = calmap.yearplot(spread, fillcolor='white', cmap='Greens', linewidth=0.5)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-34-49ee01b24edc> in <module>
      3
      4 plt.figure(figsize=(20, 5))
----> 5 ax = calmap.yearplot(spread, fillcolor='white', cmap='Greens', linewidth=0.5)

H:\Anaconda_Python_3.7\lib\site-packages\calmap\__init__.py in yearplot(data, year, how, vmin, vmax, cmap, fillcolor, linewidth, linecolor, daylabels, dayticks, monthlabels, mont
hticks, ax, **kwargs)
    219     ax.set_xlabel('')
    220     ax.set_xticks([by_day.ix[datetime.date(year, i + 1, 15)].week
--> 221                    for i in monthticks])
    222     ax.set_xticklabels([monthlabels[i] for i in monthticks], ha='center')
    223

H:\Anaconda_Python_3.7\lib\site-packages\calmap\__init__.py in <listcomp>(.0)
    219     ax.set_xlabel('')
    220     ax.set_xticks([by_day.ix[datetime.date(year, i + 1, 15)].week
--> 221                    for i in monthticks])
    222     ax.set_xticklabels([monthlabels[i] for i in monthticks], ha='center')
    223

H:\Anaconda_Python_3.7\lib\site-packages\pandas\core\generic.py in __getattr__(self, name)
   5272             if self._info_axis._can_hold_identifiers_and_holds_name(name):
   5273                 return self[name]
-> 5274             return object.__getattribute__(self, name)
   5275
   5276     def __setattr__(self, name: str, value) -> None:

AttributeError: 'DataFrame' object has no attribute 'ix'
```

Comparison with similar epidemics

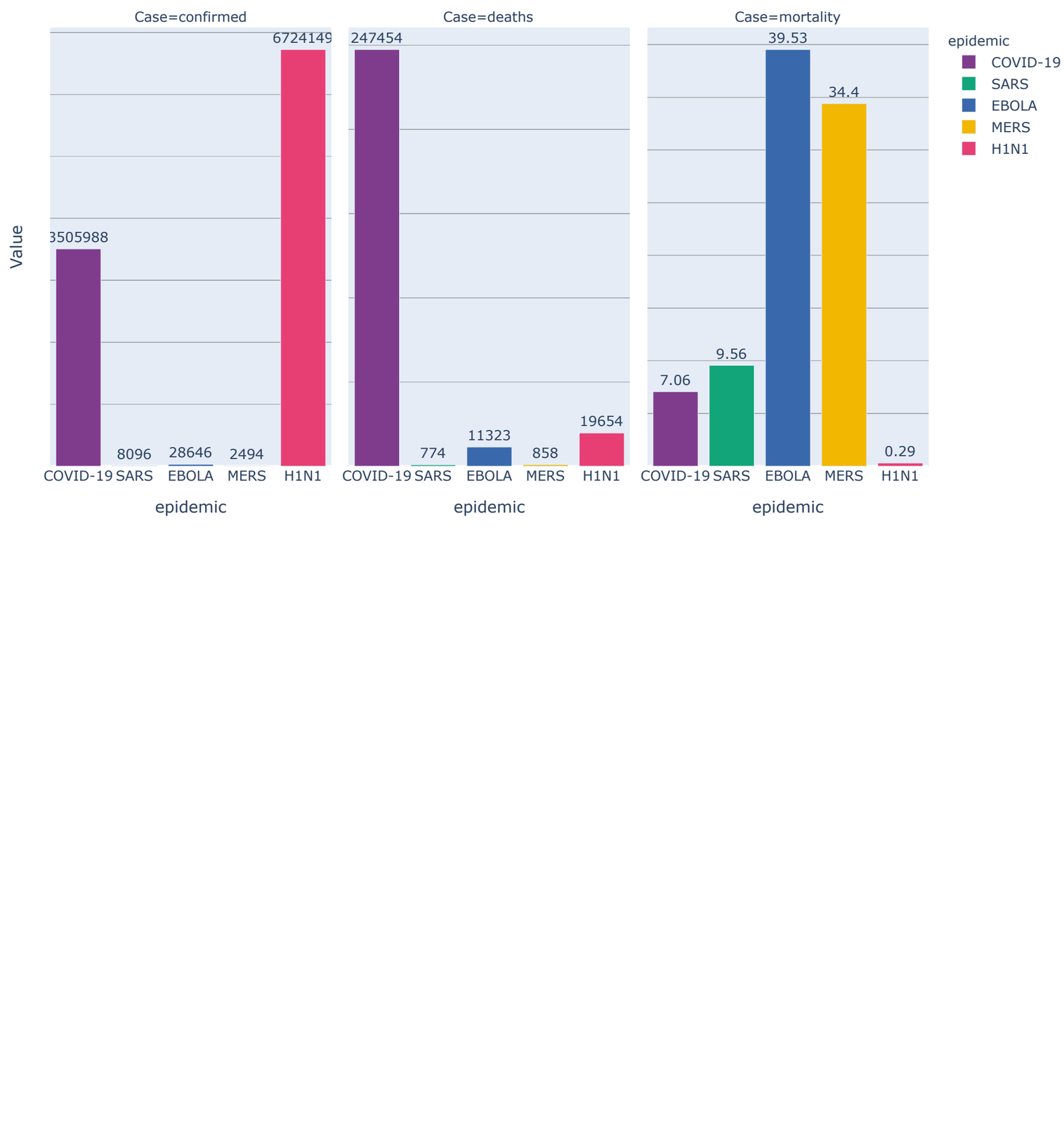
https://www.kaggle.com/imdevskp/covid19-vs-sars-vs-mers-vs-ebola-vs-h1n1 (https://www.kaggle.com/imdevskp/covid19-vs-sars-vs-mers-vs-ebola-vs-h1n1)

In [35]:
```
epidemics = pd.DataFrame({
    'epidemic' : ['COVID-19', 'SARS', 'EBOLA', 'MERS', 'H1N1'],
    'start_year' : [2019, 2003, 2014, 2012, 2009],
    'end_year' : [2020, 2004, 2016, 2017, 2010],
    'confirmed' : [full_latest['Confirmed'].sum(), 8096, 28646, 2494, 6724149],
    'deaths' : [full_latest['Deaths'].sum(), 774, 11323, 858, 19654]
})

epidemics['mortality'] = round((epidemics['deaths']/epidemics['confirmed'])*100, 2)

epidemics.head()
```

Out[35]:

	epidemic	start_year	end_year	confirmed	deaths	mortality
0	COVID-19	2019	2020	3505988	247454	7.06
1	SARS	2003	2004	8096	774	9.56
2	EBOLA	2014	2016	28646	11323	39.53
3	MERS	2012	2017	2494	858	34.40
4	H1N1	2009	2010	6724149	19654	0.29

In [36]:
```
temp = epidemics.melt(id_vars='epidemic', value_vars=['confirmed', 'deaths', 'mortality'],
                      var_name='Case', value_name='Value')

fig = px.bar(temp, x="epidemic", y="Value", color='epidemic', text='Value', facet_col="Case",
             color_discrete_sequence = px.colors.qualitative.Bold)
fig.update_traces(textposition='outside')
fig.update_layout(uniformtext_minsize=8, uniformtext_mode='hide')
fig.update_yaxes(showticklabels=False)
fig.layout.yaxis2.update(matches=None)
fig.layout.yaxis3.update(matches=None)
fig.show()
```



Limitations

There are a lot of limitations:

- The pandemic situation is still prevailing and data is changing and growing fast.
- Continuous data growth causes limitations of analysis.
- Many countries are recovering while other countries are still in the red zone which can’t be visualize continuously.
- The contamination based on gender could have been shown, but lack of data caused limitations.
- Generally the old people are highly affected due to this pandemic but there are exceptions too which have been ignored.

Conclusions

The pandemic situation intensifies the value of this research which can give certain insights about the COVID-19. The nature of its spread, multiplication rate, death rate, recoveries, economic conditions, predictions of next decades can easily be expressed by visualizing these simple graphs. The helpless military and nuclear power proves that simple virus can be the reason of destruction of the whole mankind indicating another prediction called ‘Biological War’. The whole situation is dependent upon the researchers, biologists, chemists, data scientists, doctors who are putting restless efforts to formulate a vaccine and save the human being. The research is not totally correct, limitations lie within data and more sophisticated codes could be performed which hasn’t been run. So this overall procedures have been performed to leave a mark and contribute to the pandemic history.

Acknowledgements

I would like to acknowledge Johns Hopkins University for open-sourcing their dataset. Their dataset is transformed into a format that is easier for Jupyter Notebook to handle.
I believe that epidemic data should be openly available and easily accessible for health professionals and data scientists. This dataset would serve as a starting point for people to gather more data about epidemics, not just statistics, but also new stories, government responses etc.

References

1. https://cgdv.github.io/challenges/COVID-19/datasource/
2. https://github.com/CSSEGISandData/COVID-19
3. https://github.com/imdevskp/covid_19_jhu_data_web_scrap_and_cleaning

The datasets have been collected from the above sites and rest of the data processing, cleaning, transformation, analysis and visualization have been done by me.