COVID-19 Analysis, Visualization and Comparison

This project was made as a part of the Data Insight Program of 2020

Author : Omar Ossama Mahmoud Ahmed

ID #: 87

Introduction to COVID-19

Coronavirus is a family of viruses that can cause illness, which can vary from *common cold* and *cough* to sometimes more severe disease. **Middle East Respiratory Syndrome (MERS-CoV)** and **Severe Acute Respiratory Syndrome (SARS-CoV)** were such severe cases with the world already has faced.
SARS-CoV-2 (n-coronavirus) is the new virus of the coronavirus family, which first *discovered* in 2019, which has not been identified in humans before. It is a *contiguous* virus which started from **Wuhan** in **December 2019**. Which later declared as **Pandemic** by **WHO** due to high rate spreads throughout the world. Currently (on date 23rd of April 2020), this leads to a total of 189K+ Deaths across the globe, including *110K+ deaths* alone in *Europe*.
Pandemic is spreading all over the world; it becomes more important to understand about this spread. This NoteBook is an effort to analyze the data of confirmed, deaths, and recovered cases over time. In this notebook, the main focus is to analyze the spread trend of this virus all over the world.

SOURSES:

- [WHO (https://www.who.int/emergencies/diseases/novel-coronavirus-2019)](https://www.who.int/emergencies/diseases/novel-coronavirus-2019)
- [CDC (https://www.cdc.gov/coronavirus/2019-nCoV/index.html)](https://www.cdc.gov/coronavirus/2019-nCoV/index.html)
- [Worldometers COVID-19 Tracker (https://www.worldometers.info/world-population/population-by-country/)](https://www.worldometers.info/world-population/population-by-country/)
- [COVID-19 Tracker by Johns Hopkins University (https://www.arcgis.com/apps/opsdashboard/index.html#/bda7594740fd40299423467b48e9ecf6)](https://www.arcgis.com/apps/opsdashboard/index.html#/bda7594740fd40299423467b48e9ecf6)

Dataset

- 2019 Novel Coronavirus COVID-19 (2019-nCoV) [Data Repository (https://github.com/CSSEGISandData/COVID-19)](https://github.com/CSSEGISandData/COVID-19) by Johns Hopkins CSSE
- This dataset is updated on daily basis by Johns Hopkins CSSE

- 2020 Educational and Population Global [Data Repository (http://data.uis.unesco.org/)](http://data.uis.unesco.org/) by UNESCO UIS Statistics
- This dataset is updated on annual basis by UNESCO UIS Statistics

- 2020 World Population by country and population density by [Worldometer.info (https://www.worldometers.info/world-population/population-by-country/)](https://www.worldometers.info/world-population/population-by-country/)
- This dataset is updated on monthly basis by Worldometers.info

---

Table of Content :

- Countries with the highest confirmed to population ratio
- Exploring Population density wrt confirmed cases
    - Exploring densities of countries with confirmed cases within IQR of 50%
- Exploring Population illiteracy rate wrt confirmed cases
    - Exploring the countries with the highest illiteracy rate
- Final Thoughts

Installing Libraries

In [ ]:
```
pip install pycountry
```

In [ ]:
```
pip install empiricaldist
```

In [ ]:
```
pip install plotly_express
```

Imports and Datasets

- Pandas : for dataset handeling
- Numpy : Support for Pandas and calculations
- Datetime: for date and times calculations
- Math : for mathimatical operations
- Matplotlib : for visualization (basic)
- Empiricaldist : for statistical analysis
- Seaborn : for visualization and plotting (Presentable)
- pycountry : Library for getting continent (name) to from their country names
- plotly : for interative plots

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import datetime as dt
import math
import pycountry
import pycountry_convert as pc
from plotly.subplots import make_subplots
import plotly_express as px
import plotly.graph_objects as go
import plotly.figure_factory as ff
from plotly.subplots import make_subplots
import empiricaldist as emp
%matplotlib inline
sns.set_style('darkgrid')
import warnings
warnings.filterwarnings('ignore')
```

```
C:\ProgramData\Anaconda3\lib\site-packages\dask\config.py:168: YAMLLoadWarning:

calling yaml.load() without Loader=... is deprecated, as the default Loader is unsafe. Please read https://msg.pyyaml.org/load
for full details.
```

Dataset Used

2019 Novel Coronavirus COVID-19 (2019-nCoV) Data Repository by Johns Hopkins CSSE (LINK (https://github.com/CSSEGISandData/COVID-19))

Dataset consists of time-series data from 22 JAN 2020 up to this date (Updated on daily Basis).

Three Time-series dataset :

- time_series_covid19_confirmed_global.csv (Link Raw File (https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_time_series/time_series_covid19_confirmed_global.csv))
- time_series_19-covid-Deaths.csv (Link Raw File (https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_time_series/time_series_19-covid-Deaths.csv))
- time_series_covid19_deaths_global (Link Raw File (https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_time_series/time_series_covid19_deaths_global.csv))

Defining Functions

- ***ecdf() :*** for CDF calculation
- ***country_pick() :*** for filtering country by name from dataframe
- ***pxplotline() :*** for plotting lineplots using plotly_express
- ***mplotbar_single_country() :*** for plotting bar plots for one country using seaborn
- ***mplotline_single_country() :*** for plotting line plots for one country using seaborn
- ***mplotline_list_country() :*** for plotting line plots for multiple countries using seaborn
- ***add_daily() :*** adds daily columns with daily counts calculated
- ***gplotbar() :*** for plotting interactive bar plots using plotly
- ***get_country_details() :*** fetches country ISO and continent using pycountry
- ***count_cat() :*** divides count columns into categories
- ***convert() :*** converts column to float

In [2]:
```python
def ecdf(data):
    #credits DataCamp Justin Bois
    """Compute ECDF for a one-dimensional array of measurements."""
    # Number of data points: n
    n = len(data)

    # x-data for the ECDF: x
    x = np.sort(data)

    # y-data for the ECDF: y
    y = np.arange(1, n+1) / n

    return x, y

def country_pick(main_df,country_name,startdate):

    """Filters Selected Dataframe using country name"""

    df = main_df[main_df['country'] == country_name].reset_index().drop('index',axis=1)
    df = df[df.Date >= startdate]
    return df

def pxplotline(main_df,sub_df,y,x ='Date',title='No Title',hd=['pop']):

    """Plots line plot using plotly_express from selected Dataframe"""

    df = main_df.groupby(['country','Date','confirmed%','mortality%'],as_index=False)['confirmed','death','recovered','active']
.sum()
    df = df.merge(sub_df,on='country')
    df.drop(['Date_y','confirmed_y','death_y','recovered_y','active_y','confirmed%_y','mortality%_y'],axis=1,inplace=True)
    df.rename(columns={'Date_x':'Date','confirmed_x':'confirmed','death_x':'death','recovered_x':'recovered','active_x':'activ
e','confirmed%_x':'confirmed%','mortality%_x':'mortality%'},inplace=True)
    fig = px.line(df,x = x, y = y, color='country',title = title,hover_data=hd)
    fig.show()

def mplotbar_single_country(main_df,country_name,startdate,y,title='No Title'):

    """Plots bar plot using seaborn for a single country from selected Dataframe"""

    data = country_pick(main_df,country_name,startdate)
    fig, ax = plt.subplots(figsize = (20,10))
    fig2 = sns.barplot(data = data, x = 'Date', y = y, ax = ax,color = '#6495ED')
    ax.set_xticklabels(labels=data.Date.dt.strftime('%Y-%m-%d'), rotation=45, ha='right')
    plt.title(title)

def mplotline_single_country(main_df,country_name,startdate,y,title='No Title'):

    """Plots line plot using seaborn for a single country from selected Dataframe"""

    data = country_pick(main_df,country_name,startdate)
    fig, ax = plt.subplots(figsize = (20,10))
    fig = sns.lineplot(data = data , x = 'Date',y = y,marker = 'o',ax = ax)
    ax.set(xticks=data.Date.values)
    _=ax.set_xticklabels(labels=data.Date.dt.strftime('%m-%d'), rotation=45)
    plt.title(title)

def mplotline_list_country(main_df,country_names,startdate,y,fig=(20,10)):

    """Plots line plot using seaborn for a list of countries from selected Dataframe"""

    fig, ax = plt.subplots(figsize = fig)
    for i in country_names:
        df = main_df[main_df['country'] == i]
        df = df[df.Date >= startdate]
        fig = sns.lineplot(data = df , x = 'Date',y = y,marker = '.',ax = ax,label = i)
    ax.set(xticks=df.Date.values)
    _=ax.set_xticklabels(labels=df.Date.dt.strftime('%m-%d'), rotation=45)
    plt.legend()

def add_daily(df):

    """Adds columns of daily counts increase to selected Dataframe"""

    df.loc[0,'daily_confirmed'] = df.loc[0,'confirmed']
    df.loc[0,'daily_death'] = df.loc[0,'death']
    df.loc[0,'daily_recovered'] = df.loc[0,'recovered']
    df.loc[0,'daily_active'] = df.loc[0,'active']
    for i in range(1,len(df)):
        df.loc[i,'daily_confirmed'] = df.loc[i,'confirmed'] - df.loc[i-1,'confirmed']
        df.loc[i,'daily_death'] = df.loc[i,'death'] - df.loc[i-1,'death']
        df.loc[i,'daily_recovered'] = df.loc[i,'recovered'] - df.loc[i-1,'recovered']
        df.loc[i,'daily_active'] = df.loc[i,'active'] - df.loc[i-1,'active']
    df.loc[0,'daily_confirmed'] = 0
    df.loc[0,'daily_death'] = 0
    df.loc[0,'daily_recovered'] = 0
    df.loc[0,'daily_active'] = 0
    return df

def gplotbar(main_df,countryname,cols,startdate='1/1/2020',daily=False,title='No Title'):

    """Plots bar plot using plotly_express for a multiple countries from selected Dataframe"""
```

```
    if daily == True:
        if countryname == 'all':
            df = add_daily(main_df.groupby('Date',as_index=False).sum())
            df = df[df.Date >= startdate]
            data=[]
            for i in cols:
                data.append(go.Bar(name = f'daily_{i}',x = df['Date'], y = df[f'daily_{i}']))
            fig = go.Figure(data=data)
            fig.update_layout(barmode='overlay', title=title)
            fig.show()
        else:
            df = add_daily(main_df[main_df['country'] == countryname].groupby('Date',as_index=False).sum())
            df = df[df.Date >= startdate]
            data=[]
            for i in cols:
                data.append(go.Bar(name = f'daily_{i}',x = df['Date'], y = df[f'daily_{i}']))
            fig = go.Figure(data=data)
            fig.update_layout(barmode='overlay', title=title)
            fig.show()
    else:

        if countryname == 'all':
            df = main_df.groupby('Date',as_index=False).sum()
            df = df[df.Date >= startdate]
            data=[]
            for i in cols:
                data.append(go.Bar(name = i,x = df['Date'], y = df[i]))
            fig = go.Figure(data=data)
            fig.update_layout(barmode='overlay', title=title)
            fig.show()
        else:
            df = main_df[main_df['country'] == countryname].groupby('Date',as_index=False).sum()
            df = df[df.Date >= startdate]
            data=[]
            for i in cols:
                data.append(go.Bar(name = i,x = df['Date'], y = df[i]))
            fig = go.Figure(data=data)
            fig.update_layout(barmode='overlay', title=title)
            fig.show()

def get_country_details(country):

    """Returns country ISO and continent"""

    try:
        country_obj = pycountry.countries.get(name=country)
        if country_obj is None:
            c = pycountry.countries.search_fuzzy(country)
            country_obj = c[0]
        continent_code = pc.country_alpha2_to_continent_code(country_obj.alpha_2)
        continent = pc.convert_continent_code_to_continent_name(continent_code)
        return country_obj.alpha_3, continent
    except:
        if 'Congo' in country:
            country = 'Congo'
        elif country == 'Diamond Princess' or country == 'Laos' or country == 'MS Zaandam'\
        or country == 'Holy See' or country == 'Timor-Leste':
            return country, country
        elif country == 'Korea, South' or country == 'South Korea':
            country = 'Korea, Republic of'
        elif country == 'Taiwan*':
            country = 'Taiwan'
        elif country == 'Burma':
            country = 'Myanmar'
        elif country == 'West Bank and Gaza':
            country = 'Gaza'
        else:
            return country, country
        country_obj = pycountry.countries.search_fuzzy(country)
        continent_code = pc.country_alpha2_to_continent_code(country_obj[0].alpha_2)
        continent = pc.convert_continent_code_to_continent_name(continent_code)
        return country_obj[0].alpha_3, continent


def count_cat(n):

    """Returns catagorical group of a number"""

    if n < 25:
        return '< 25'
    elif (n >= 25) & (n <= 50):
        return '< 50'
    elif (n >= 50) & (n <= 100):
        return '< 100'
    elif (n >= 100) & (n <= 200):
        return '< 200'
    elif (n >= 200) & (n <= 1000):
        return '< 1000'
    elif (n >= 1000) & (n <= 5000):
        return '< 5000'
    elif (n >= 5000) & (n <= 10000):
        return '< 10,000'
    elif (n >= 10000) & (n <=30000):
```

```
        return '< 30,000'
    elif (n >= 30000) & (n <= 100000):
        return '< 100,000'
    elif (n >= 100000) & (n <= 150000):
        return '< 150,000'
    elif (n >= 150000) & (n <= 200000):
        return '< 200,000'
    elif (n >= 200000) & (n <= 250000):
        return '< 250,000'
    elif (n >= 250000) & (n <= 300000):
        return '< 300,000'
    elif (n >= 300000) & (n <= 400000):
        return '< 400,000'
    elif (n >= 400000) & (n <= 500000):
        return '< 500,000'
    else:
        return '> 500,000'


def convert(pop):

    """Converts Dataframe column to float"""

    if pop == float('nan'):
        return 0.0
    return float(pop.replace(',',''))
```

Importing Datasets from Local files

COVID-19 Datasets from Johns Hopkins CSSE

- ***time_series_covid19_confirmed_global.csv :*** COVID-19 Confirmed Counts Dataset from Johns Hopkins CSSE
- ***time_series_covid19_deaths_global.csv :*** COVID-19 Death Counts Dataset from Johns Hopkins CSSE
- ***time_series_covid19_recovered_global.csv:*** COVID-19 Recovered Counts Dataset from Johns Hopkins CSSE

Educational Datasets from UNIESCO UIS Statistics

- ***DataEd2.csv :*** Educational Dataset from UNIESCO UIS Statistics
- ***pop.csv :*** Educational population Dataset

World Population & Density Datasets from Worldometers.info

- ***wpop2.csv :*** World Population & Density Dataset from Worldometers.info

In [3]:

```
#Importing datasets of COVID-19 Confirmed, Death and Recovered counts
confirmed_cases = pd.read_csv('https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_1
9_time_series/time_series_covid19_confirmed_global.csv')
death_cases = pd.read_csv('https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_ti
me_series/time_series_covid19_deaths_global.csv')
recovered_cases = pd.read_csv('https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_1
9_time_series/time_series_covid19_recovered_global.csv')


#Importing datasets of education and educational population
ed = pd.read_csv('D:\DATASCIENCE\Project 1\COVID-19\Datasets\\education_illiteracy.csv')
pop = pd.read_csv('D:\DATASCIENCE\Project 1\COVID-19\Datasets\\educational_population.csv')


#Importing datasets of world population and density
worldpop = pd.read_csv('D:\DATASCIENCE\Project 1\COVID-19\Datasets\\world_population.csv')
```

Exploring Imported Datasets

Using .head(),.describe() and .info() methods of pandas

In [4]:
```
display(confirmed_cases.head())
display(confirmed_cases.describe())
display(confirmed_cases.info())

display(worldpop.head())
display(worldpop.describe())
display(worldpop.info())

display(ed.head())
display(ed.describe())
display(ed.info())
```

	Province/State	Country/Region	Lat	Long	1/22/20	1/23/20	1/24/20	1/25/20	1/26/20	1/27/20	...	4/16/20	4/17/20	4/18/20	4/19/20	4/20/20	4/2
0	NaN	Afghanistan	33.0000	65.0000	0	0	0	0	0	0	...	840	906	933	996	1026	
1	NaN	Albania	41.1533	20.1683	0	0	0	0	0	0	...	518	539	548	562	584	
2	NaN	Algeria	28.0339	1.6596	0	0	0	0	0	0	...	2268	2418	2534	2629	2718	[illegible]
3	NaN	Andorra	42.5063	1.5218	0	0	0	0	0	0	...	673	696	704	713	717	
4	NaN	Angola	-11.2027	17.8739	0	0	0	0	0	0	...	19	19	24	24	24	

5 rows × 99 columns

	Lat	Long	1/22/20	1/23/20	1/24/20	1/25/20	1/26/20	1/27/20	1/28/20	1/29/20	...	4/16/20	
count	264.000000	264.000000	264.000000	264.000000	264.000000	264.000000	264.000000	264.000000	264.000000	264.000000	...	264.000000	2
mean	21.317326	22.168315	2.102273	2.477273	3.564394	5.431818	8.022727	11.087121	21.128788	23.356061	...	8149.321970	84
std	24.734994	70.669996	27.382118	27.480921	34.210982	47.612615	66.537101	89.647535	220.011922	221.352587	...	46351.053634	483
min	-51.796300	-135.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	-1.000000	
25%	6.969250	-20.026050	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	48.750000	
50%	23.488100	20.535638	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	331.500000	3
75%	41.166075	78.750000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	1533.250000	15
max	71.706900	178.065000	444.000000	444.000000	549.000000	761.000000	1058.000000	1423.000000	3554.000000	3554.000000	...	667592.000000	6997

8 rows × 97 columns

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 264 entries, 0 to 263
Data columns (total 99 columns):
Province/State    82 non-null object
Country/Region    264 non-null object
Lat               264 non-null float64
Long              264 non-null float64
1/22/20           264 non-null int64
1/23/20           264 non-null int64
1/24/20           264 non-null int64
1/25/20           264 non-null int64
1/26/20           264 non-null int64
1/27/20           264 non-null int64
1/28/20           264 non-null int64
1/29/20           264 non-null int64
1/30/20           264 non-null int64
1/31/20           264 non-null int64
2/1/20            264 non-null int64
2/2/20            264 non-null int64
2/3/20            264 non-null int64
2/4/20            264 non-null int64
2/5/20            264 non-null int64
2/6/20            264 non-null int64
2/7/20            264 non-null int64
2/8/20            264 non-null int64
2/9/20            264 non-null int64
2/10/20           264 non-null int64
2/11/20           264 non-null int64
2/12/20           264 non-null int64
2/13/20           264 non-null int64
2/14/20           264 non-null int64
2/15/20           264 non-null int64
2/16/20           264 non-null int64
2/17/20           264 non-null int64
2/18/20           264 non-null int64
2/19/20           264 non-null int64
2/20/20           264 non-null int64
2/21/20           264 non-null int64
2/22/20           264 non-null int64
2/23/20           264 non-null int64
2/24/20           264 non-null int64
2/25/20           264 non-null int64
2/26/20           264 non-null int64
2/27/20           264 non-null int64
2/28/20           264 non-null int64
2/29/20           264 non-null int64
3/1/20            264 non-null int64
3/2/20            264 non-null int64
3/3/20            264 non-null int64
3/4/20            264 non-null int64
3/5/20            264 non-null int64
3/6/20            264 non-null int64
3/7/20            264 non-null int64
3/8/20            264 non-null int64
3/9/20            264 non-null int64
3/10/20           264 non-null int64
3/11/20           264 non-null int64
3/12/20           264 non-null int64
3/13/20           264 non-null int64
3/14/20           264 non-null int64
3/15/20           264 non-null int64
3/16/20           264 non-null int64
3/17/20           264 non-null int64
3/18/20           264 non-null int64
3/19/20           264 non-null int64
3/20/20           264 non-null int64
3/21/20           264 non-null int64
3/22/20           264 non-null int64
3/23/20           264 non-null int64
3/24/20           264 non-null int64
3/25/20           264 non-null int64
3/26/20           264 non-null int64
3/27/20           264 non-null int64
3/28/20           264 non-null int64
3/29/20           264 non-null int64
3/30/20           264 non-null int64
3/31/20           264 non-null int64
4/1/20            264 non-null int64
4/2/20            264 non-null int64
4/3/20            264 non-null int64
4/4/20            264 non-null int64
4/5/20            264 non-null int64
4/6/20            264 non-null int64
4/7/20            264 non-null int64
4/8/20            264 non-null int64
4/9/20            264 non-null int64
4/10/20           264 non-null int64
4/11/20           264 non-null int64
4/12/20           264 non-null int64
4/13/20           264 non-null int64
4/14/20           264 non-null int64
4/15/20           264 non-null int64
```

```
4/16/20          264 non-null int64
4/17/20          264 non-null int64
4/18/20          264 non-null int64
4/19/20          264 non-null int64
4/20/20          264 non-null int64
4/21/20          264 non-null int64
4/22/20          264 non-null int64
4/23/20          264 non-null int64
4/24/20          264 non-null int64
4/25/20          264 non-null int64
dtypes: float64(2), int64(95), object(2)
memory usage: 204.3+ KB

None
```

	Rank	Country (or dependent territory)	Area km2	Area mi2	Population	Density pop./km2	Density pop./mi2	Date	Population source
0	–	Macau	32.90	13	6,76,100	20,550	53,224	September 30, 2019	Official quarterly estimate
1	1	Monaco	2.02	0.78	38,300	18,960	49,106	December 31, 2018	Official estimate
2	2	Singapore	722.5	279	57,03,600	7,894	20,445	July 1, 2019	Official estimate
3	–	Hong Kong	1,106	427	75,00,700	6,782	17,565	December 31, 2019	Official estimate
4	–	Gibraltar	6.8	2.6	33,701	4,956	12,836	July 1, 2019	UN projection

	Rank	Country (or dependent territory)	Area km2	Area mi2	Population	Density pop./km2	Density pop./mi2	Date	Population source
count	251	251	251	251	251	251	251	251	244
unique	195	251	248	244	251	182	189	60	32
top	–	Democratic Republic of the Congo	21	171	6,70,60,000	80	41	July 1, 2019	Official estimate
freq	57	1	3	3	1	4	4	61	75

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 251 entries, 0 to 250
Data columns (total 9 columns):
Rank                               251 non-null object
Country (or dependent territory)   251 non-null object
Area km2                           251 non-null object
Area mi2                           251 non-null object
Population                         251 non-null object
Density pop./km2                   251 non-null object
Density pop./mi2                   251 non-null object
Date                               251 non-null object
Population source                  244 non-null object
dtypes: object(9)
memory usage: 17.7+ KB

None
```

	EDULIT_IND	Indicator	LOCATION	Country	TIME	Time	Value	Flag Codes	Flags
0	ILLPOP_AG15T99_M	Adult illiterate population, 15+ years, male (...	NPL	Nepal	2011	2011	2322673.0	NaN	NaN
1	ILLPOP_AG15T99_M	Adult illiterate population, 15+ years, male (...	NPL	Nepal	2018	2018	1814355.0	‡	UIS Estimation
2	ILLPOP_AG15T99	Adult illiterate population, 15+ years, both s...	ALB	Albania	2011	2011	72533.0	NaN	NaN
3	ILLPOP_AG15T99	Adult illiterate population, 15+ years, both s...	ALB	Albania	2012	2012	63752.0	NaN	NaN
4	ILLPOP_AG15T99	Adult illiterate population, 15+ years, both s...	ALB	Albania	2018	2018	44114.0	‡	UIS Estimation

	TIME	Time	Value
count	6730.000000	6730.000000	6.706000e+03
mean	2014.057801	2014.057801	1.273797e+06
std	2.769130	2.769130	7.869407e+06
min	2010.000000	2010.000000	9.925600e-01
25%	2011.000000	2011.000000	8.039080e+01
50%	2014.000000	2014.000000	3.144300e+04
75%	2017.000000	2017.000000	3.895828e+05
max	2018.000000	2018.000000	2.661803e+08

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6730 entries, 0 to 6729
Data columns (total 9 columns):
EDULIT_IND    6730 non-null object
Indicator     6730 non-null object
LOCATION      6730 non-null object
Country       6730 non-null object
TIME          6730 non-null int64
Time          6730 non-null int64
Value         6706 non-null float64
Flag Codes    2151 non-null object
Flags         2151 non-null object
dtypes: float64(1), int64(2), object(6)
memory usage: 473.3+ KB
```

None

Preprocessing of Datasets

Cleaning Educational Dataset and Educational Population

In [5]:
```
#Filtering using Indicators
ed = ed[ed.Indicator == 'Youth illiterate population, 15-24 years, both sexes (number)']

#Filtering & renaming important columns
ed = ed.drop(['EDULIT_IND','Indicator','LOCATION','Time','Flag Codes','Flags'],axis =1).rename(columns={'Country':'country','TI
ME':'year','Value':'illiterate'})

#Drop missing values
dropped = ed[ed['illiterate'] >= 0]

#Getting the mean of the available data
max_ed = dropped.groupby(['country']).mean()

#Making sure year is int
max_ed['year'] = max_ed['year'].apply(math.trunc)

#Reseting index
max_ed = max_ed.reset_index()

#Filtering & renaming important columns
pop = pop[(pop.Indicator == 'School age population, upper secondary education, both sexes (number)') | (pop.Indicator =='School
age population, tertiary education, both sexes (number)')]
pop = pop.drop(['EDULIT_IND','Indicator','LOCATION','Time','Flag Codes','Flags'],axis =1).rename(columns={'Country':'country',
'TIME':'year','Value':'pop'})
pop = pop.groupby(['country','year'],as_index = False).sum()

#Merging educational data
edu_df = max_ed.merge(pop,on = ['country','year'])
edu_df['illiterate%'] = (edu_df['illiterate'] * 100) / (edu_df['pop'])
edu_df.drop([46,75,123],axis=0,inplace = True)
edu_df = edu_df.reset_index(drop=True)
edu_df['ISO'] = 'ISO'
edu_df["continent"] = 'continent'


for i in range(len(edu_df)):
    if edu_df['country'][i] == 'Sint Maarten':
        edu_df['ISO'][i] = 'NLSX'
        edu_df["continent"][i] = 'Europe'
    elif edu_df['country'][i] == 'North Korea':
        edu_df['ISO'][i] = 'PRK'
        edu_df["continent"][i] = 'Asia'
    else:
        edu_df['ISO'][i] = get_country_details(edu_df['country'][i])[0]
        edu_df['continent'][i] = get_country_details(edu_df['country'][i])[1]
```

Exploring Cleaned Datasets

Using .head(),.describe() and .info() methods of pandas

In [6]:
```
display(edu_df.head())
display(edu_df.describe())
display(edu_df.info())
```

	country	year	illiterate	pop	illiterate%	ISO	continent
0	Afghanistan	2014	2.933132e+06	5533145.0	53.010214	AFG	Asia
1	Albania	2013	4.603667e+03	432889.0	1.063475	ALB	Europe
2	Algeria	2018	1.562170e+05	4809336.0	3.248203	DZA	Africa
3	Angola	2014	1.164512e+06	4301328.0	27.073313	AGO	Africa
4	Argentina	2013	3.738562e+04	5633922.0	0.663581	ARG	South America

	year	illiterate	pop	illiterate%
count	145.000000	1.450000e+02	1.450000e+02	145.000000
mean	2013.896552	6.996892e+05	5.960092e+06	14.528252
std	1.544363	2.574294e+06	2.253124e+07	22.743539
min	2010.000000	2.000000e+00	2.439000e+03	0.000041
25%	2013.000000	2.799000e+03	3.005410e+05	0.899710
50%	2014.000000	3.738562e+04	1.087928e+06	2.274229
75%	2014.000000	3.659450e+05	3.878108e+06	16.961934
max	2018.000000	2.660088e+07	2.180239e+08	87.632241

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 145 entries, 0 to 144
Data columns (total 7 columns):
country        145 non-null object
year           145 non-null int64
illiterate     145 non-null float64
pop            145 non-null float64
illiterate%    145 non-null float64
ISO            145 non-null object
continent      145 non-null object
dtypes: float64(3), int64(1), object(3)
memory usage: 8.0+ KB
```

```
None
```

Cleaning COVID-19 and World Population & Density Datasets

In [7]:

```
#world population and country information dataframe

#Renaming columns in worldpop df to simplify use
worldpop.rename(columns={'Country (or dependent territory)':'country','Population':'pop','Density pop./km2':'density pop/km2'},
inplace=True)

#Selecting columns of interest
worldpop = worldpop[['country','pop','density pop/km2']]

#Adding columns to use as reference
worldpop["ISO"] = 'ISO'
worldpop["continent"] = 'continent'
for i in range(len(worldpop)):
    if worldpop['country'][i] == 'Sint Maarten':
        worldpop['ISO'][i] = 'NLSX'
        worldpop["continent"][i] = 'Europe'
    elif worldpop['country'][i] == 'North Korea':
        worldpop['ISO'][i] = 'PRK'
        worldpop["continent"][i] = 'Asia'
    else:
        worldpop['ISO'][i] = get_country_details(worldpop['country'][i])[0]
        worldpop['continent'][i] = get_country_details(worldpop['country'][i])[1]
worldpop['density pop/km2'] = worldpop.apply(lambda x: convert(x['density pop/km2']),axis=1)
worldpop['pop'] = worldpop.apply(lambda x: convert(x['pop']),axis=1)

#Selecting columns of interest
worldpop = worldpop[['ISO','pop','density pop/km2','continent']]
worldpop.drop(191,axis = 0,inplace=True)
worldpop.drop(124,axis = 0,inplace=True)


#confirmed cases dataframe cleaning

#dropping columns insted of selecting many columns of interest
confirmed_cases.drop(['Lat','Long','Province/State'],axis = 1,inplace=True)

#Renaming columns in confirmed cases df to simplify use
confirmed_cases.rename(columns={'Country/Region':'country'},inplace=True)

#Creating column ISO for referencing
confirmed_cases['ISO'] ='ISO'
for i in range(len(confirmed_cases)):
    confirmed_cases['ISO'][i] = get_country_details(confirmed_cases['country'][i])[0]

#transforming df through groupby and melt to reshape date columns
confirmed_cases = confirmed_cases.groupby(['country','ISO'],as_index=False).sum()
confirmed_cases = confirmed_cases.melt(id_vars=['country','ISO'],var_name='Date',value_name='confirmed')

#creating catagorical column to simplify distribution analysis
confirmed_cases['confirmed_cat'] = 'BASE'
for i in range(len(confirmed_cases)):
    confirmed_cases['confirmed_cat'][i] = count_cat(confirmed_cases['confirmed'][i])
confirmed_cases['confirmed_cat'] = pd.Categorical(confirmed_cases['confirmed_cat'],categories=['< 25','< 50','< 100','< 200','<
1000','< 5000','< 10,000','< 30,000','< 100,000','< 150,000','< 200,000','< 250,000','< 300,000','< 400,000','< 500,000','> 50
0,000'],ordered=True)


#death cases dataframe

#dropping columns insted of selecting many columns of interest
death_cases.drop(['Lat','Long','Province/State'],axis = 1,inplace=True)

#Renaming columns in death cases df to simplify use
death_cases.rename(columns={'Country/Region':'country'},inplace=True)

#transforming df through groupby and melt to reshape date columns
death_cases = death_cases.groupby('country',as_index=False).sum()
death_cases = death_cases.melt(id_vars='country',var_name='Date',value_name='death')

#creating catagorical column to simplify distribution analysis
death_cases['death_cat'] = 'BASE'
for i in range(len(death_cases)):
    death_cases['death_cat'][i] = count_cat(death_cases['death'][i])
death_cases['death_cat'] = pd.Categorical(death_cases['death_cat'],categories=['< 25','< 50','< 100','< 200','< 1000','< 5000',
'< 10,000','< 30,000','< 100,000','< 150,000','< 200,000','< 250,000','< 300,000','< 400,000','< 500,000','> 500,000'],ordered=
True)


#recovered cases dataframe

#dropping columns insted of selecting many columns of interest
recovered_cases.drop(['Lat','Long','Province/State'],axis = 1,inplace=True)

#Renaming columns in recovered cases df to simplify use
recovered_cases.rename(columns={'Country/Region':'country'},inplace=True)

#transforming df through groupby and melt to reshape date columns
recovered_cases = recovered_cases.groupby('country',as_index=False).sum()
recovered_cases = recovered_cases.melt(id_vars='country',var_name='Date',value_name='recovered')

#creating catagorical column to simplify distribution analysis
recovered_cases['recovered_cat'] = 'BASE'
```

```
for i in range(len(recovered_cases)):
    recovered_cases['recovered_cat'][i] = count_cat(recovered_cases['recovered'][i])
recovered_cases['recovered_cat'] = pd.Categorical(recovered_cases['recovered_cat'],categories=['< 25','< 50','< 100','< 200','< 1000','< 5000','< 10,000','< 30,000','< 100,000','< 150,000','< 200,000','< 250,000','< 300,000','< 400,000','< 500,000','> 500,000'],ordered=True)

# Main df (full_df) with combined dfs using merge

#using ISO as base to reference worldpop df
full_df = confirmed_cases
full_df = full_df.merge(death_cases)
full_df = full_df.merge(recovered_cases)

#merging on ISO
full_df = full_df.merge(worldpop,on = 'ISO')

#converting date column to date object
full_df.Date = pd.to_datetime(full_df.Date,format = '%m/%d/%y')

#creating active cases column and its catagorical column to simplify distribution analysis
#initializing columns
full_df['active'] = 0
full_df['active_cat'] = 'BASE'

#Calculating values
for i in range(len(full_df)):
    full_df['active'][i] = (full_df['confirmed'][i]) - (full_df['death'][i] + full_df['recovered'][i])
    full_df['active_cat'][i] = count_cat(full_df['active'][i])

#Categorising column
full_df['active_cat'] = pd.Categorical(full_df['active_cat'],categories=['< 25','< 50','< 100','< 200','< 1000','< 5000','< 10,000','< 30,000','< 100,000','< 150,000','< 200,000','< 250,000','< 300,000','< 400,000','< 500,000','> 500,000'],ordered=True)


#adding mortality percentage, confirmed cases to population percentage and active to confirmed percentage
#initializing columns
full_df['mortality%'] = 'mort'
full_df['confirmed%'] = 'per'
full_df['active%'] = 'perc'

#creating columns
for i in range(len(full_df)):
    full_df['confirmed%'][i] = round(((100 * full_df.confirmed[i]) / full_df['pop'][i]), 4)
    if full_df.confirmed[i] == 0:
        full_df['mortality%'][i] = 0
        full_df['active%'][i] = 0
    else:
        full_df['mortality%'][i] = (100 * full_df.death[i]) / full_df.confirmed[i]
        full_df['active%'][i] = full_df['active'][i] * 100 / full_df['confirmed'][i]


#Converting Datatypes of added columns to floats
full_df['mortality%'] = pd.to_numeric(full_df['mortality%'], downcast="float")
full_df['confirmed%'] = pd.to_numeric(full_df['confirmed%'], downcast="float")
full_df['active%'] = pd.to_numeric(full_df['active%'], downcast="float")
```

Exploring Cleaned Datasets

Using .head(),.describe() and .info() methods of pandas

In [8]:

```
display(full_df.head())
display(full_df.describe())
display(full_df.info())
```

	country	ISO	Date	confirmed	confirmed_cat	death	death_cat	recovered	recovered_cat	pop	density pop/km2	continent	active	active_cat	mortali
0	Afghanistan	AFG	2020-01-22	0	< 25	0	< 25	0	< 25	31575018.0	49.0	Asia	0	< 25	
1	Afghanistan	AFG	2020-01-23	0	< 25	0	< 25	0	< 25	31575018.0	49.0	Asia	0	< 25	
2	Afghanistan	AFG	2020-01-24	0	< 25	0	< 25	0	< 25	31575018.0	49.0	Asia	0	< 25	
3	Afghanistan	AFG	2020-01-25	0	< 25	0	< 25	0	< 25	31575018.0	49.0	Asia	0	< 25	
4	Afghanistan	AFG	2020-01-26	0	< 25	0	< 25	0	< 25	31575018.0	49.0	Asia	0	< 25	

	confirmed	death	recovered	pop	density pop/km2	active	mortality%	confirmed%	active%
count	17290.000000	17290.000000	17290.000000	1.729000e+04	17290.000000	17290.000000	17290.000000	17290.000000	17290.000000
mean	3360.480972	203.269578	855.935223	4.195277e+07	304.249451	2301.276171	1.549112	0.015465	45.713116
std	27349.768229	1797.504260	6474.981487	1.490127e+08	1520.788906	22033.795637	4.436579	0.071694	44.163860
min	0.000000	0.000000	0.000000	3.464100e+04	1.900000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	2.681735e+06	29.000000	0.000000	0.000000	0.000000	0.000000
50%	2.000000	0.000000	0.000000	9.767264e+06	81.000000	2.000000	0.000000	0.000000	50.000000
75%	144.000000	2.000000	10.000000	3.157502e+07	201.000000	118.000000	1.212445	0.002100	92.441864
max	938154.000000	53755.000000	109800.000000	1.401812e+09	18960.000000	784027.000000	100.000000	1.480900	100.000000

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 17290 entries, 0 to 17289
Data columns (total 17 columns):
country            17290 non-null object
ISO                17290 non-null object
Date               17290 non-null datetime64[ns]
confirmed          17290 non-null int64
confirmed_cat      17290 non-null category
death              17290 non-null int64
death_cat          17290 non-null category
recovered          17290 non-null int64
recovered_cat      17290 non-null category
pop                17290 non-null float64
density pop/km2    17290 non-null float64
continent          17290 non-null object
active             17290 non-null int64
active_cat         17290 non-null category
mortality%         17290 non-null float32
confirmed%         17290 non-null float32
active%            17290 non-null float32
dtypes: category(4), datetime64[ns](1), float32(3), float64(2), int64(4), object(3)
memory usage: 2.3+ MB

None
```

Statistical Analysis of COVID-19 Dataset

Correlation analysis of cleaned data

Using .corr() method to find correlations between data knowing that correlation does not necessarily mean causation

In [9]:

```
full_df.corr().style.background_gradient(cmap='Blues')
```

Out[9]:

	confirmed	death	recovered	pop	density pop/km2	active	mortality%	confirmed%	active%
confirmed	1	0.89635	0.662471	0.198961	-0.0128587	0.973463	0.124633	0.233664	0.0633487
death	0.89635	1	0.64811	0.125594	-0.0120006	0.84057	0.181397	0.282065	0.0533526
recovered	0.662471	0.64811	1	0.387923	-0.0132538	0.475564	0.129255	0.219358	-0.00598058
pop	0.198961	0.125594	0.387923	1	-0.0173831	0.12272	0.0490098	-0.0292779	0.0514815
density pop/km2	-0.0128587	-0.0120006	-0.0132538	-0.0173831	1	-0.0110871	-0.0248008	0.067074	0.0342366
active	0.973463	0.84057	0.475564	0.12272	-0.0110871	1	0.101921	0.202566	0.0760375
mortality%	0.124633	0.181397	0.129255	0.0490098	-0.0248008	0.101921	1	0.156226	0.188382
confirmed%	0.233664	0.282065	0.219358	-0.0292779	0.067074	0.202566	0.156226	1	0.139271
active%	0.0633487	0.0533526	-0.00598058	0.0514815	0.0342366	0.0760375	0.188382	0.139271	1

Correlation shows:

- A strong positive correlation between confirmed, death, recovered and active columns (Which to be expected)
- A moderate positive correlation between population and confirmed columns
- A weak negative between density and confirmed columns (which is unexpected) probably due to US high cases count and low density

Ploting CDF for confirmed cases counts

Using ecdf() function and plotly library to plot CDF for the beginning of the outbreak on the 22nd of January and for the last recorded day to compare the spread of data

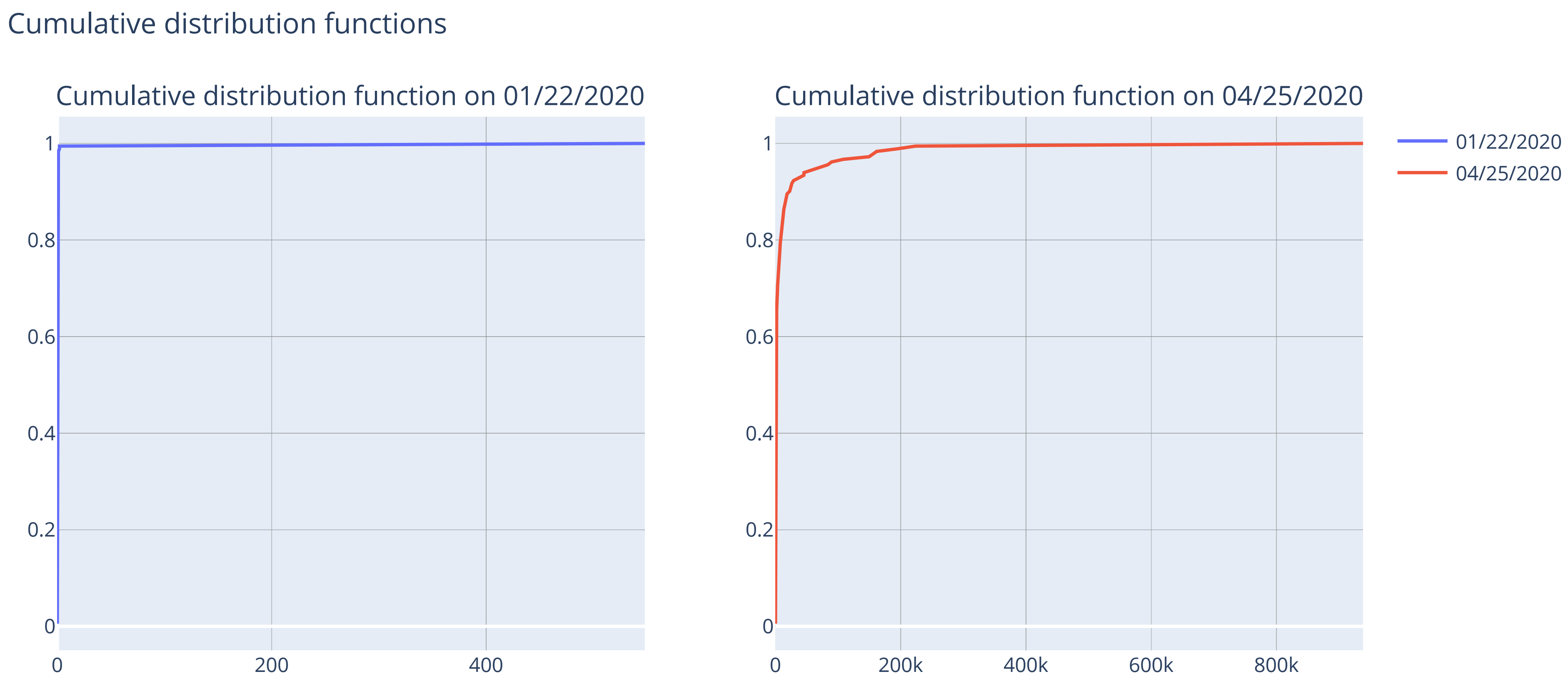
In [10]:
```
#Using ecdf to compute the CDF
x1,y1 = list(ecdf(full_df[(full_df.Date == full_df.Date.min())].confirmed))
x,y = list(ecdf(full_df[(full_df.Date == full_df.Date.max())].confirmed))

#Create a subplot to fit two axis
fig = make_subplots(rows=1, cols=2,subplot_titles=(f'''Cumulative distribution function on {full_df.Date.min().strftime('%m/%d
/%Y')}''', f'''Cumulative distribution function on {full_df.Date.max().strftime("%m/%d/%Y")}'''))

#add first plot at the minimum date recorded
fig.add_trace(
    go.Scatter(x= x1,y = y1,name = f'''{full_df.Date.min().strftime('%m/%d/%Y')}'''),
    row=1, col=1
)

#add second plot at the maximum date recorded
fig.add_trace(
    go.Scatter(x = x,y = y,name = f'''{full_df.Date.max().strftime('%m/%d/%Y')}'''),
    row=1, col=2
)

#control title and figure dimentions
fig.update_layout(height=500, width=1000, title_text="Cumulative distribution functions")
fig.show()
```



CDF shows:

- Spread of data is starting to occure with 95% of the confirmed cases counts are 80K or less

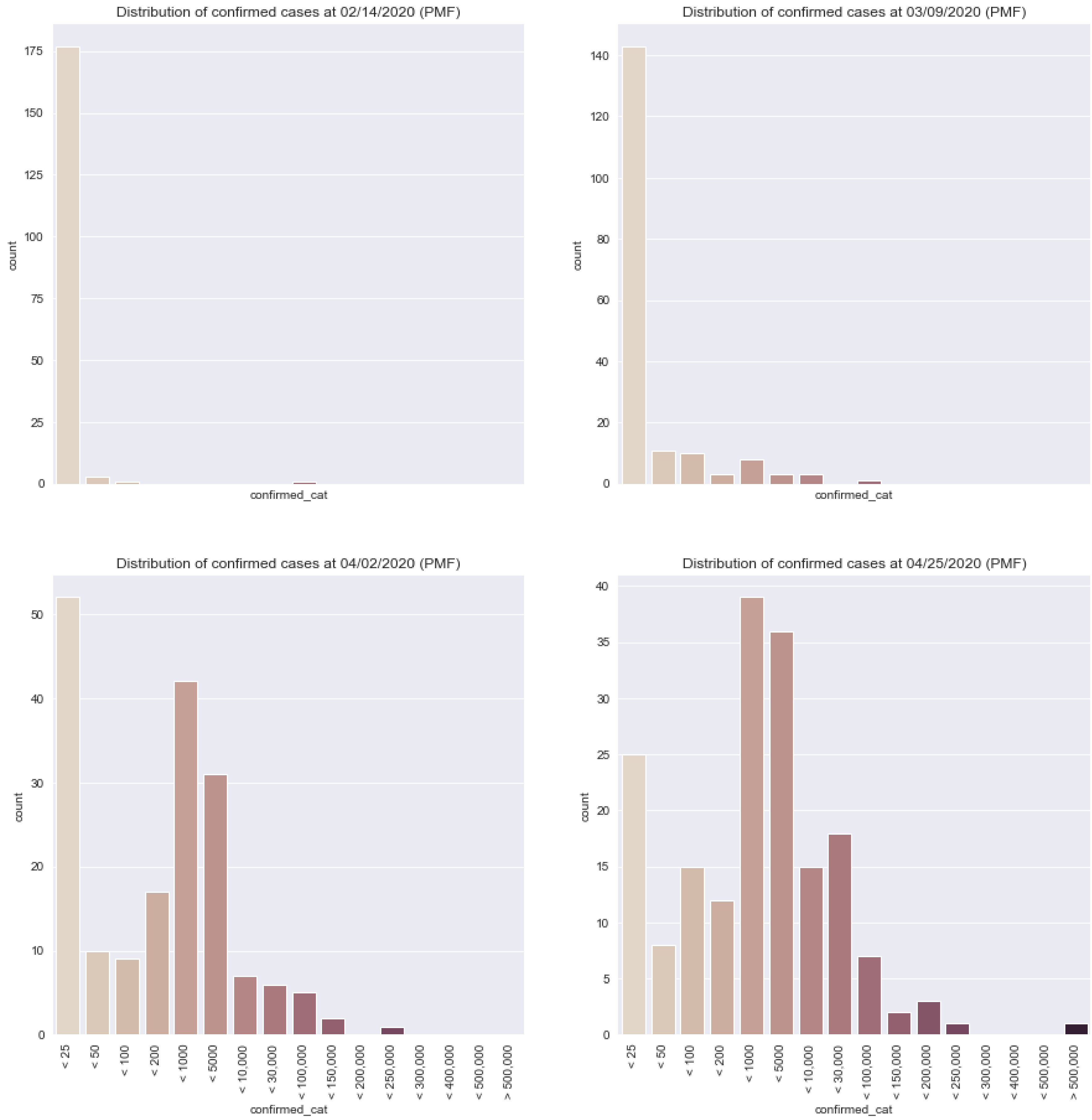
Plotting PMF for categorical confirmed cases count through time:

Plotting probability mass function for confirmed cases distribution through the four quantiles in the data, to observe the evolution of spread over time.

```
In [11]: # Set up the matplotlib figure
         f, axes = plt.subplots(2, 2, figsize=(15, 15),sharex=True);
         _=sns.despine(left=True);


         #distribution of data at 4 quantiles of Dates
         for i , j in {0.25:axes[0,0],0.5:axes[0,1],0.75:axes[1,0],1:axes[1,1]}.items():
             d = full_df[full_df.Date == full_df.Date.quantile(i).strftime('%m/%d/%Y')].sort_values(by = 'confirmed_cat');
             _=sns.catplot(data = d, x="confirmed_cat", kind="count", palette="ch:.25", ax=j);
             _=j.title.set_text(f'''Distribution of confirmed cases at {full_df.Date.quantile(i).strftime('%m/%d/%Y')} (PMF)''');
             plt.close()

         #Rotating x labels
         for axes in f.axes:
              plt.sca(axes)
              plt.xticks(rotation=90)
```



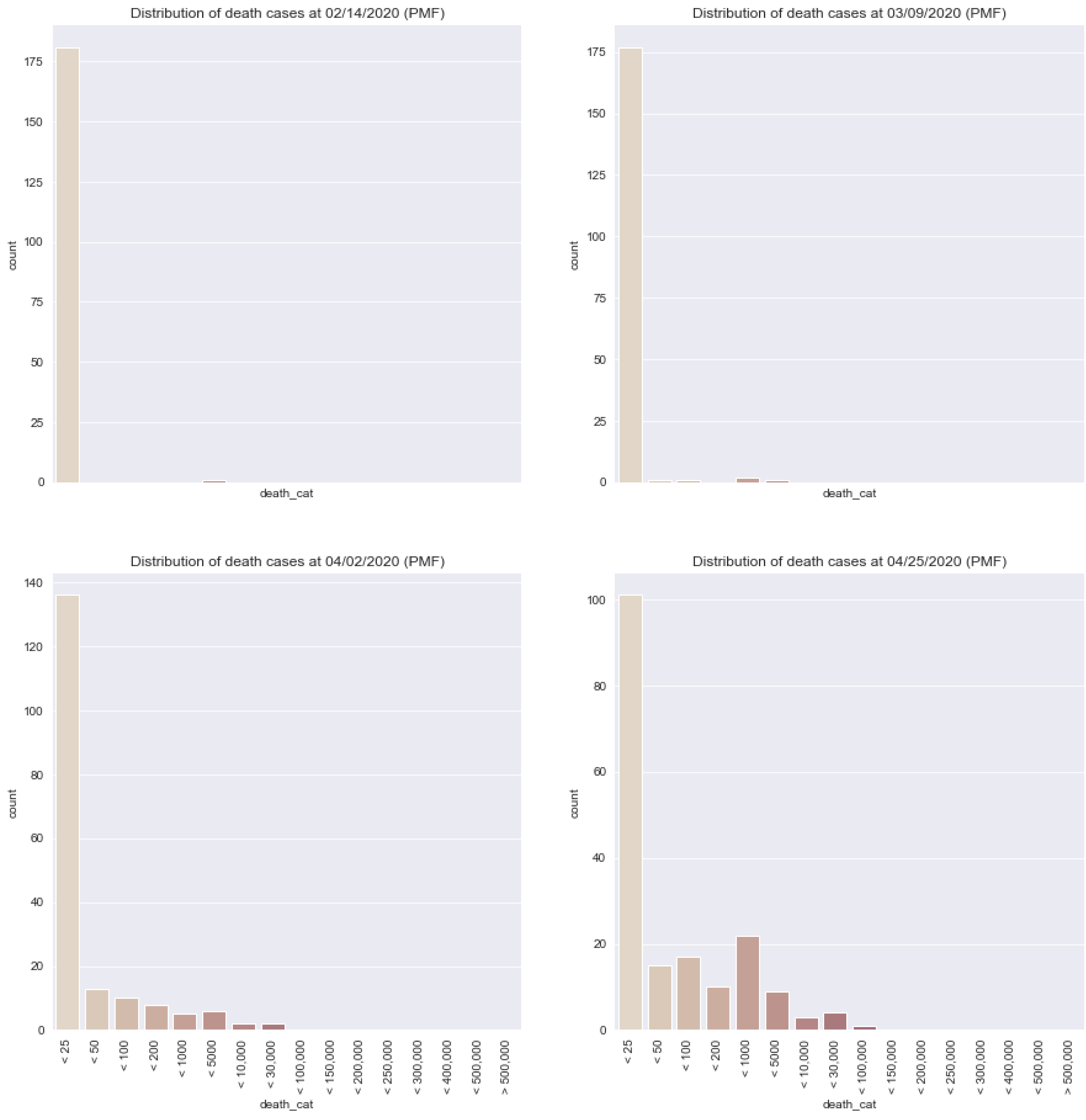
Plotting PMF for categorical death cases count through time:

Plotting probability mass function for death cases distribution through the four quantiles in the data, to observe the evolution of spread over time.

In [12]:
```
# Set up the matplotlib figure
f, axes = plt.subplots(2, 2, figsize=(15, 15),sharex=True);
_=sns.despine(left=True);


#distribution of data at 4 quantiles of Dates
for i , j in {0.25:axes[0,0],0.5:axes[0,1],0.75:axes[1,0],1:axes[1,1]}.items():
    d = full_df[full_df.Date == full_df.Date.quantile(i).strftime('%m/%d/%Y')].sort_values(by = 'death_cat');
    _=sns.catplot(data = d, x="death_cat", kind="count", palette="ch:.25", ax=j);
    _=j.title.set_text(f'''Distribution of death cases at {full_df.Date.quantile(i).strftime('%m/%d/%Y')} (PMF)''');
    plt.close()

#Rotating x labels
for axes in f.axes:
     plt.sca(axes)
     plt.xticks(rotation=90)
```



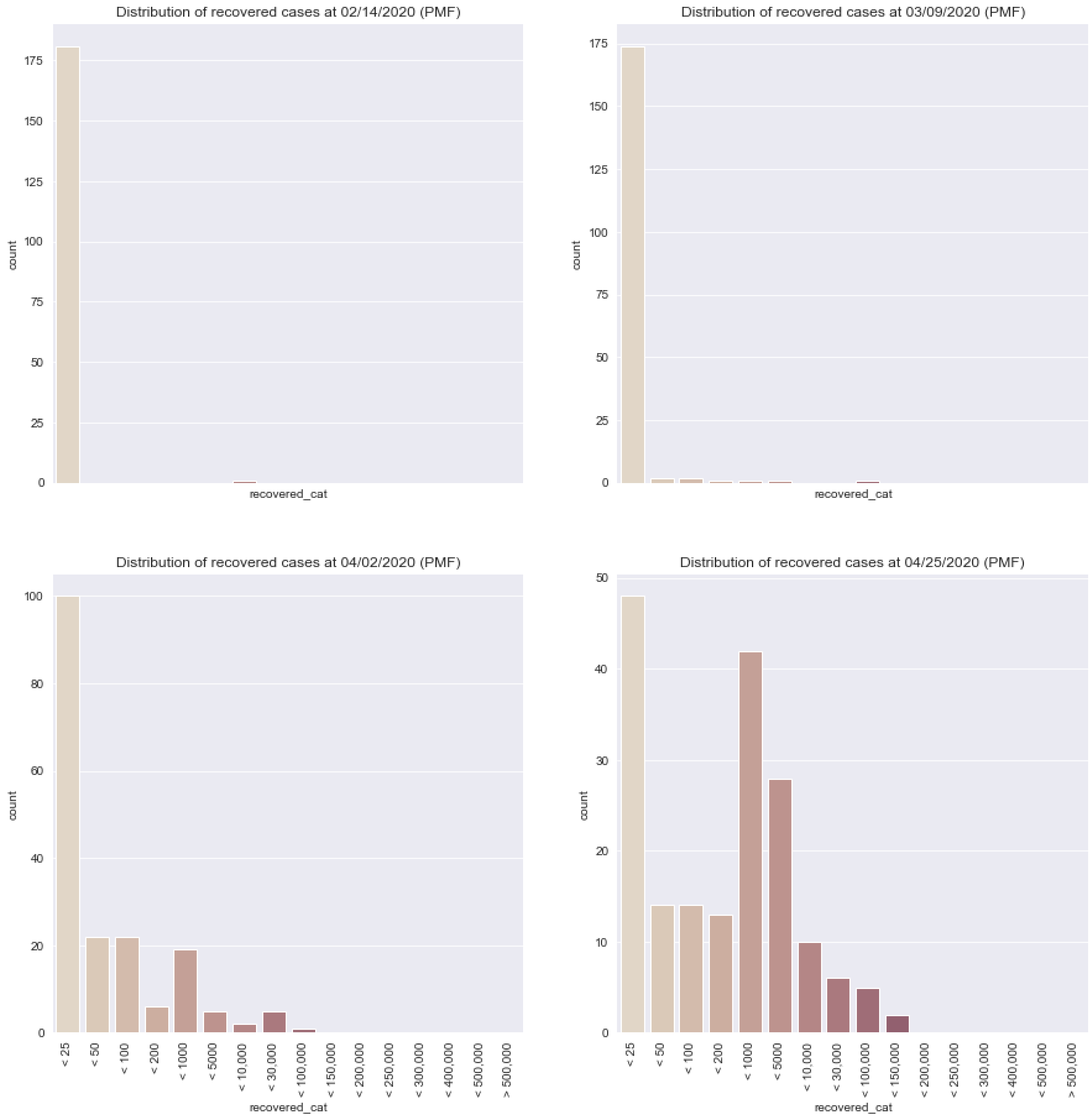
Plotting PMF for categorical recovered cases count through time:

Plotting probability mass function for recovered cases distribution through the four quantiles in the data, to observe the evolution of spread over time.

In [13]:
```python
# Set up the matplotlib figure
f, axes = plt.subplots(2, 2, figsize=(15, 15),sharex=True);
_=sns.despine(left=True);


#distribution of data at 4 quantiles of Dates
for i , j in {0.25:axes[0,0],0.5:axes[0,1],0.75:axes[1,0],1:axes[1,1]}.items():
    d = full_df[full_df.Date == full_df.Date.quantile(i).strftime('%m/%d/%Y')].sort_values(by = 'recovered_cat');
    _=sns.catplot(data = d, x="recovered_cat", kind="count", palette="ch:.25", ax=j);
    _=j.title.set_text(f'''Distribution of recovered cases at {full_df.Date.quantile(i).strftime('%m/%d/%Y')} (PMF)''');
    plt.close()

#Rotating x labels
for axes in f.axes:
     plt.sca(axes)
     plt.xticks(rotation=90)
```



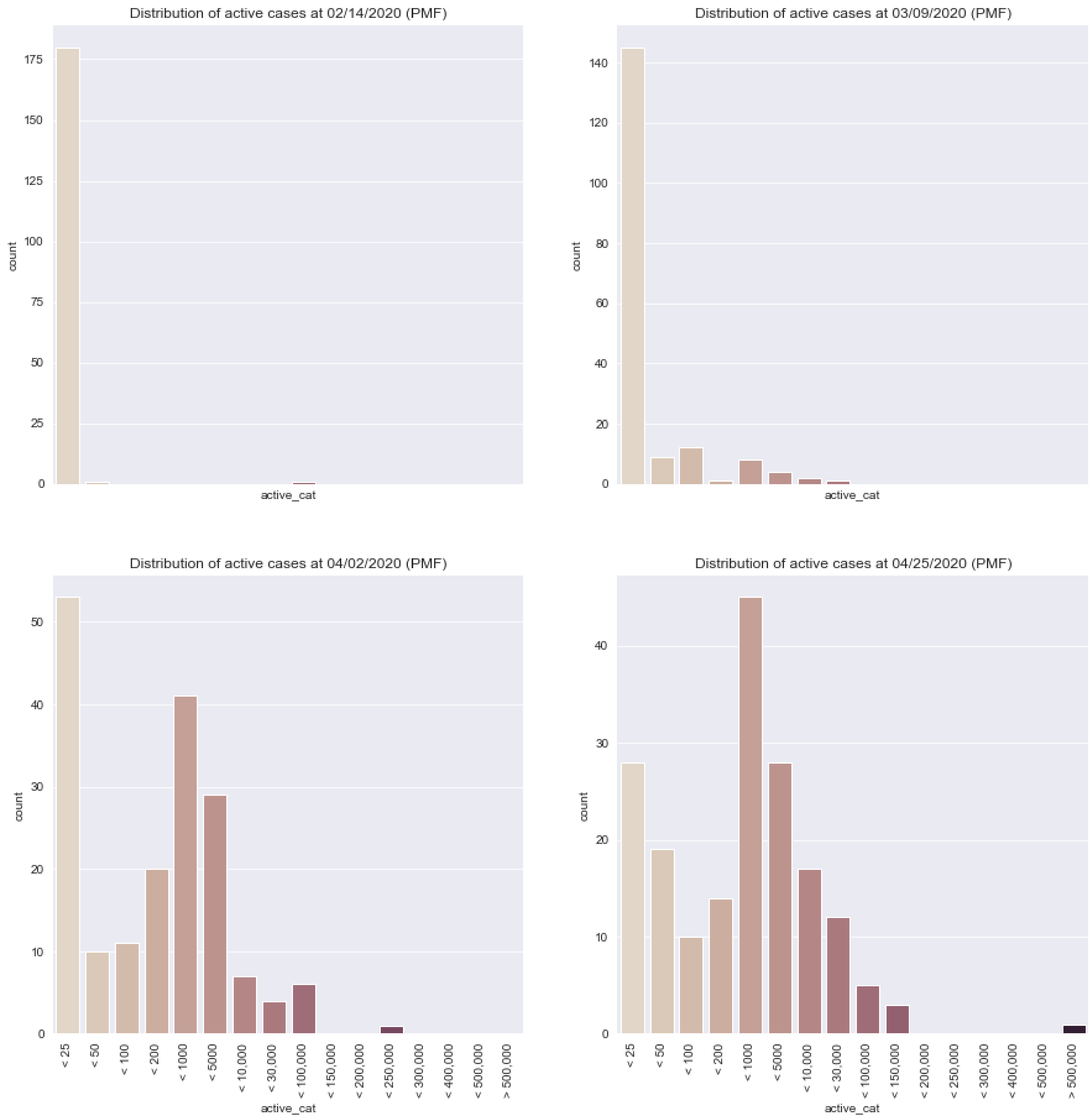
Plotting PMF for categorical active cases count through time:

Plotting probability mass function for active cases distribution through the four quantiles in the data, to observe the evolution of spread over time.

```
In [14]: # Set up the matplotlib figure
f, axes = plt.subplots(2, 2, figsize=(15, 15),sharex=True);
_=sns.despine(left=True);


#distribution of data at 4 quantiles of Dates
for i , j in {0.25:axes[0,0],0.5:axes[0,1],0.75:axes[1,0],1:axes[1,1]}.items():
    d = full_df[full_df.Date == full_df.Date.quantile(i).strftime('%m/%d/%Y')].sort_values(by = 'active_cat');
    _=sns.catplot(data = d, x="active_cat", kind="count", palette="ch:.25", ax=j);
    _=j.title.set_text(f'''Distribution of active cases at {full_df.Date.quantile(i).strftime('%m/%d/%Y')} (PMF)''');
    plt.close()

#Rotating x labels
for axes in f.axes:
     plt.sca(axes)
     plt.xticks(rotation=90)
```



PMF shows:

- Spread of cases and change in their distribution through time, where all cases types spreading to higher counts through short interval of time.

General Data Analysis

Latest Total counts on Global Scale

In [15]:
```
global_latest_count = full_df[full_df.Date == full_df.Date.max()].groupby('Date').sum()[['confirmed','death','recovered','activ
e']]
global_latest_count['mortality%'] = global_latest_count.death * 100 / global_latest_count.confirmed
display(global_latest_count)
```

Date	confirmed	death	recovered	active	mortality%
2020-04-25	2895463	202848	816839	1875776	7.005719

Latest Total counts on Continental Scale

In [16]:
```
global_latest_count = full_df[full_df.Date == full_df.Date.max()].groupby('continent').sum()[['confirmed','death','recovered',
'active']]
global_latest_count['mortality%'] = global_latest_count.death * 100 / global_latest_count.confirmed
display(global_latest_count[global_latest_count.confirmed>1000].style.background_gradient(cmap='Blues',subset=["confirmed"])\
                        .background_gradient(cmap='Reds',subset=["death"])\
                        .background_gradient(cmap='Greens',subset=["recovered"])\
                        .background_gradient(cmap='YlOrBr',subset=["mortality%"])\
                        .background_gradient(cmap='Purples',subset=["active"])
                        )
```

continent	confirmed	death	recovered	active	mortality%
Africa	29748	1394	8729	19625	4.68603
Asia	460651	16953	220763	222935	3.68023
Europe	1252439	120162	407223	725054	9.59424
North America	1013136	58222	125756	829158	5.74671
Oceania	8190	98	6528	1564	1.19658
South America	131250	6019	47826	77405	4.5859

Latest Total counts on National Scale

In [17]:

```
global_latest_count = full_df[full_df.Date == full_df.Date.max()].groupby('country').sum()[['confirmed','death','recovered','active']]
global_latest_count['mortality%'] = global_latest_count.death * 100 / global_latest_count.confirmed
display(global_latest_count.sort_values(by = 'confirmed',ascending = False).style.background_gradient(cmap='Blues',subset=["confirmed"])\
                        .background_gradient(cmap='Reds',subset=["death"])\
                        .background_gradient(cmap='Greens',subset=["recovered"])\
                        .background_gradient(cmap='YlOrBr',subset=["mortality%"])\
                        .background_gradient(cmap='Purples',subset=["active"])
                        )
```

country	confirmed	death	recovered	active	mortality%
US	938154	53755	100372	784027	5.72987
Spain	223759	22902	95708	105149	10.2351
Italy	195351	26384	63120	105847	13.5059
France	161644	22648	45372	93624	14.011
Germany	156513	5877	109800	40836	3.75496
United Kingdom	149569	20381	774	128414	13.6265
Turkey	107773	2706	25582	79485	2.51083
Iran	89328	5650	68193	15485	6.325
China	83909	4636	78175	1098	5.52503
Russia	74588	681	6250	67657	0.913015
Brazil	59324	4057	29160	26107	6.83872
Canada	45491	2547	16013	26931	5.59891
Belgium	45325	6917	10417	27991	15.2609
Netherlands	37384	4424	102	32858	11.8339
Switzerland	28894	1599	21300	5995	5.53402
India	26283	825	5939	19519	3.13891
Peru	25331	700	7797	16834	2.76341
Portugal	23392	880	1277	21235	3.76197
Ecuador	22719	576	1366	20777	2.53532
Ireland	18561	1063	9233	8265	5.72706
Sweden	18177	2192	1005	14980	12.0592
Saudi Arabia	16299	136	2215	13948	0.834407
Israel	15298	199	6435	8664	1.30082
Austria	15148	536	12103	2509	3.53842
Mexico	13842	1305	7149	5388	9.42783
Japan	13231	360	1656	11215	2.72088
Chile	12858	181	6746	5931	1.40768
Pakistan	12723	269	2866	9588	2.11428
Singapore	12693	12	1002	11679	0.0945403
Poland	11273	524	2126	8623	4.64827
Korea, South	10728	242	8717	1769	2.25578
Romania	10635	601	2890	7144	5.65115
United Arab Emirates	9813	71	1887	7855	0.72353
Belarus	9590	67	1573	7950	0.698644
Qatar	9358	10	929	8419	0.10686
Denmark	8643	418	5858	2367	4.83628
Indonesia	8607	720	1042	6845	8.36528
Ukraine	8125	201	782	7142	2.47385
Norway	7499	201	32	7266	2.68036
Czechia	7352	218	2453	4681	2.96518
Philippines	7294	494	792	6008	6.77269
Australia	6694	80	5376	1238	1.1951
Serbia	6630	125	870	5635	1.88537
Dominican Republic	5926	273	822	4831	4.60682
Malaysia	5742	98	3762	1882	1.70672
Panama	5538	159	338	5041	2.87107
Colombia	5142	233	1067	3842	4.53131
Bangladesh	4998	140	113	4745	2.80112
Finland	4475	186	2500	1789	4.15642
South Africa	4361	86	1473	2802	1.97202
Egypt	4319	307	1114	2898	7.10813
Morocco	3897	159	537	3201	4.08006
Argentina	3780	185	1030	2565	4.89418
Luxembourg	3711	85	3088	538	2.29049
Moldova	3304	94	825	2385	2.84504
Algeria	3256	419	1479	1358	12.8686

country	confirmed	death	recovered	active	mortality%
Thailand	2907	51	2547	309	1.75439
Kuwait	2892	19	656	2217	0.656985
Kazakhstan	2601	25	646	1930	0.961169
Bahrain	2588	8	1160	1420	0.309119
Greece	2506	130	577	1799	5.18755
Hungary	2443	262	458	1723	10.7245
Croatia	2016	54	1034	928	2.67857
Oman	1905	10	329	1566	0.524934
Uzbekistan	1862	8	707	1147	0.429646
Iceland	1790	10	1570	210	0.558659
Iraq	1763	86	1224	453	4.87805
Armenia	1677	28	803	846	1.66965
Estonia	1635	46	228	1361	2.81346
Azerbaijan	1617	21	1080	516	1.2987
Cameroon	1518	53	697	768	3.49144
Bosnia and Herzegovina	1486	57	592	837	3.8358
New Zealand	1470	18	1142	310	1.22449
Afghanistan	1463	47	188	1228	3.21258
Lithuania	1426	41	460	925	2.87518
Slovenia	1388	81	219	1088	5.83573
Slovakia	1373	17	386	970	1.23816
North Macedonia	1367	59	374	934	4.31602
Cuba	1337	51	437	849	3.81451
Ghana	1279	10	134	1135	0.781861
Bulgaria	1247	55	197	995	4.41059
Nigeria	1182	35	222	925	2.96108
Djibouti	1008	2	373	633	0.198413
Guinea	996	7	208	781	0.702811
Tunisia	939	38	207	694	4.04686
Bolivia	866	46	54	766	5.31178
Congo (Kinshasa)	832	56	98	678	6.73077
Cyprus	810	14	148	648	1.7284
Latvia	804	12	267	525	1.49254
Andorra	738	40	344	354	5.42005
Albania	712	27	403	282	3.79213
Lebanon	704	24	143	537	3.40909
Costa Rica	693	6	242	445	0.865801
Niger	684	27	325	332	3.94737
Kyrgyzstan	665	8	345	312	1.20301
Burkina Faso	629	41	442	146	6.51828
Honduras	627	59	65	503	9.40989
Senegal	614	7	276	331	1.14007
Uruguay	596	14	370	212	2.34899
San Marino	513	40	64	409	7.79727
Kosovo	510	12	93	405	2.35294
Guatemala	473	13	45	415	2.74841
Sri Lanka	460	7	118	335	1.52174
Georgia	456	5	139	312	1.09649
Malta	448	4	249	195	0.892857
Jordan	444	7	332	105	1.57658
Taiwan*	429	6	275	148	1.3986
Congo (Brazzaville)	400	12	38	350	3
Somalia	390	18	8	364	4.61538
Mali	370	21	91	258	5.67568
Kenya	343	14	98	231	4.08163
West Bank and Gaza	342	2	92	248	0.584795

country	confirmed	death	recovered	active	mortality%
Mauritius	331	9	295	27	2.71903
Venezuela	323	10	132	181	3.09598
Montenegro	320	6	153	161	1.875
Jamaica	305	7	28	270	2.29508
Tanzania	299	10	48	241	3.34448
El Salvador	274	8	75	191	2.91971
Vietnam	270	0	225	45	0
Equatorial Guinea	258	1	7	250	0.387597
Paraguay	228	9	85	134	3.94737
Sudan	213	17	19	177	7.98122
Rwanda	183	0	88	95	0
Maldives	177	0	17	160	0
Gabon	176	3	30	143	1.70455
Burma	146	5	10	131	3.42466
Brunei	138	1	121	16	0.724638
Madagascar	123	0	62	61	0
Ethiopia	122	3	29	90	2.45902
Cambodia	122	0	117	5	0
Liberia	120	11	25	84	9.16667
Trinidad and Tobago	115	8	53	54	6.95652
Togo	96	6	62	28	6.25
Monaco	94	4	42	48	4.25532
Zambia	84	3	37	44	3.57143
Sierra Leone	82	2	10	70	2.43902
Liechtenstein	81	1	55	25	1.23457
Barbados	79	6	31	42	7.59494
Bahamas	78	11	15	52	14.1026
Uganda	75	0	46	29	0
Guyana	73	7	12	54	9.58904
Haiti	72	6	6	60	8.33333
Mozambique	70	0	12	58	0
Libya	61	2	18	41	3.27869
Eswatini	56	1	10	45	1.78571
Benin	54	1	27	26	1.85185
Guinea-Bissau	52	0	3	49	0
Nepal	49	0	12	37	0
Chad	46	0	15	31	0
Syria	42	3	11	28	7.14286
Eritrea	39	0	13	26	0
Mongolia	37	0	9	28	0
Malawi	33	3	4	26	9.09091
Zimbabwe	31	4	2	25	12.9032
Angola	25	2	6	17	8
Antigua and Barbuda	24	3	11	10	12.5
Timor-Leste	24	0	2	22	0
Botswana	22	1	0	21	4.54545
Laos	19	0	7	12	0
Belize	18	2	5	11	11.1111
Grenada	18	0	7	11	0
Fiji	18	0	10	8	0
Namibia	16	0	7	9	0
Dominica	16	0	13	3	0
Central African Republic	16	0	10	6	0
Saint Lucia	15	0	15	0	0
Saint Kitts and Nevis	15	0	2	13	0
Saint Vincent and the Grenadines	14	0	5	9	0

country	confirmed	death	recovered	active	mortality%
Nicaragua	12	3	7	2	25
Burundi	11	1	4	6	9.09091
Seychelles	11	0	6	5	0
Suriname	10	1	7	2	10
Gambia	10	1	8	1	10
Papua New Guinea	8	0	0	8	0
Bhutan	7	0	3	4	0
Mauritania	7	1	6	0	14.2857
Western Sahara	6	0	5	1	0
South Sudan	5	0	0	5	0
Sao Tome and Principe	4	0	0	4	0
Yemen	1	0	1	0	0

Conclusions :

- US has the highest confirmed, deaths and active cases counts.
- Germany has the highest Recovery counts.
- Belgium has the highest Mortality Rate.
- Europe has more counts than North America, South America and Africa combined.

Visualization on Map

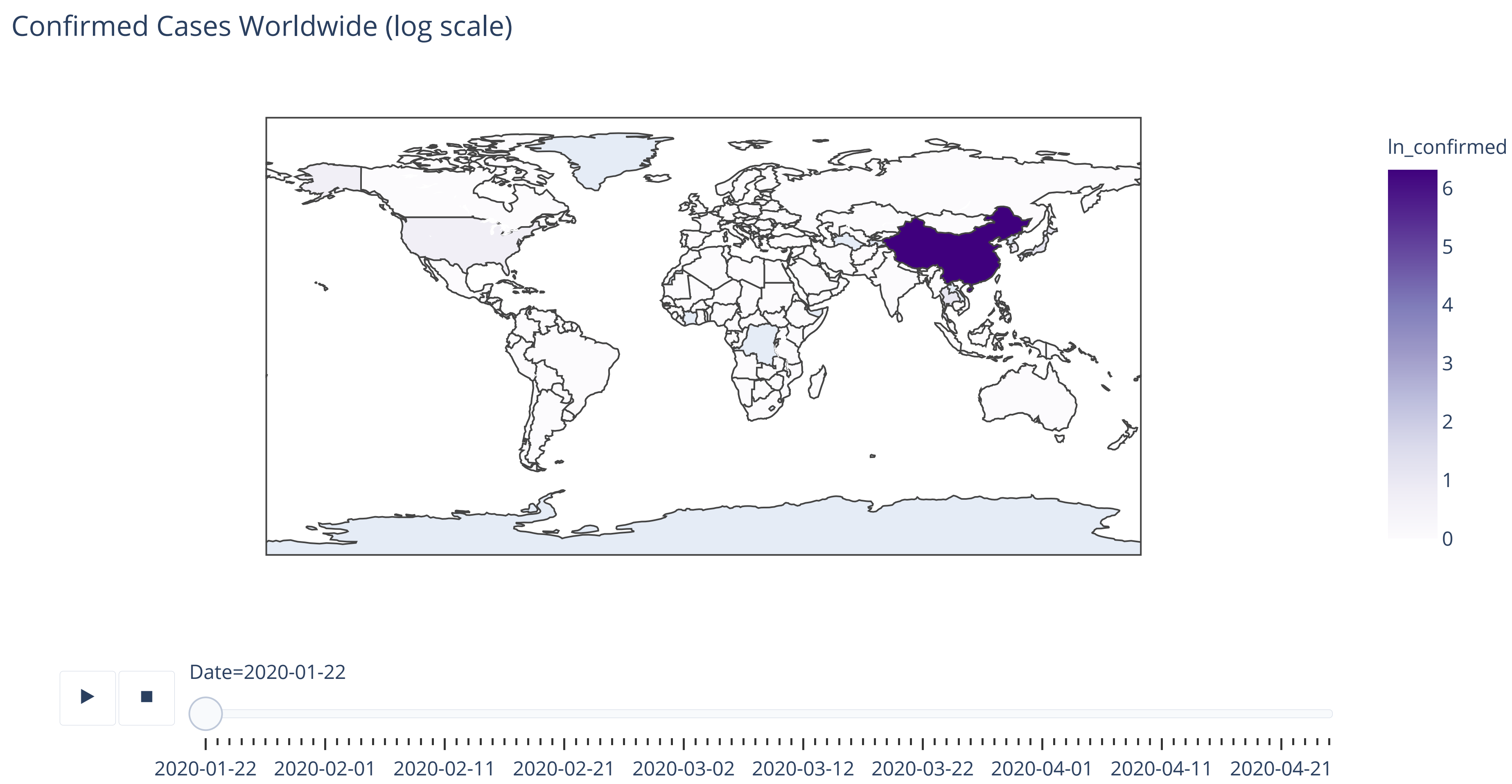
> Since, cases and deaths have grown exponentially over the past three months through out the world, I have plotted the choropleth map on logarithmic scale. You can hover on the country to know the total confirmed cases or deaths.

Subsetting Data for map visualization using plotly_express

In [18]:
```
#subsetting from full_df for mapping with log scale
world_df = full_df.groupby(['country','Date','ISO','mortality%','confirmed%'],as_index=False).sum()
world_df['Date'] = world_df.Date.apply(lambda x: x.date()).apply(str)
world_df['ln_confirmed'] = np.log(world_df.confirmed + 1)
world_df['ln_death'] = np.log(world_df.death + 1)
world_df['ln_recovered'] = np.log(world_df.recovered + 1)
world_df['ln_mortality%'] = np.log(world_df['mortality%'] + 1)
world_df['ln_active'] = np.log(world_df['active'] + 1)
```

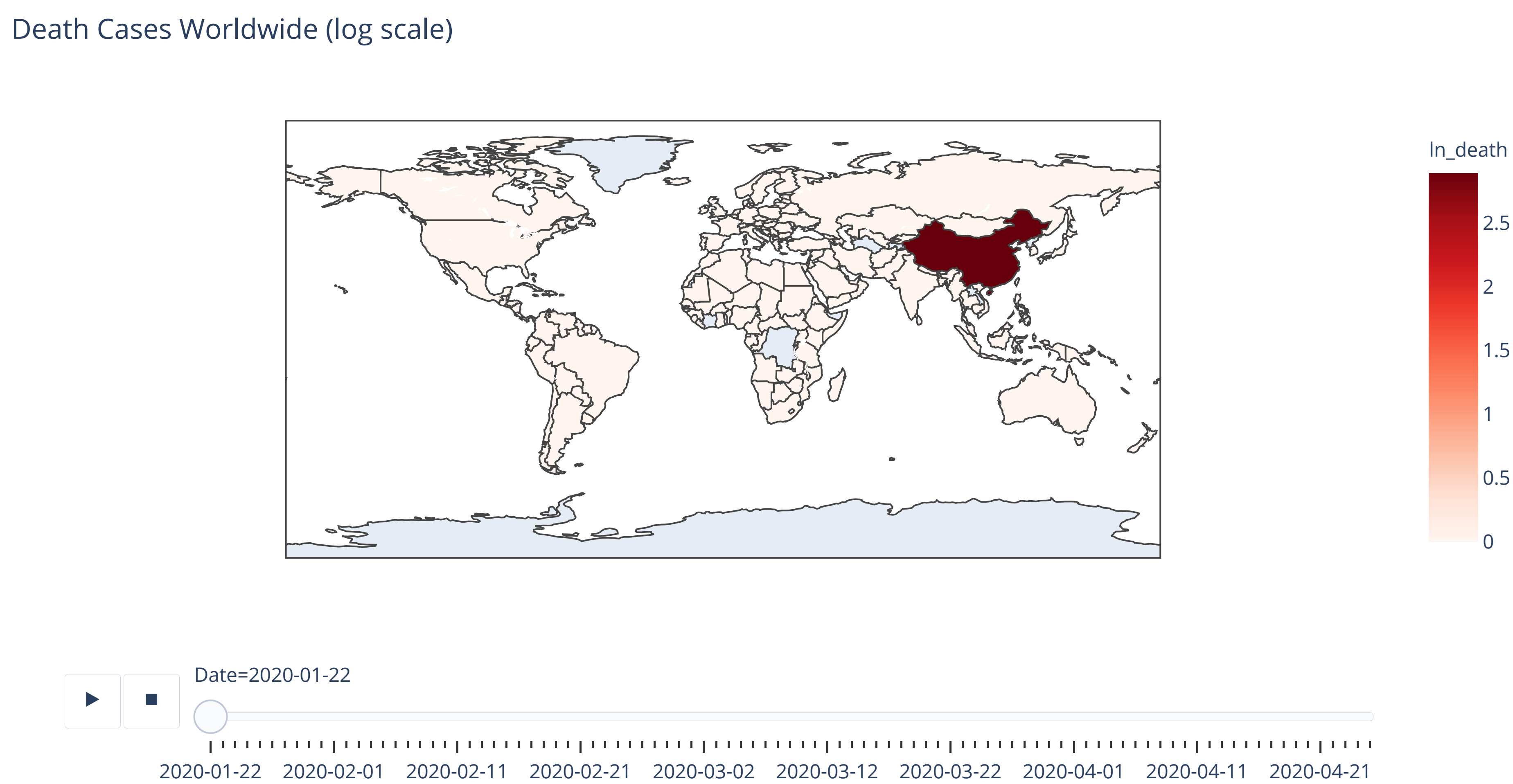
Animated Global Confirmed Cases count through time

```
In [19]: px.choropleth(world_df,
                   locations="ISO",
                   color="ln_confirmed",
                   hover_name="country",
                   hover_data=["death"] ,
                   animation_frame="Date",
                   color_continuous_scale='Purples',title='Confirmed Cases Worldwide (log scale)')
```



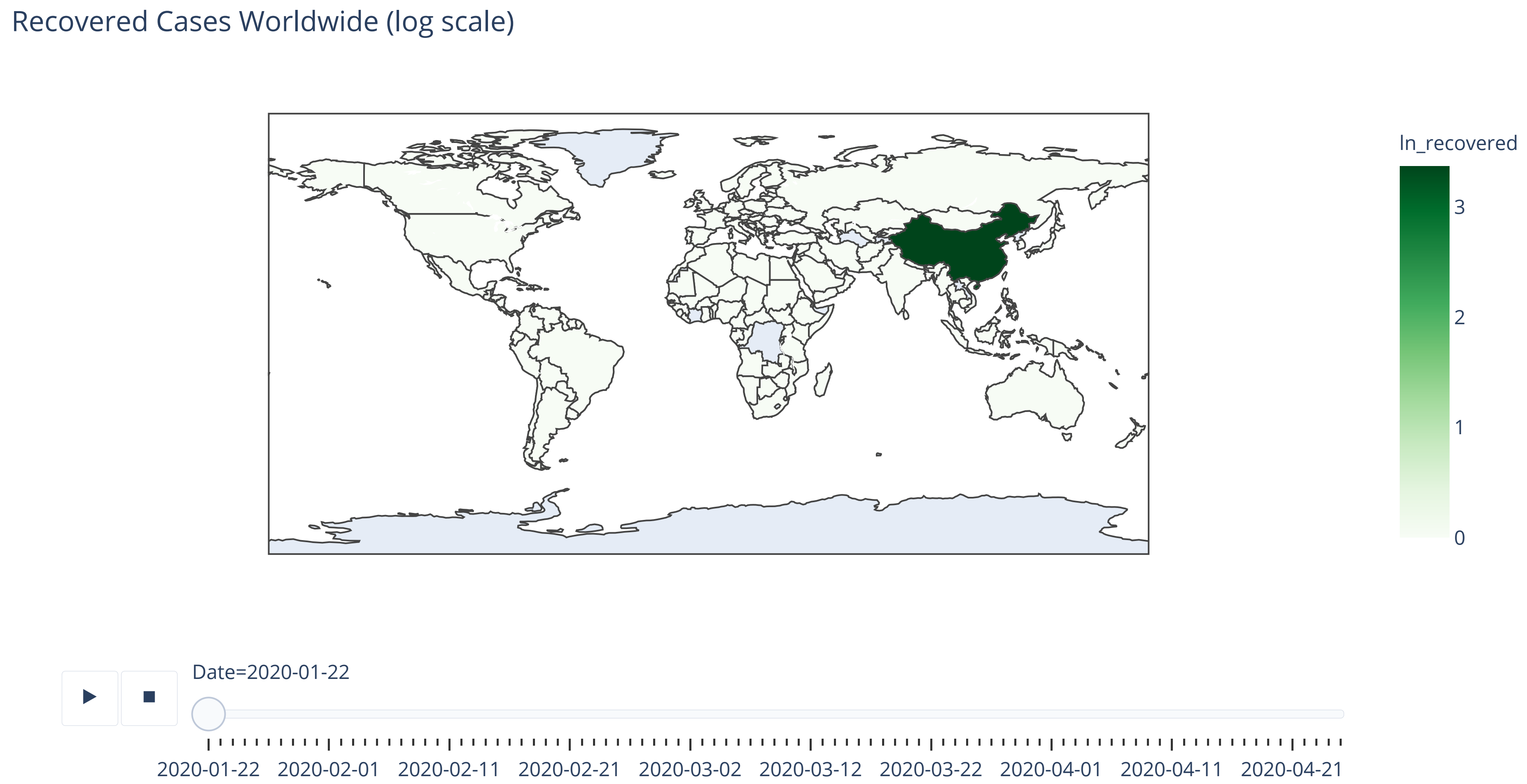
Animated Global Death Cases count through time

```
In [20]: px.choropleth(world_df,
                   locations="ISO",
                   color="ln_death",
                   hover_name="country",
                   hover_data=["recovered"] ,
                   animation_frame="Date",
                   color_continuous_scale='Reds',title='Death Cases Worldwide (log scale)')
```



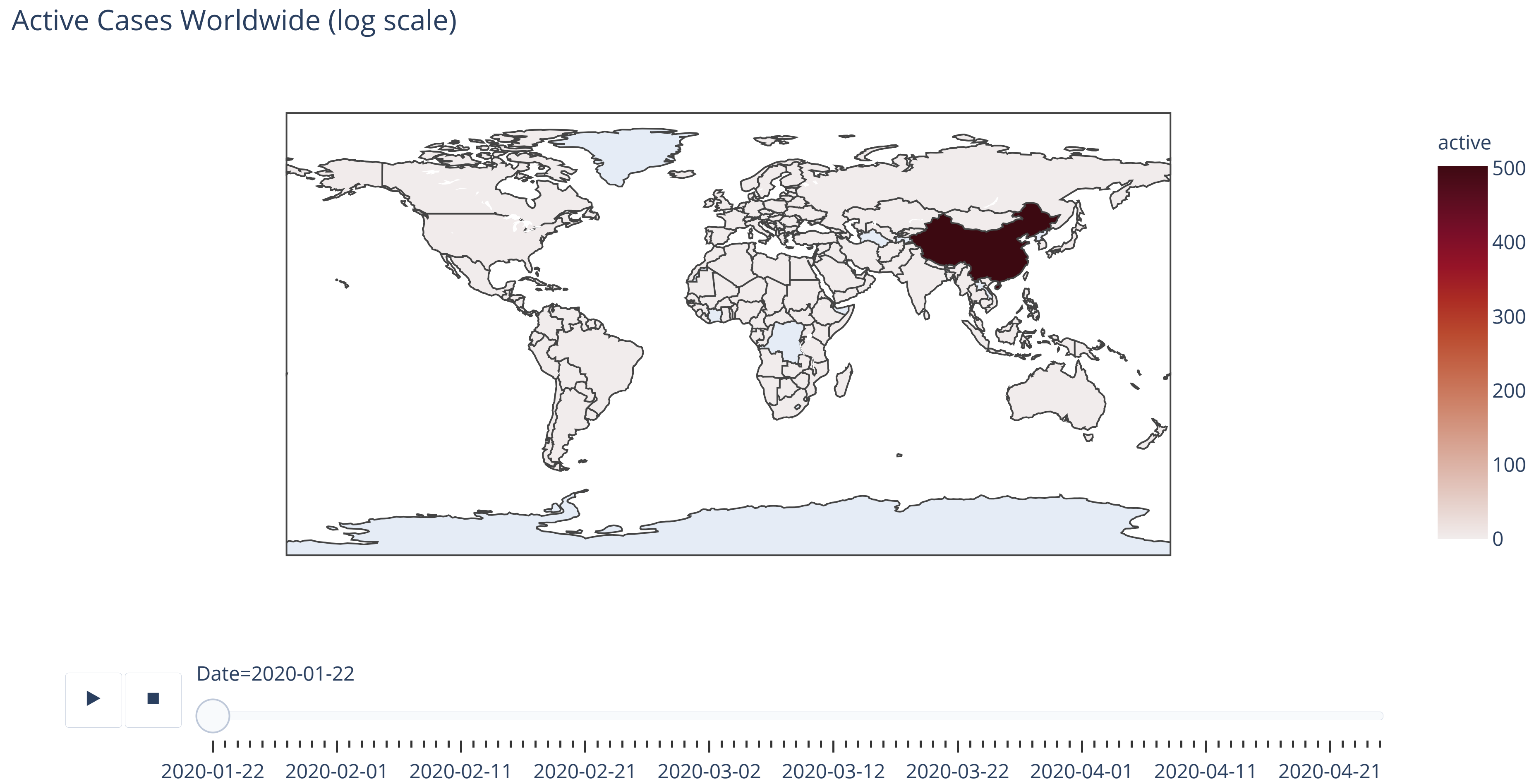
Animated Global Recovered Cases count through time

In [21]:
```
px.choropleth(world_df,
              locations="ISO",
              color="ln_recovered",
              hover_name="country",
              hover_data=["death"] ,
              animation_frame="Date",
              color_continuous_scale='Greens',title='Recovered Cases Worldwide (log scale)')
```



Animated Global Active Cases count through time

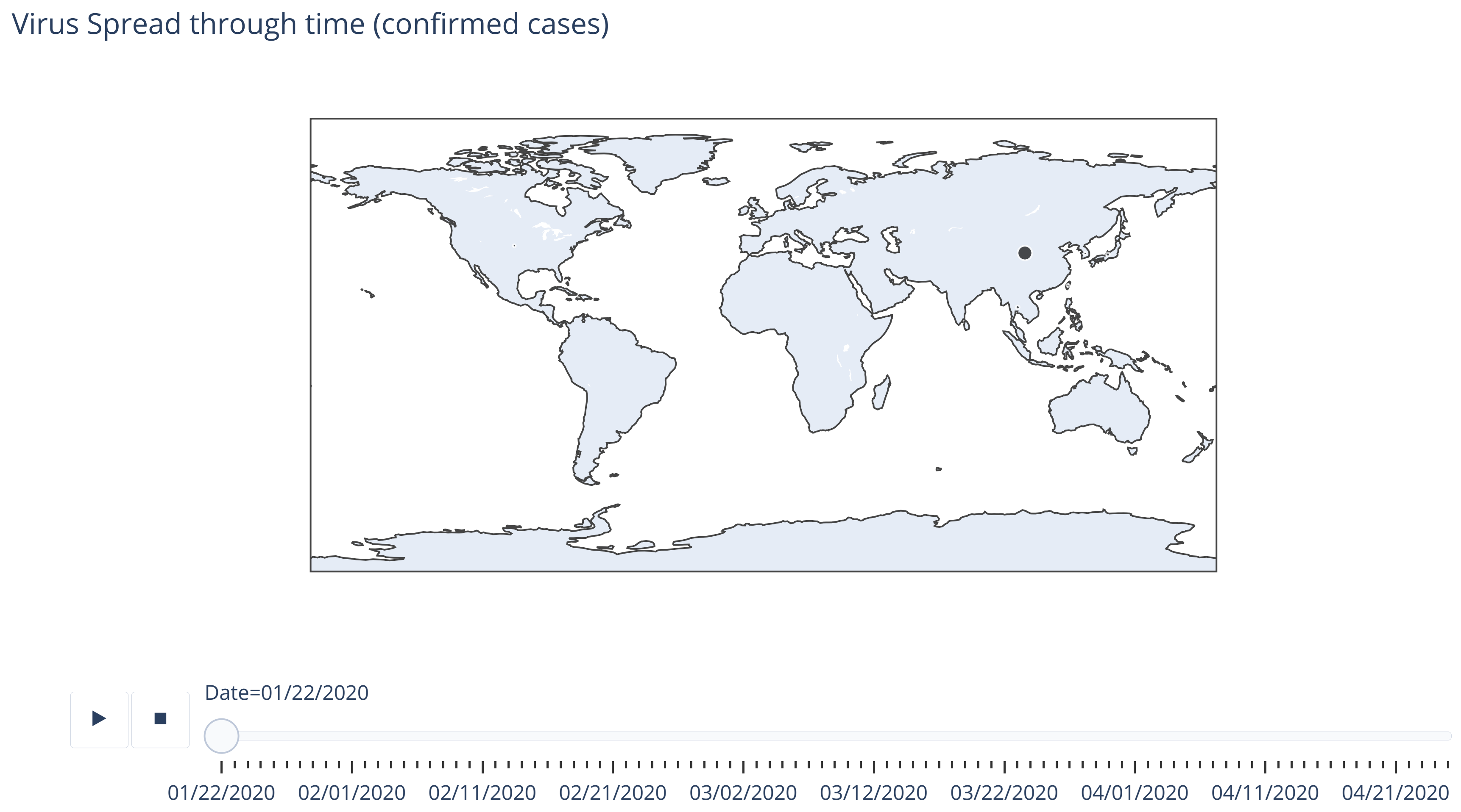
In [22]:
```
px.choropleth(world_df,
              locations="ISO",
              color='active',
              hover_name="country",
              hover_data=["death"] ,
              animation_frame="Date",
              color_continuous_scale='amp',title='Active Cases Worldwide (log scale)')
```



Animated Global Spread through time

In [23]:
```
df_data = full_df.groupby(['country','Date'],as_index = False)['confirmed','death'].max()
df_data["Date"] = pd.to_datetime( df_data["Date"]).dt.strftime('%m/%d/%Y')

fig = px.scatter_geo(df_data, locations="country", locationmode='country names',
                     color=df_data["confirmed"],
                     size= np.power(df_data["confirmed"]+1,0.3)-1,
                     hover_name="country",
                     hover_data=["confirmed"],
                     range_color= [0, max(df_data["confirmed"])-1],
                     animation_frame="Date",
                     color_continuous_scale=px.colors.sequential.Plasma,
                     title='Virus Spread through time (confirmed cases)'
                    )
fig.update_coloraxes(colorscale="hot")
fig.update(layout_coloraxis_showscale=False)
fig.show()
```



Conclusions :

- China was the first country to experience the COVID-19 outbreak.
- US, Spain and Italy, which are the worst affected countries, didn't record almost any cases from the beginning of the outbreak in january. This indicates the fast spread of the virus.
- US and Western Europe are the worst affected. Therefore as the virus spreads from Eastern Asia to Western Europe and US,they are considered the new virus epicenter.
- Partial/Total Lockdown or quarantine in China led to controling spread and managing deaths and activity with minimum active cases and low death rate.

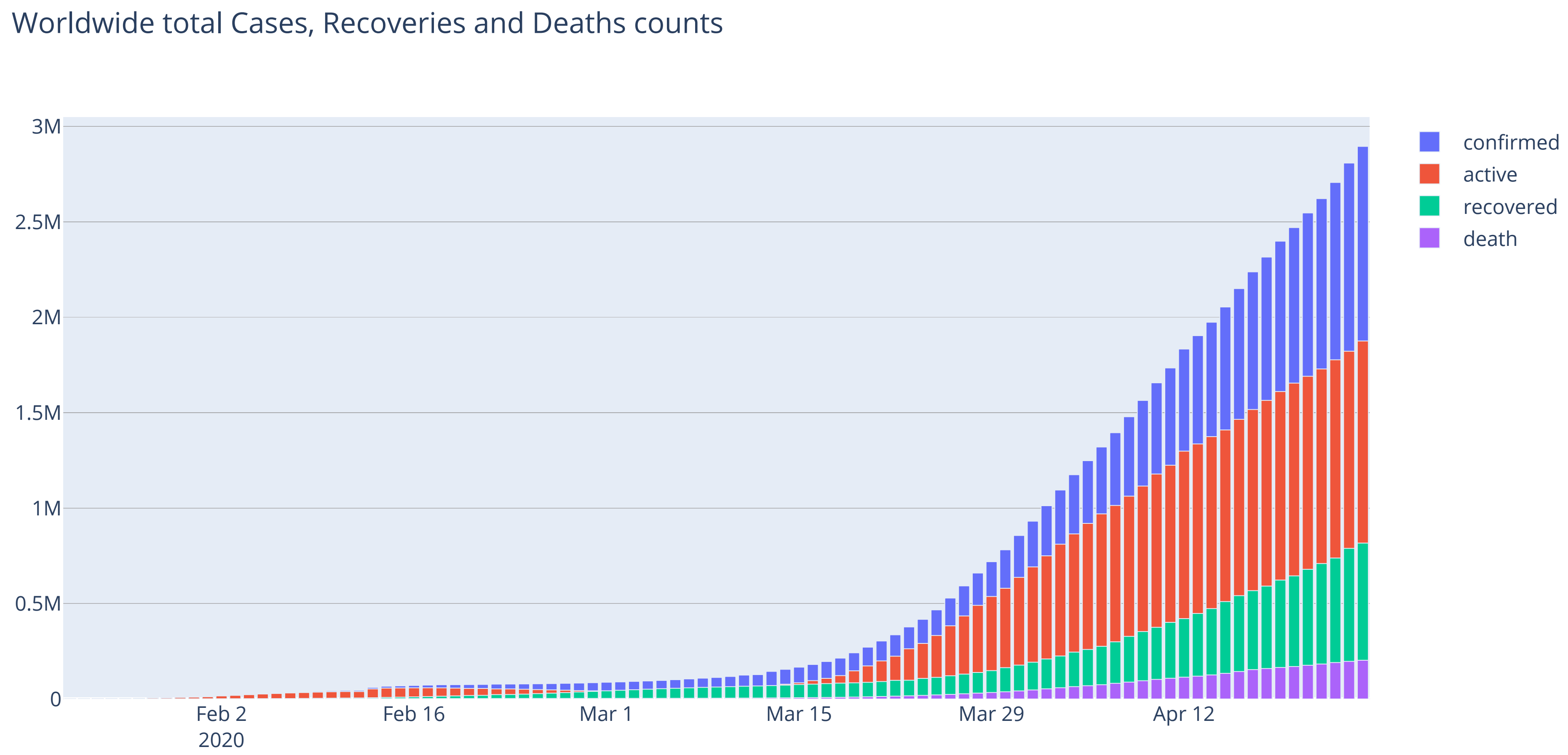
Trend of Cases on Global Scale

Finding top 10 countries affected. Since, the Confirmed cases and Deaths are the cummulative sums till date. Adding daily counts is recommended.

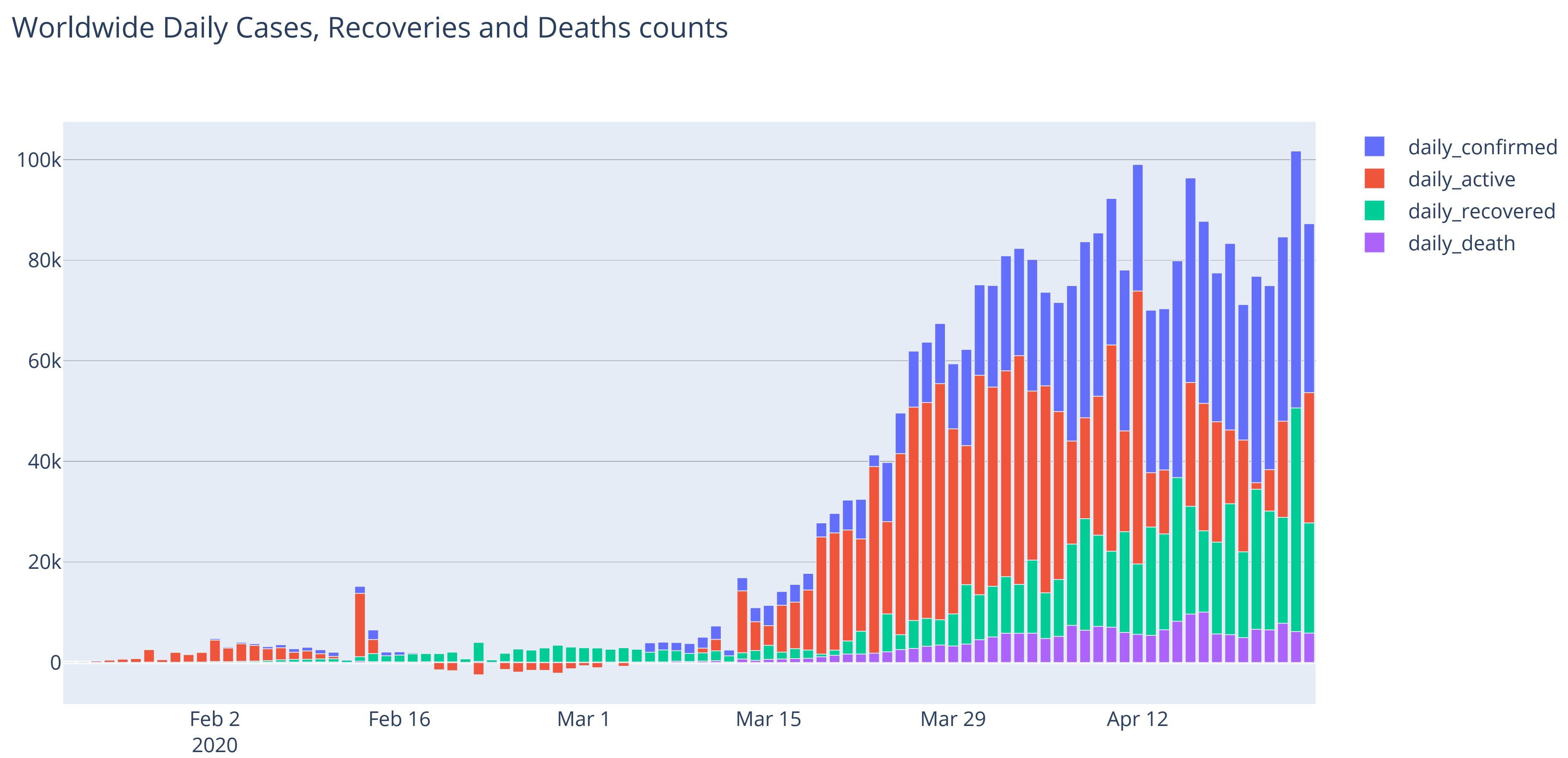
Worldwide Analysis

Starting off with worldwide data analysis using bar plots to get a general sense of how data growth seems to behave.

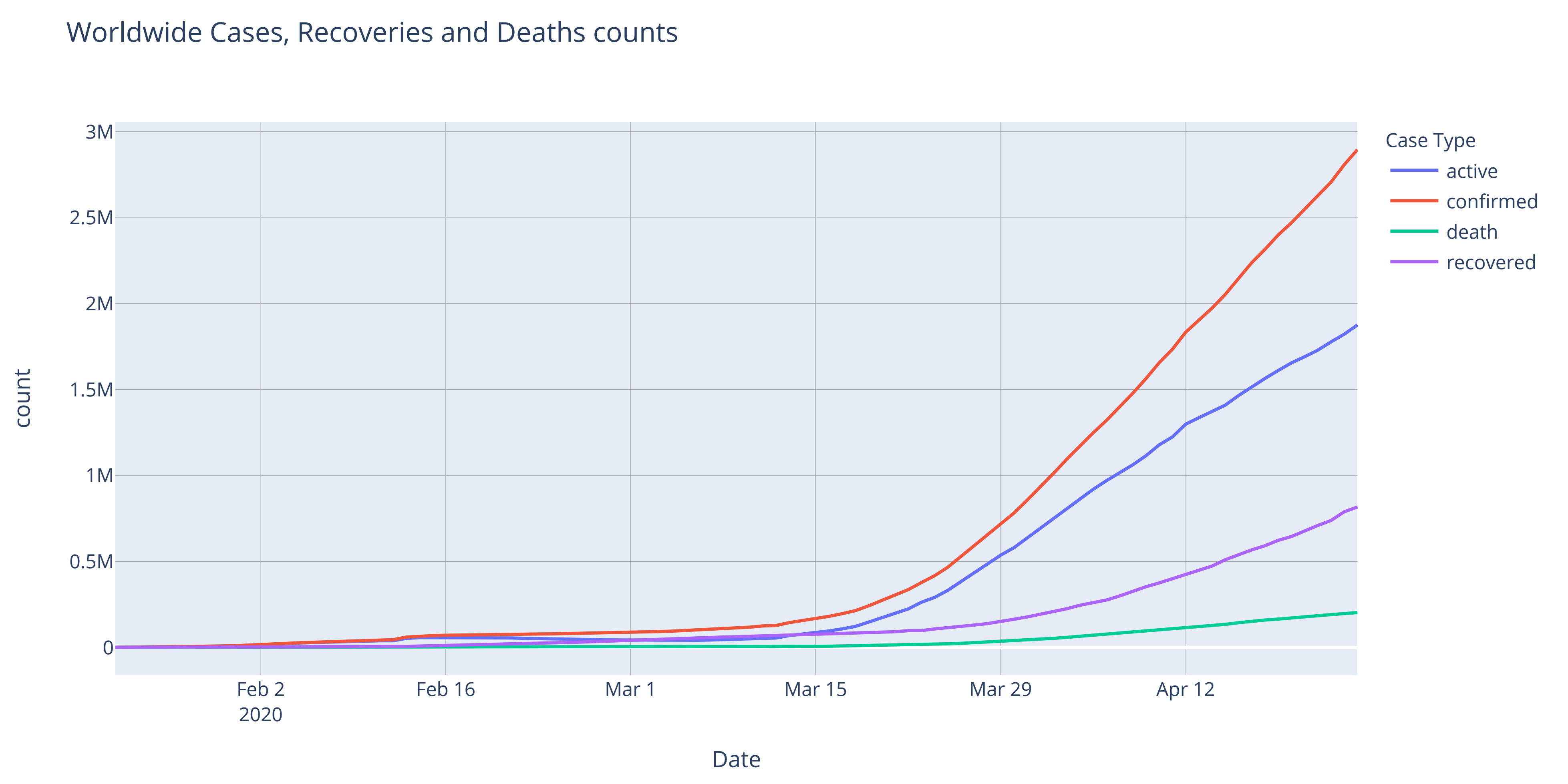
```
In [24]: gplotbar(full_df,cols=['confirmed','active','recovered','death'],title='Worldwide total Cases, Recoveries and Deaths counts',countryname='all')
```



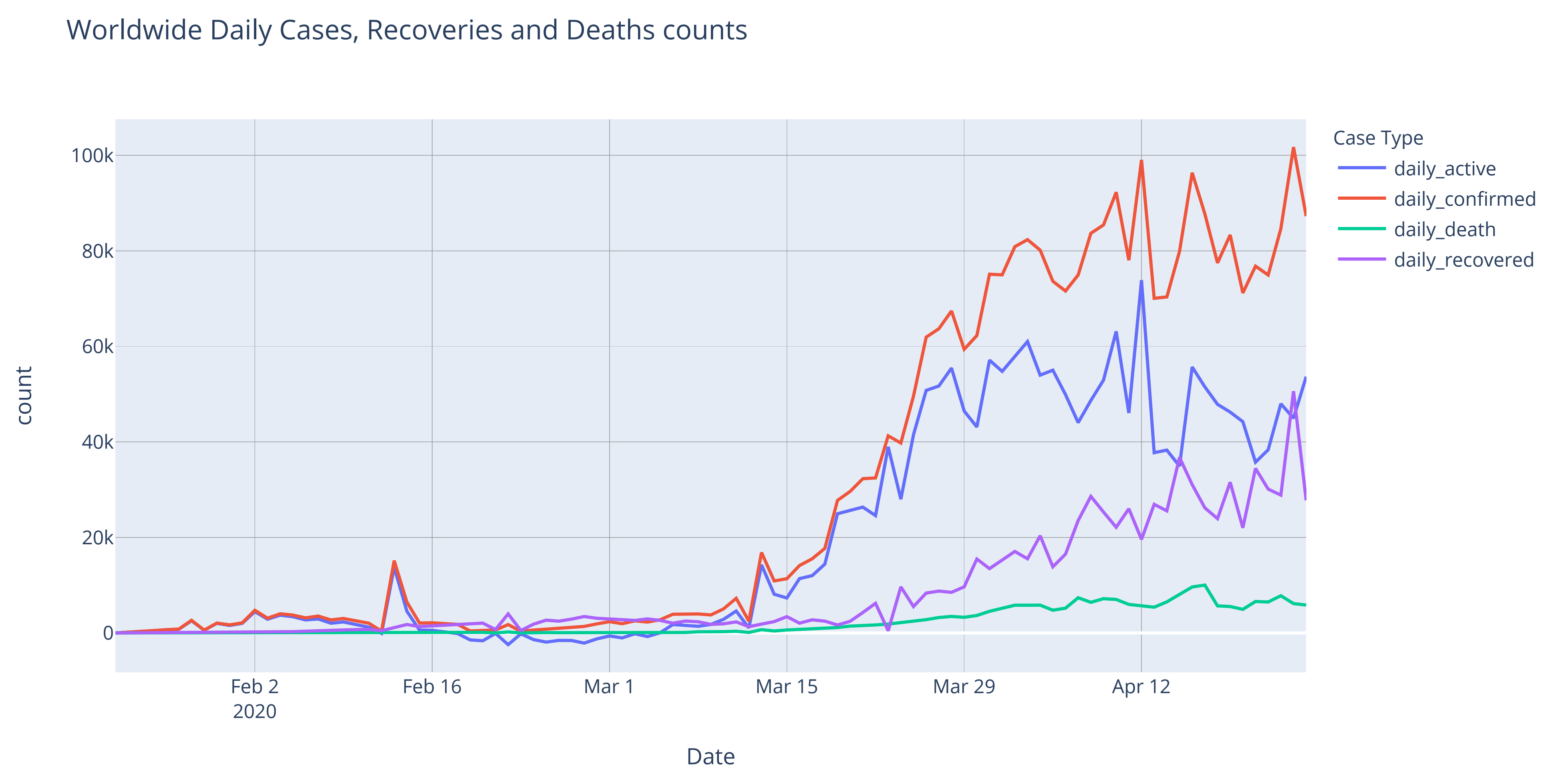

```
In [25]: gplotbar(full_df,daily=True,countryname='all',cols=['confirmed','active','recovered','death'],title='Worldwide Daily Cases, Recoveries and Deaths counts')
```



In [26]:
```
df_temp = full_df.melt(id_vars = ['country','Date','ISO','confirmed%','mortality%','pop','confirmed_cat','death_cat','recovered_cat','density pop/km2','continent','active_cat','active%'],var_name = 'Case Type',value_name='count').groupby(['Date','Case Type'],as_index=False).sum()[['Date','Case Type','count']]
px.line(df_temp,x = "Date", y = 'count',color = 'Case Type',title='Worldwide Cases, Recoveries and Deaths counts')
```



In [27]:
```
df_temp = add_daily(full_df.groupby('Date',as_index=False).sum()[['Date','confirmed','death','recovered','active']])[['Date','daily_confirmed','daily_death','daily_recovered','daily_active']]
df_temp = df_temp.melt(id_vars = ['Date'],var_name = 'Case Type',value_name='count').groupby(['Date','Case Type'],as_index=False).sum()[['Date','Case Type','count']]
px.line(df_temp,x = "Date", y = 'count',color = 'Case Type',title='Worldwide Daily Cases, Recoveries and Deaths counts')
```



Conclusions :

- World Confirmed cases seem to increase almost exponentially and started to take off from mid March 2020.
- World Recovered cases started to increase almost exponentially from the beginning of April 2020.
- World Active cases started to decline from the end of March 2020.
- World Death cases seem to be in steady increase with low curve line.

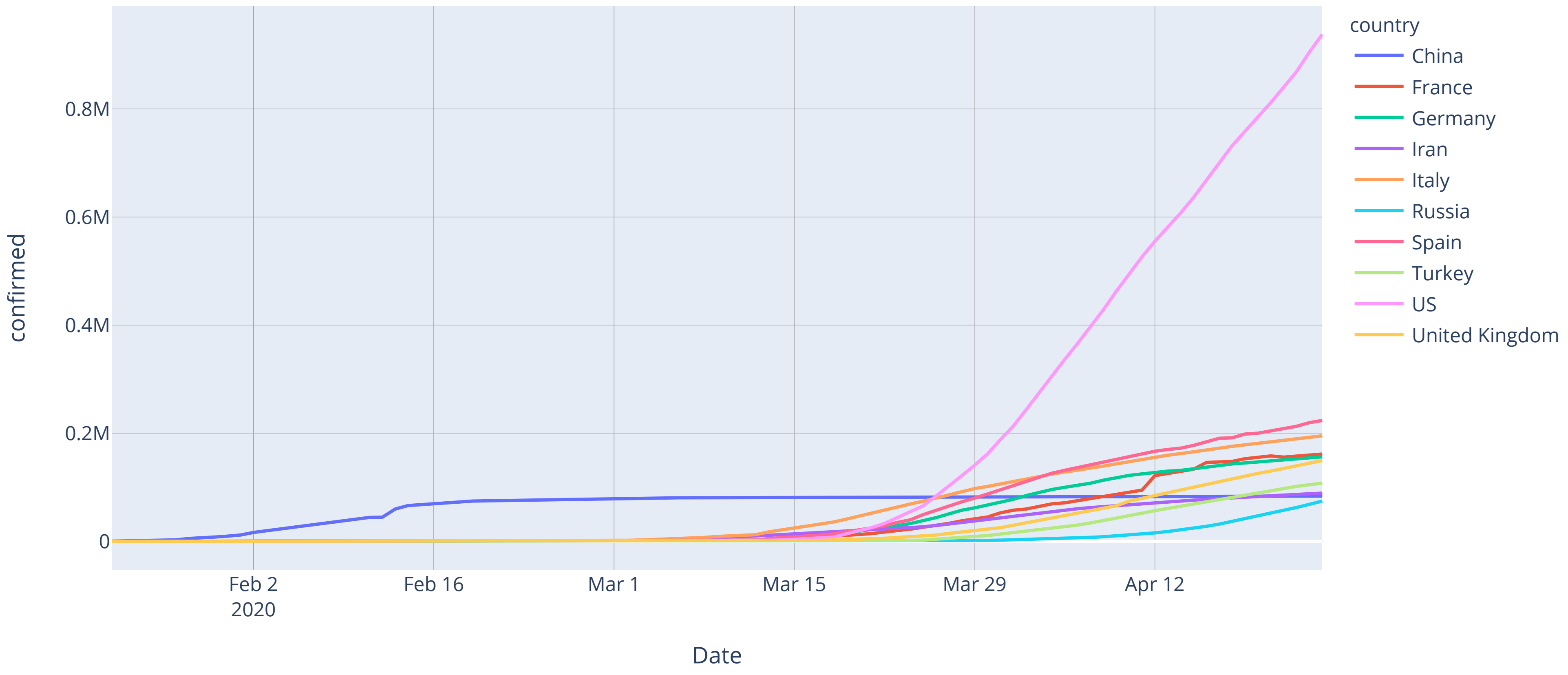
International Analysis

Figuring out top countries with confirmed, death, recovered, active cases and deduce trends and conclusions from.

> ***TIP :*** Click on US in the legend of the graph to have a more clear view of the data.

In [28]:
```
sub_df = full_df[full_df.Date == full_df.Date.max()].nlargest(10,'confirmed')
pxplotline(full_df,sub_df,'confirmed',x ='Date',title='Total # Cases for top 10 affected countries')
# sub_df.country.apply(get_country_details)
```

Total # Cases for top 10 affected countries

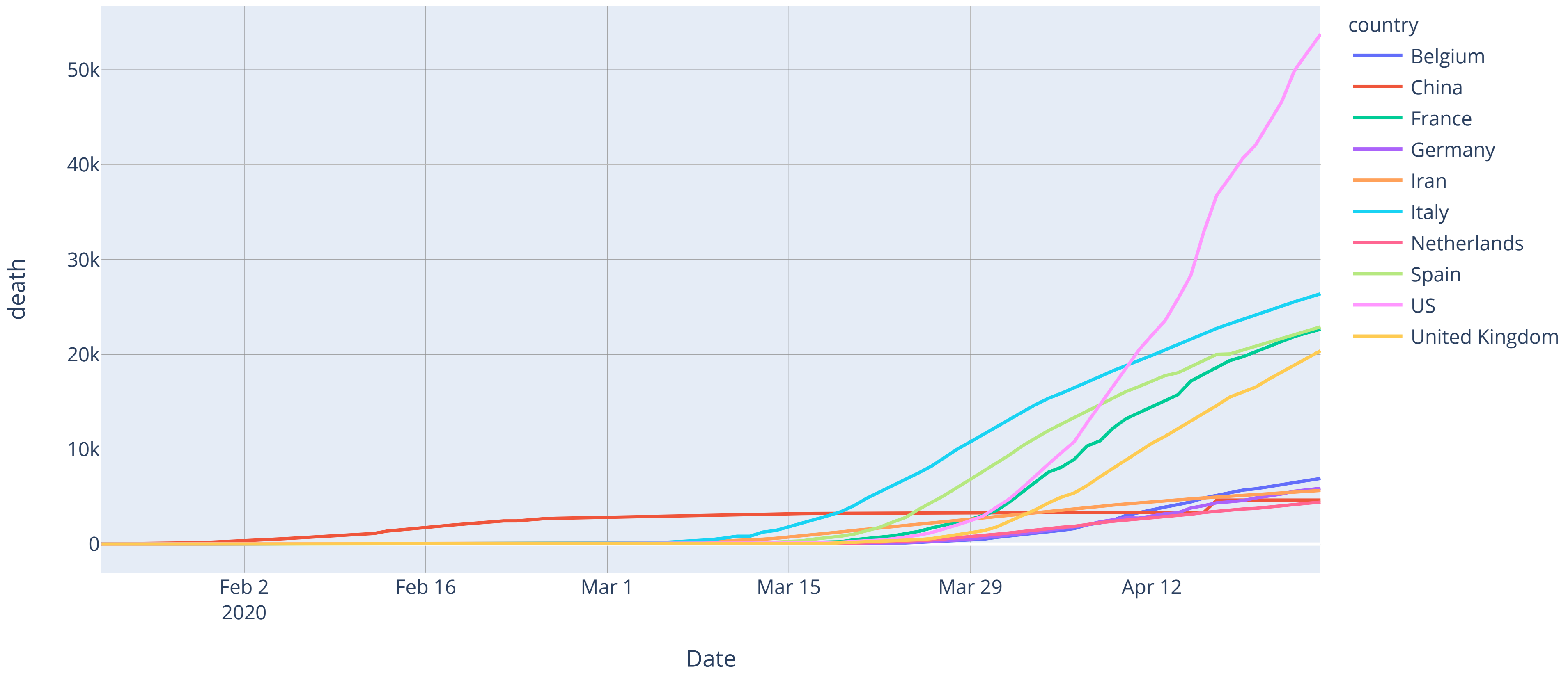


Deductions :

- Six of the top ten countries Confirmed cases are European countries.
- US is the top count country.
- Belgium is the 10th country with confirmed cases.

In [29]:
```
sub_df = full_df[full_df.Date == full_df.Date.max()].nlargest(10,'death')
pxplotline(full_df,sub_df,'death',x ='Date',title='Total # Deaths for top 10 affected countries')
#sub_df.country.apply(get_country_details)
```

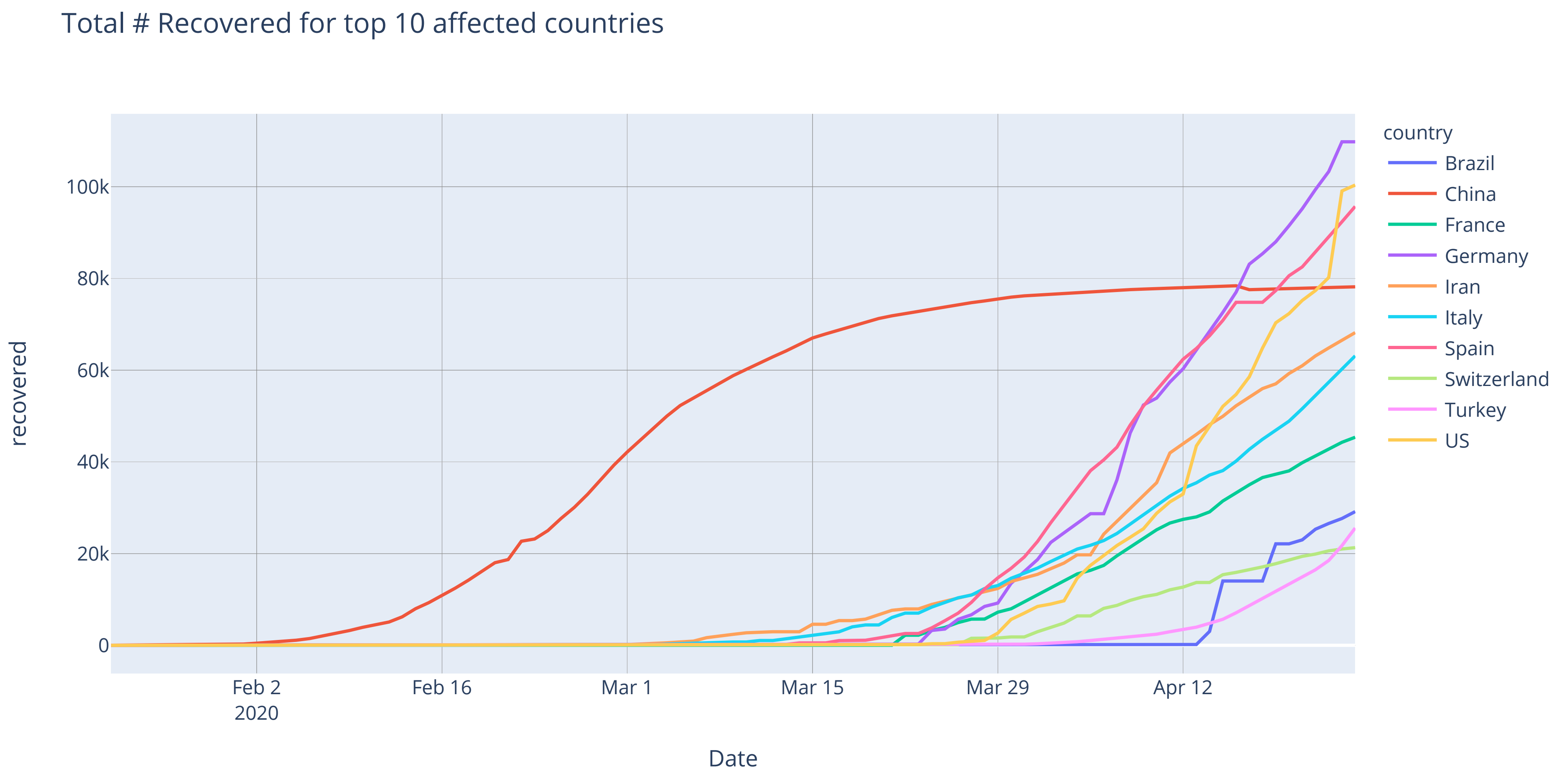
Total # Deaths for top 10 affected countries



Deductions :

- Seven of the top ten countries Death cases are European countries.
- US is the top Death cases count country.
- Netherlands is the 10th country with Death cases.

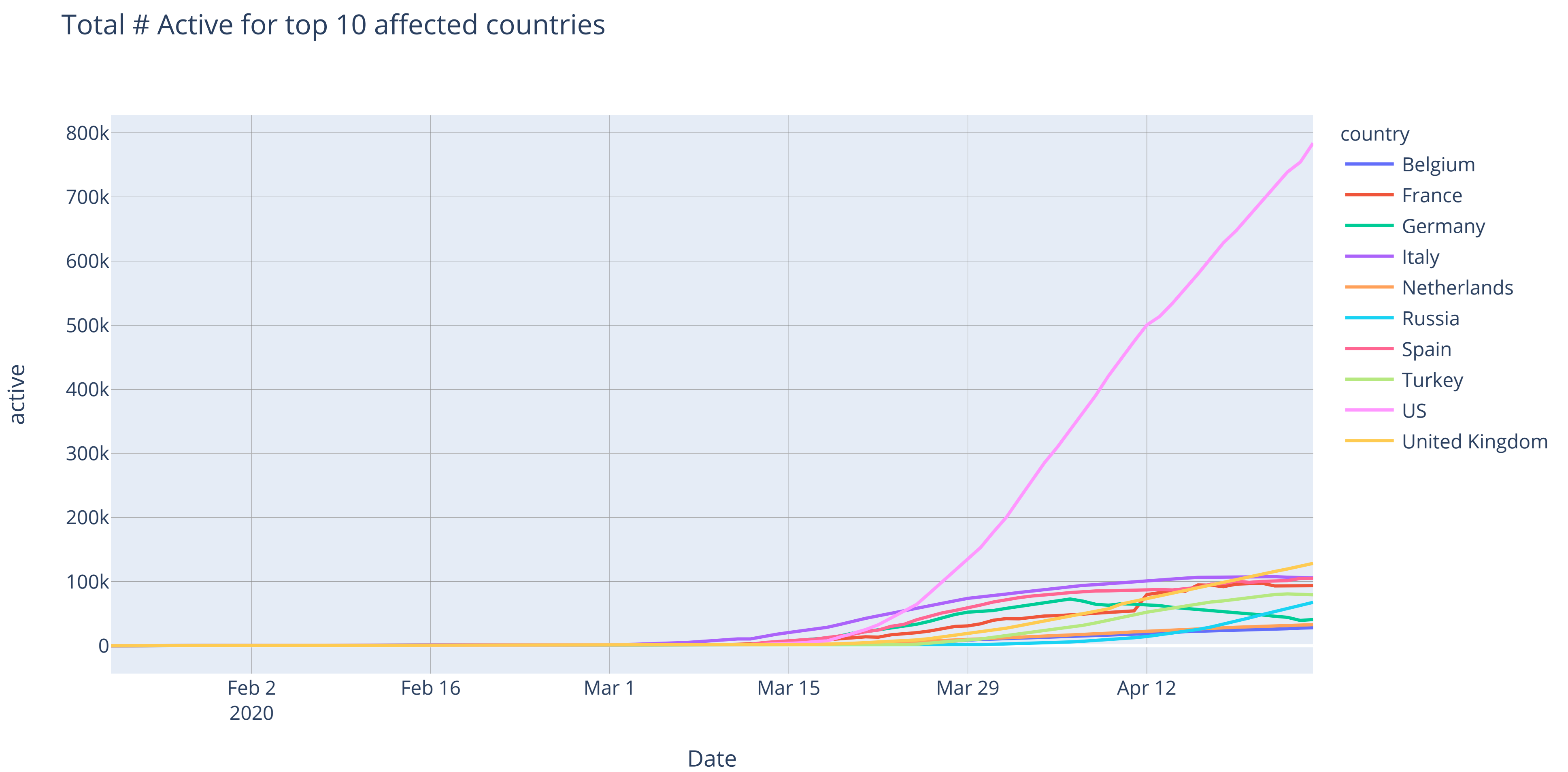
In [30]:
```
sub_df = full_df[full_df.Date == full_df.Date.max()].nlargest(10,'recovered')
pxplotline(full_df,sub_df,'recovered',x ='Date',title='Total # Recovered for top 10 affected countries')
#sub_df.country.apply(get_country_details)
```



Deductions :

- Five of the top ten countries Reccovered cases are European countries.
- Germany is the top Recovered cases count country.
- Canada is the 10th country with Recovered cases.

In [31]:
```
sub_df = full_df[full_df.Date == full_df.Date.max()].nlargest(10,'active')
pxplotline(full_df,sub_df,'active',x ='Date',title='Total # Active for top 10 affected countries')
#sub_df.country.apply(get_country_details)
```



Deductions :

- Eight of the top ten countries Active cases are European countries.
- US is the top Active cases count country.
- Belgium is the 10th country with Death cases.

Conclusions :

- Europe is the most critical continent with very high confirmed cases, high deaths and high active cases.
- US is with most confirmed, death and active cases and is considered the new virus epicenter.
- China is the most rapid recovery rate yet Germany has the most count.
- No African countries and only one Asian country are in the top ten countries with active cases .

Finding the least and most active countries

> Finding the most active and least active in the top ten most affected countries is reflective of how the virus envelops one country after the other and how some countries seemed to overcome the challange.

In [32]:
```
least_active = full_df[full_df.Date == full_df.Date.max()].sort_values('confirmed',ascending = False)[['country','confirmed','d
eath','recovered','active','active%']]
least_active = least_active.nlargest(10,'confirmed')
least_active = least_active.sort_values(by = 'active%')
```

In [33]:
```
display(least_active.style.background_gradient(cmap='Blues',subset=["confirmed"])\
                          .background_gradient(cmap='Reds',subset=["death"])\
                          .background_gradient(cmap='Greens',subset=["recovered"])\
                          .background_gradient(cmap='YlOrBr',subset=["active"])\
                          .background_gradient(cmap='Purples',subset=["active%"]))
```

	country	confirmed	death	recovered	active	active%
3419	China	83909	4636	78175	1098	1.30856
7504	Iran	89328	5650	68193	15485	17.335
6174	Germany	156513	5877	109800	40836	26.0911
14629	Spain	223759	22902	95708	105149	46.9921
7884	Italy	195351	26384	63120	105847	54.183
5794	France	161644	22648	45372	93624	57.9199
15959	Turkey	107773	2706	25582	79485	73.7522
16054	US	938154	53755	100372	784027	83.5713
16434	United Kingdom	149569	20381	774	128414	85.856
13014	Russia	74588	681	6250	67657	90.7076

Conclusions :

- US is the top country with confirmed cases
- United Kingdom is the top country with active cases percentage of total confirmed cases
- China is the least country with active cases percentage of total confirmed cases
- Germany is the top country with recovered cases

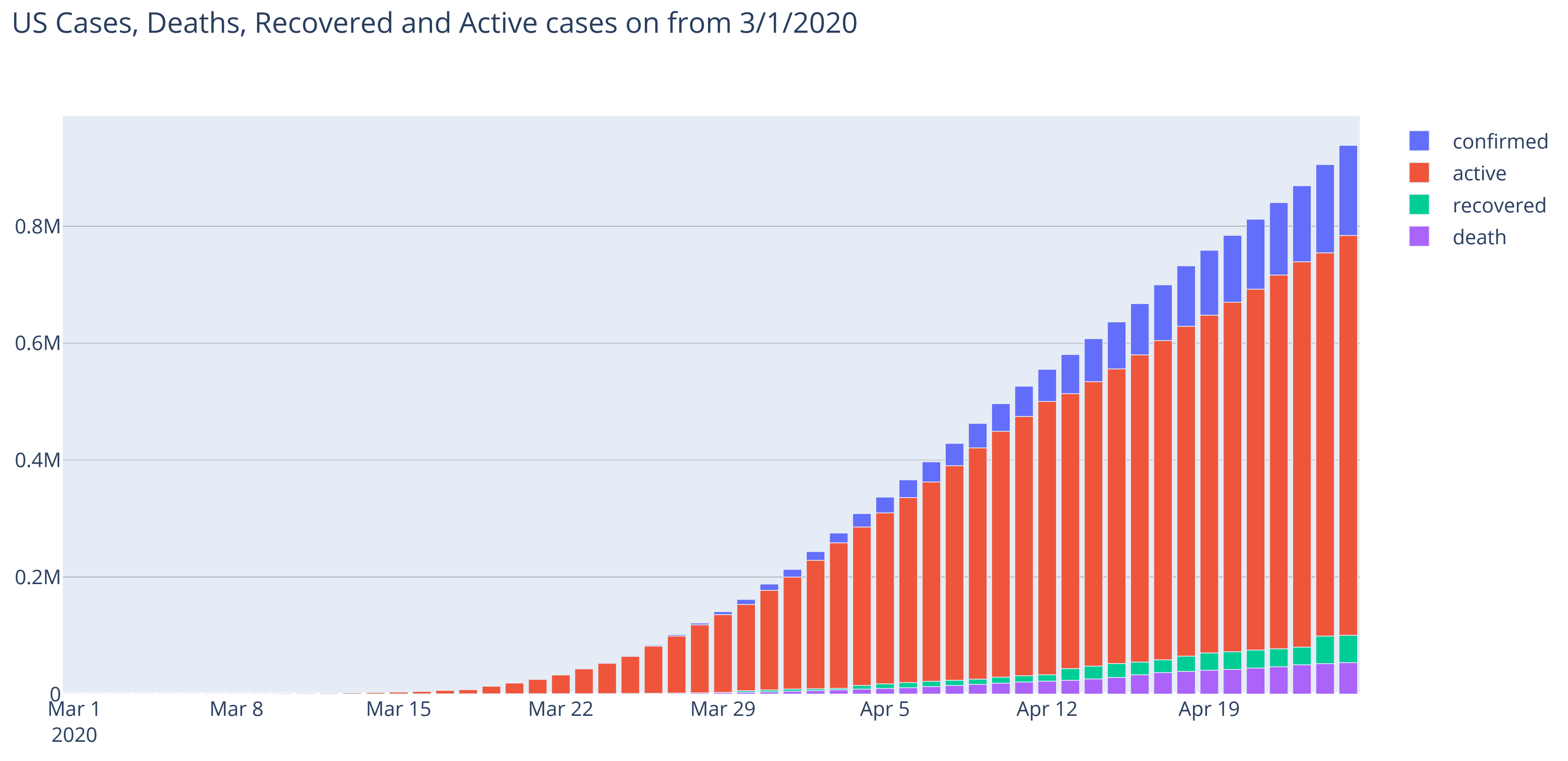
Further Analysis on most and least active countries

Starting analysis with US as it is the top country with confirmed cases, following up with United Kingdom as it has the highest percentage of active cases. Finalizing this part of the analysis with contrary countries. Starting with China, having the least percentage of active cases, following with Germany, having the most recovered cases.

United States National Trends

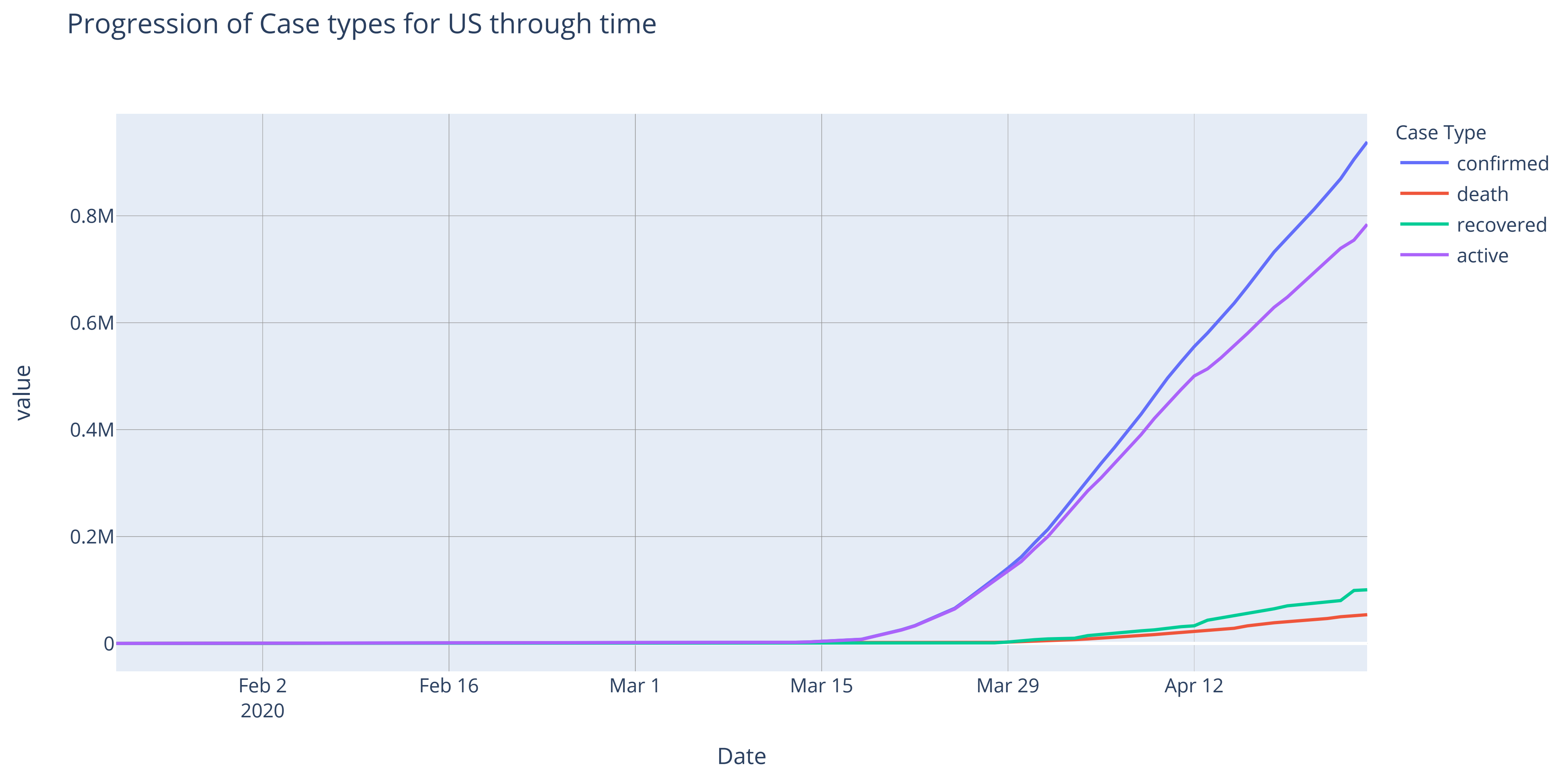
Starting analysis with progression from the 1st of March 2020.

In [34]:
```
gplotbar(full_df,'US',cols=['confirmed','active','recovered','death'],daily=False,title = 'US Cases, Deaths, Recovered and Acti
ve cases on from 3/1/2020',startdate='3/1/2020')
```

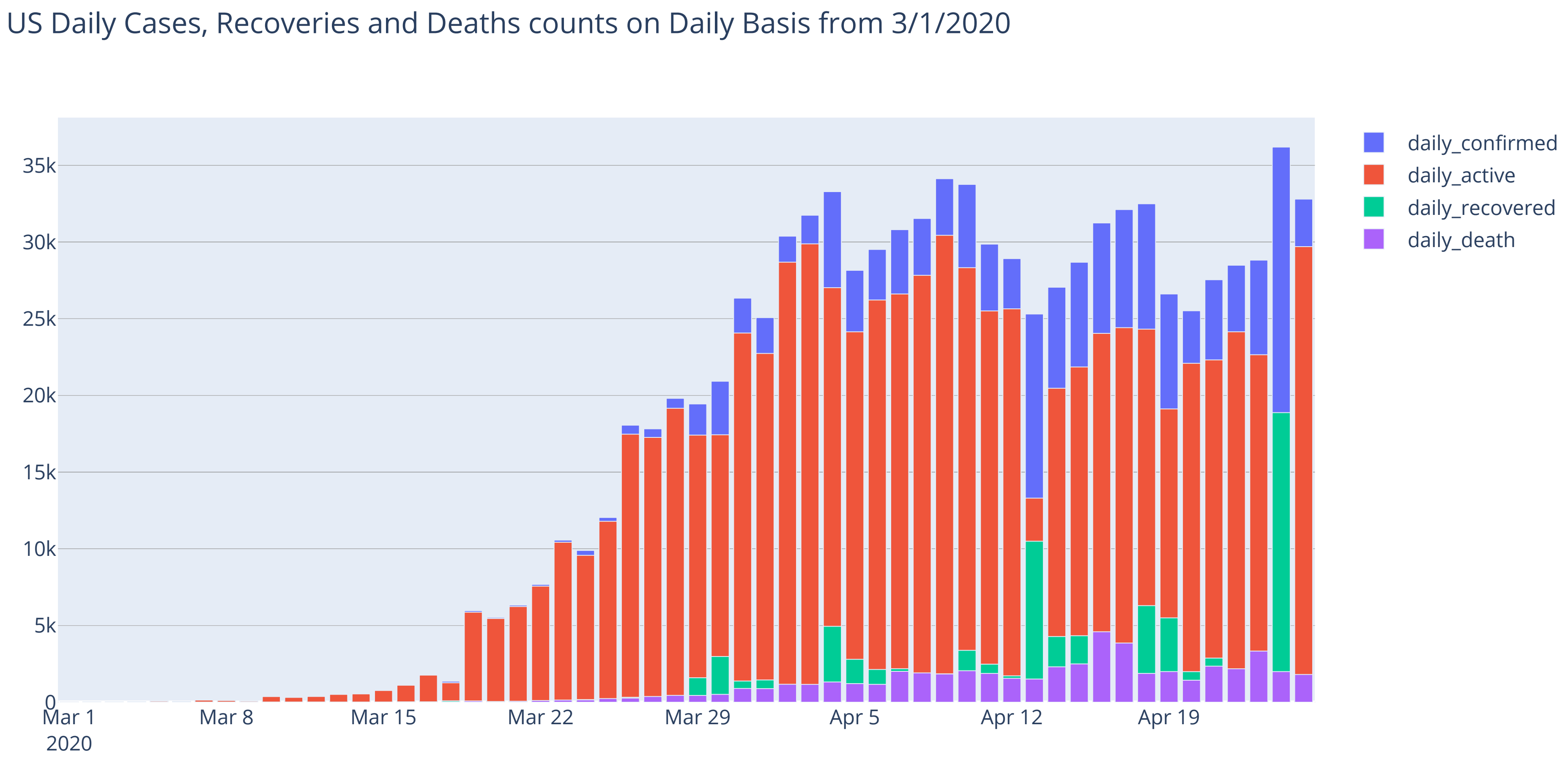


In [35]:
```
#Creating a reshaped df with Case Type as one column
country_all = full_df.melt(id_vars = ['country','Date','ISO','confirmed%','mortality%','pop','confirmed_cat','death_cat','recov
ered_cat','density pop/km2','continent','active_cat','active%'],var_name = 'Case Type')


countryname= 'US'
fig = px.line(country_all[(country_all['country'] == countryname)],x = 'Date', y = 'value',color = 'Case Type',title=f'Progress
ion of Case types for {countryname} through time')
fig.show()
```



In [36]:
```
gplotbar(full_df,daily=True,countryname='US',cols=['confirmed','active','recovered','death'],title='US Daily Cases, Recoveries and Deaths counts on Daily Basis from 3/1/2020',startdate='3/1/2020')
```



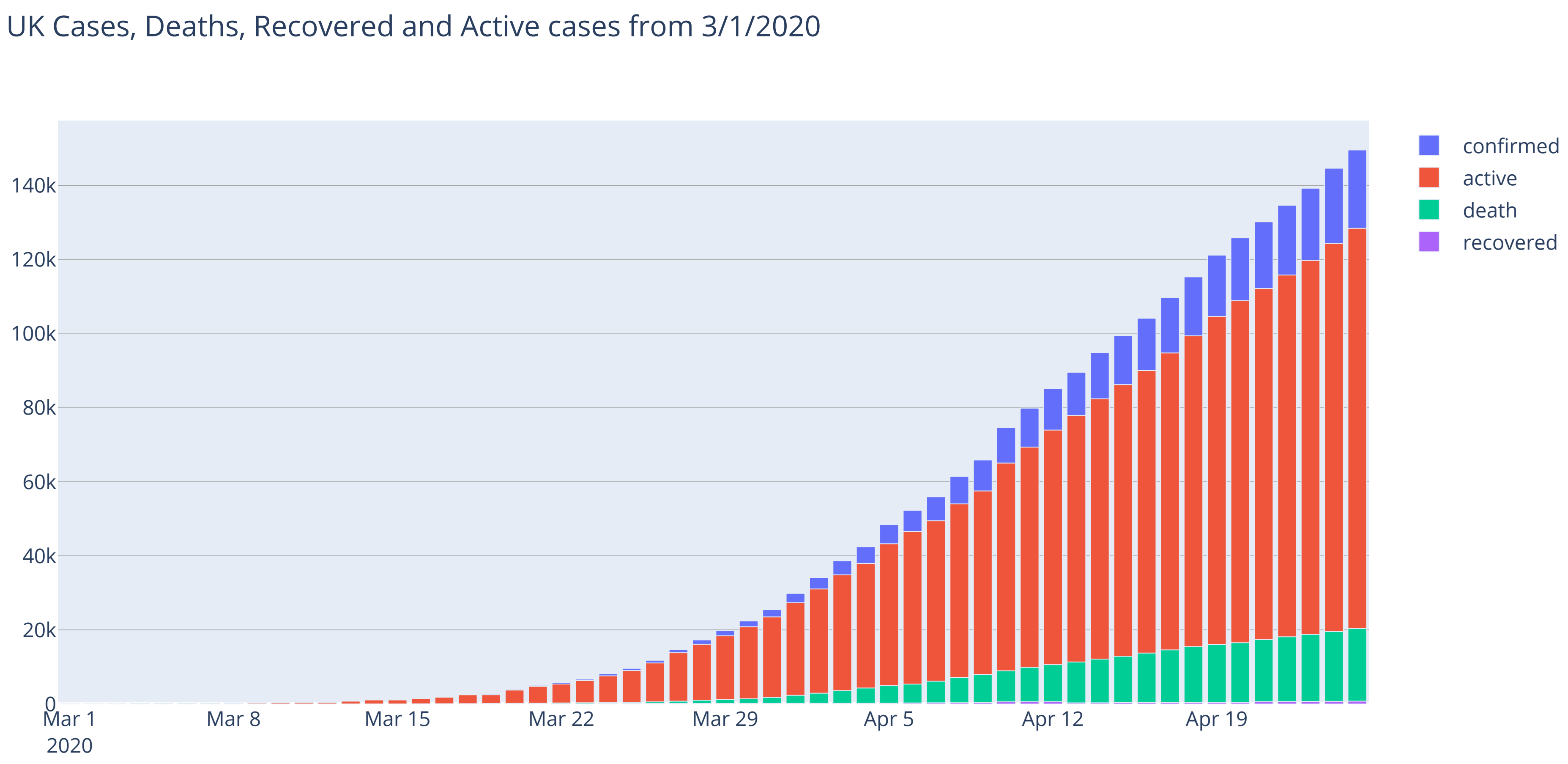
Conclusions :

- US Started Lockdown on the 22nd of March.
- Since the lockdown 'Active Cases' has ever been dropping.
- Since 29th of March Recovered Cases has started to increase yet death rate is increasing too. Which may suggest a weak healthcare system.
- Cases in the US are growing almost exponentially.
- US is considered and due to this data a virus epicenter.

United Kingdom National Trends

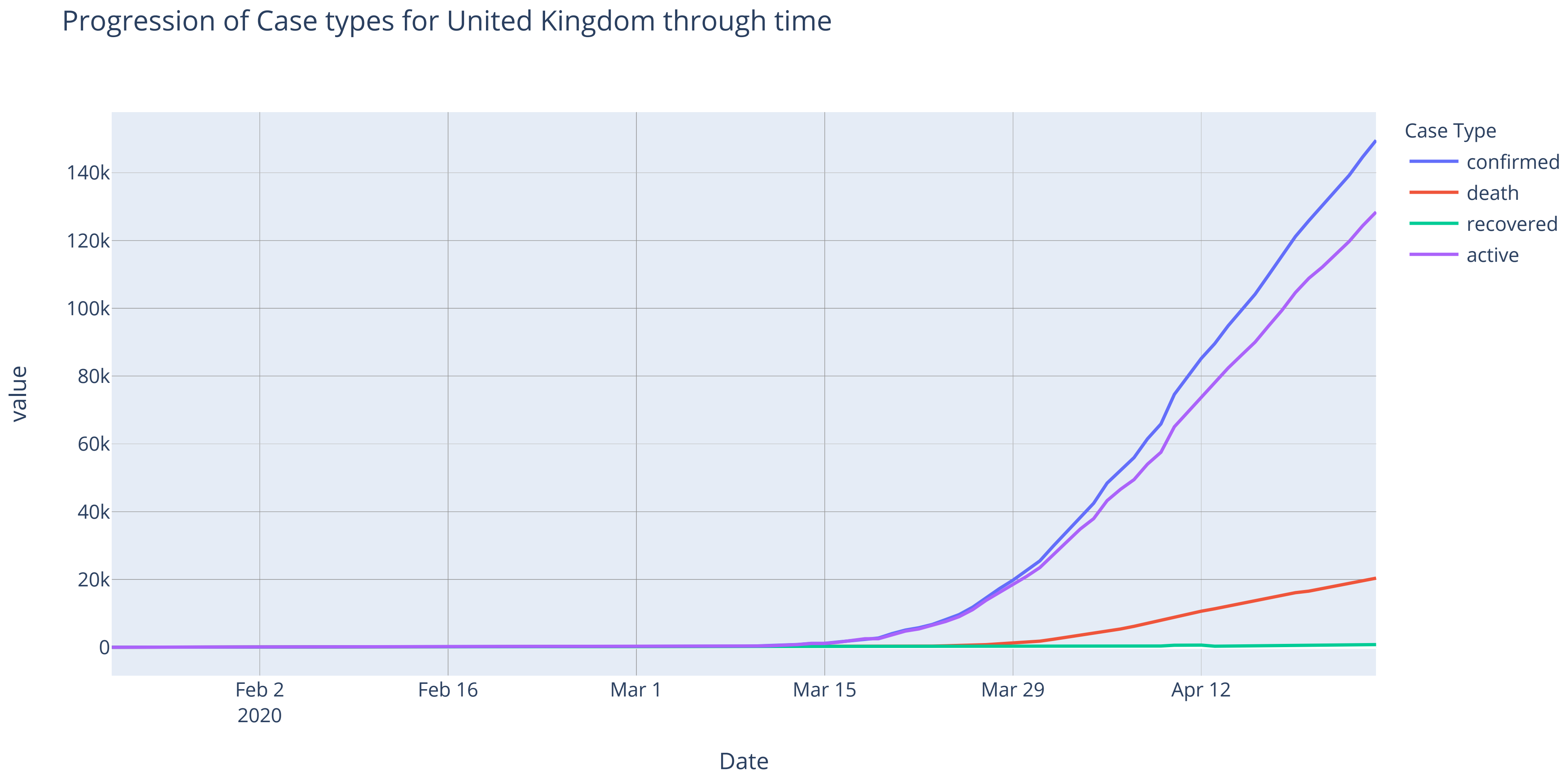
Starting analysis with progression from the 1st of March 2020.

In [37]:
```
gplotbar(full_df,'United Kingdom',cols=['confirmed','active','death','recovered'],daily=False,title = 'UK Cases, Deaths, Recovered and Active cases from 3/1/2020',startdate='3/1/2020')
```

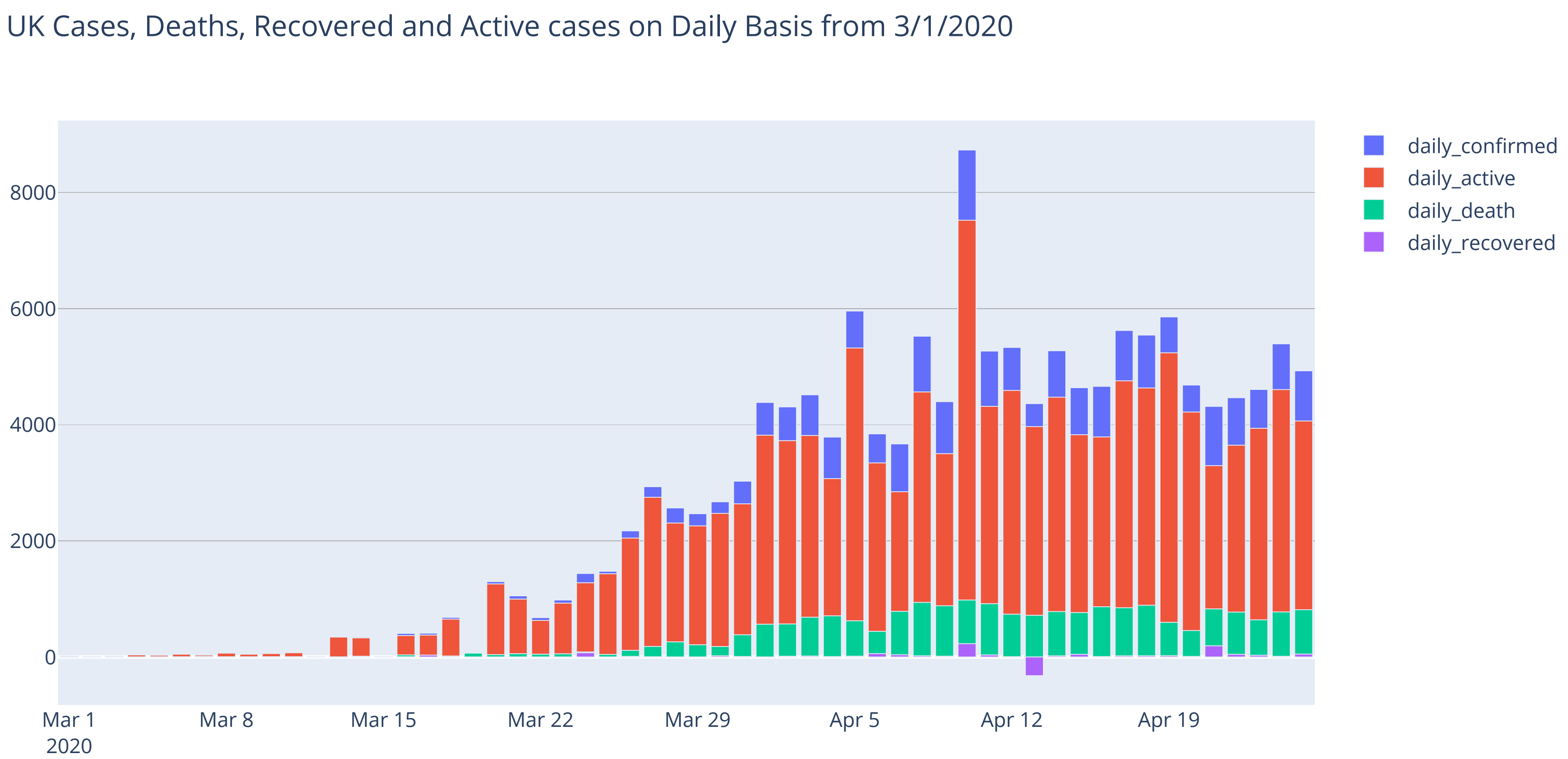


In [38]:
```python
#Creating a reshaped df with Case Type as one column
country_all = full_df.melt(id_vars = ['country','Date','ISO','confirmed%','mortality%','pop','confirmed_cat','death_cat','recovered_cat','density pop/km2','continent','active_cat','active%'],var_name = 'Case Type')


countryname= 'United Kingdom'
fig = px.line(country_all[(country_all['country'] == countryname)],x = 'Date', y = 'value',color = 'Case Type',title=f'Progression of Case types for {countryname} through time')
fig.show()
```



In [39]:
```python
gplotbar(full_df,daily=True,countryname='United Kingdom',cols=['confirmed','active','death','recovered'],title='UK Cases, Deaths, Recovered and Active cases on Daily Basis from 3/1/2020',startdate='3/1/2020')
```



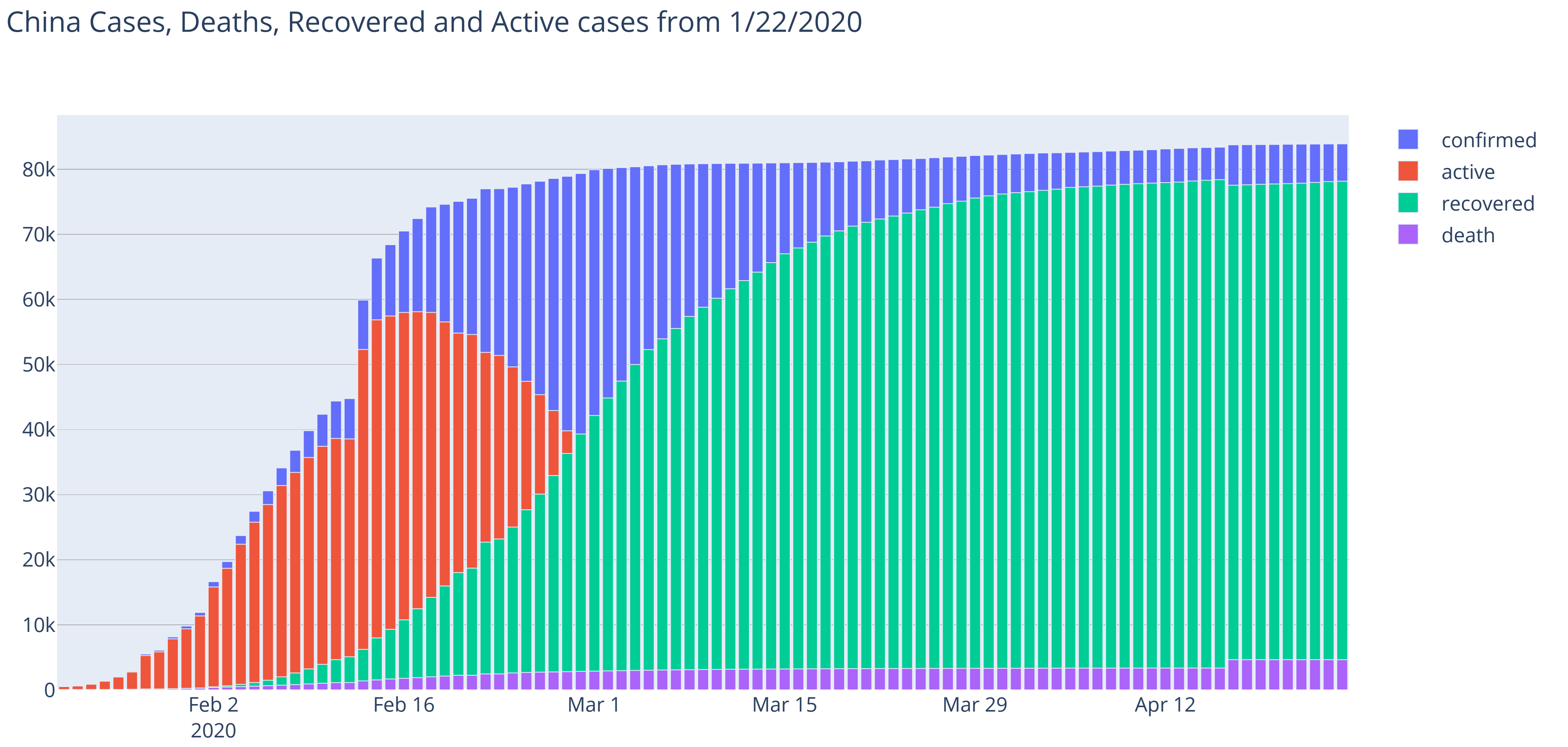
Conclusions :

- UK Started Lockdown on the 23rd of March.
- Since the lockdown 'Active Cases' has ever been dropping with low rate until the 1st of April.
- Since 26th of March Death Cases has started to increase yet Recovery rate is very low. Which may suggest a very weak healthcare system.
- Cases in the UK are growing almost exponentially.
- UK has a very high active case rate and very low recovery rate, as the data shows, UK is in a severe crisis.
- Despite the low recovery rate in the UK, the cases daily count is stabilizing since the 11th of April.

People's Republic of China National Trends

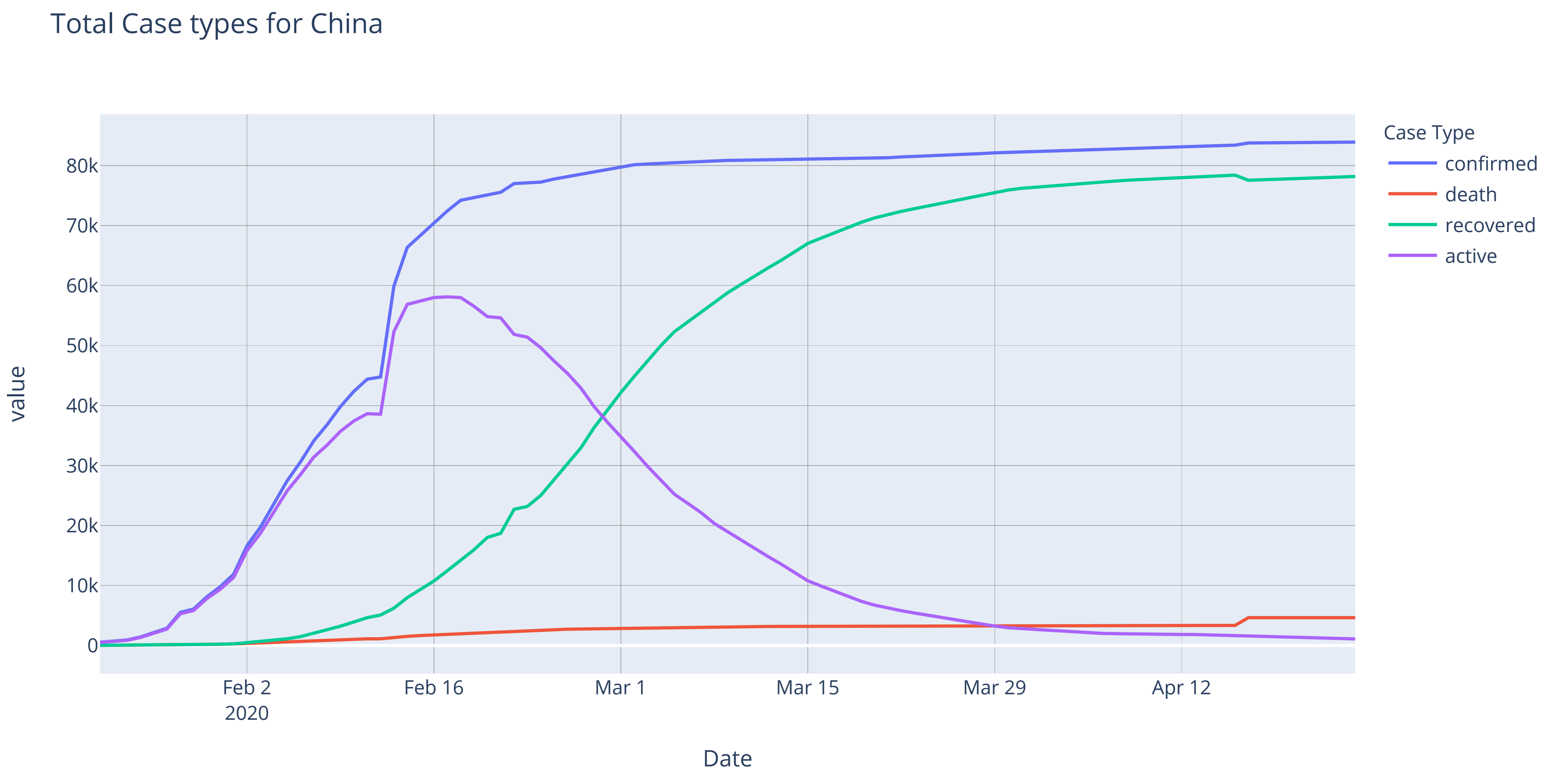
Starting analysis with progression from the 22nd of January 2020.

In [40]:
```
gplotbar(full_df,daily=False,countryname='China',cols=['confirmed','active','recovered','death'],title='China Cases, Deaths, Re
covered and Active cases from 1/22/2020')
```

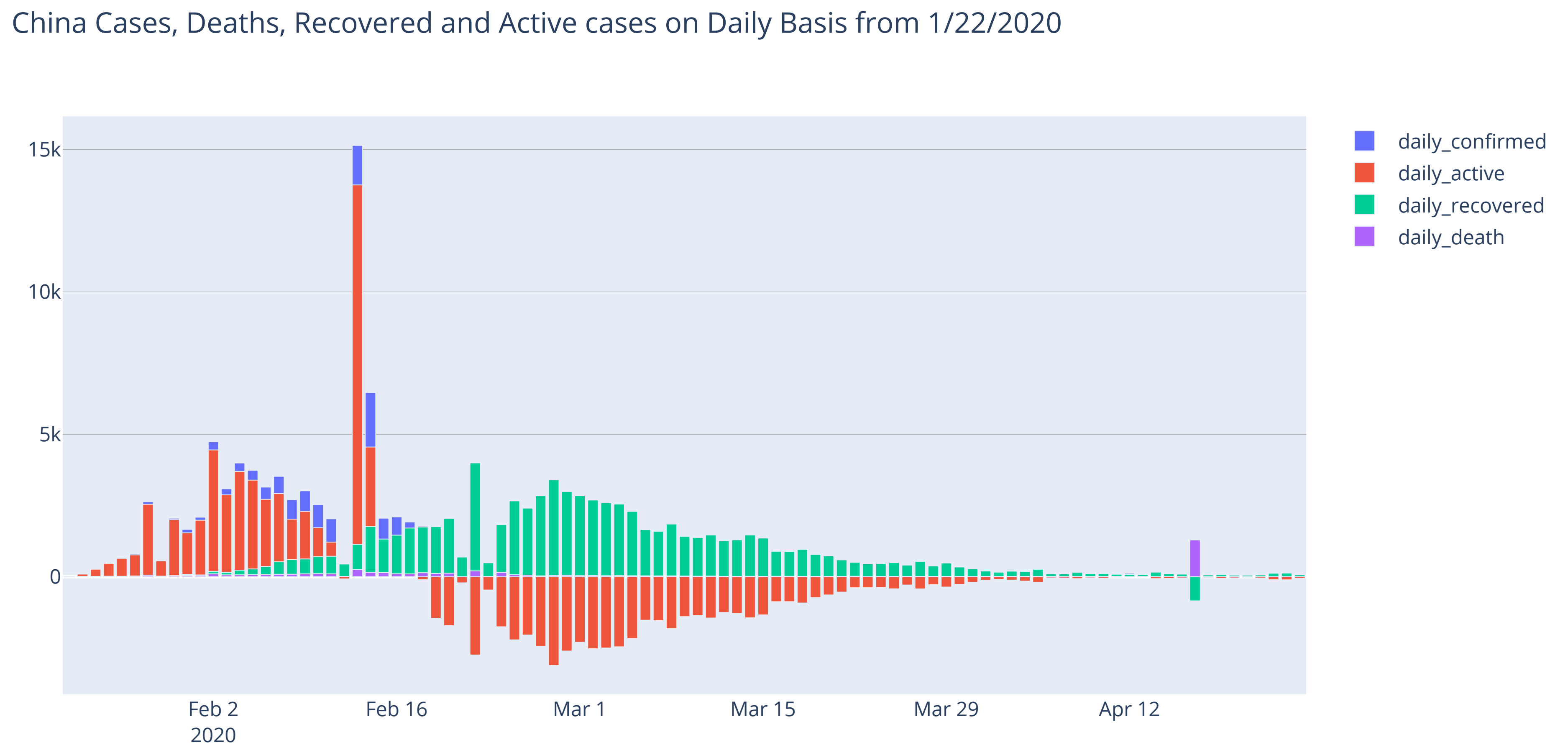


In [41]:
```
#Creating a reshaped df with Case Type as one column
country_all = full_df.melt(id_vars = ['country','Date','ISO','confirmed%','mortality%','pop','confirmed_cat','death_cat','recov
ered_cat','density pop/km2','continent','active_cat','active%'],var_name = 'Case Type')


countryname= 'China'
fig = px.line(country_all[(country_all['country'] == countryname)],x = 'Date', y = 'value',color = 'Case Type',title=f'Total Ca
se types for {countryname}')
fig.show()
```



In [42]:
```
gplotbar(full_df,daily=True,countryname='China',cols=['confirmed','active','recovered','death'],title='China Cases, Deaths, Recovered and Active cases on Daily Basis from 1/22/2020')
```



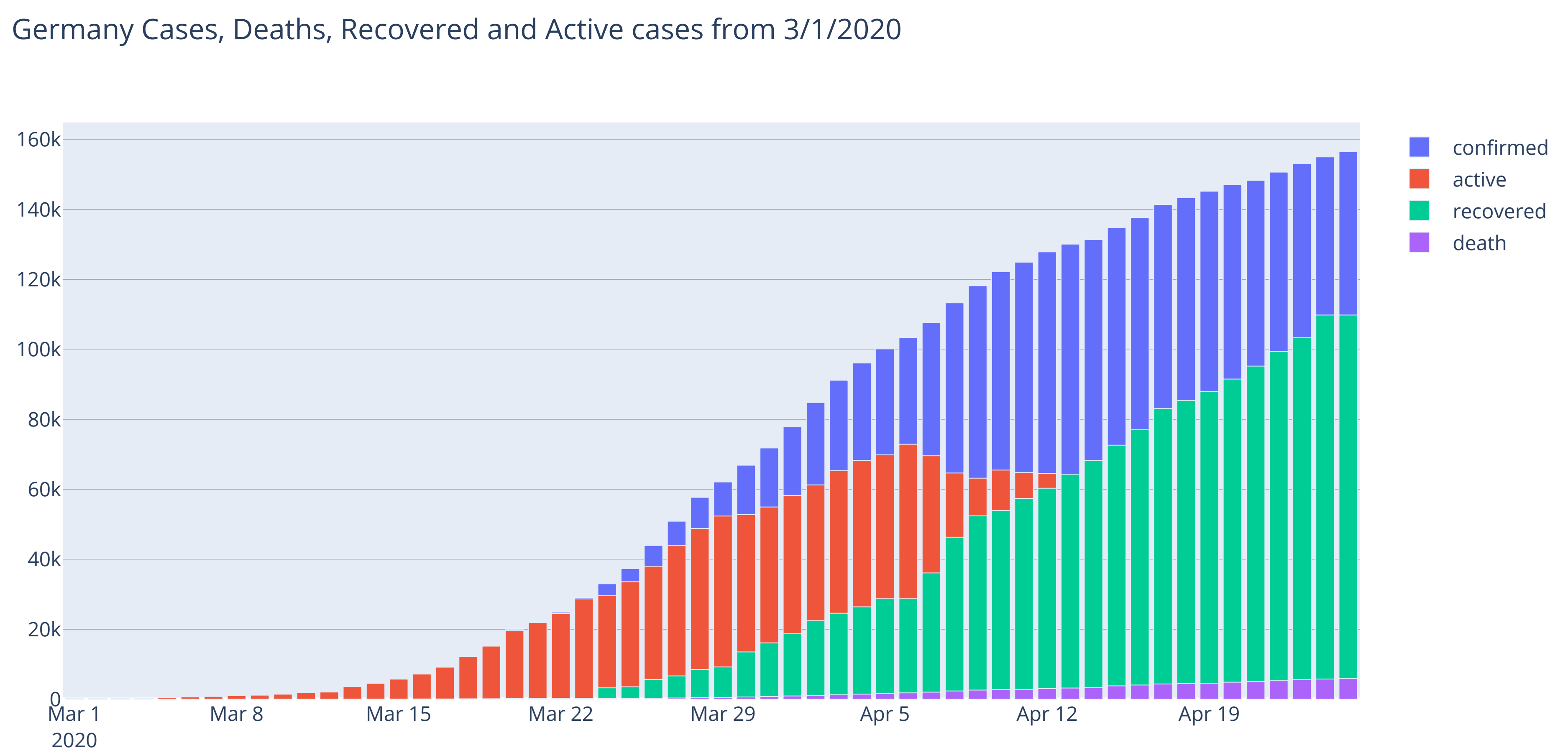
Conclusions :

- China is the virus origin and first epicenter. An outbreak that started on the 22nd of January 2020.
- China Started Lockdown on the 23rd of January. Which is the earliest country to start Lockdown.
- Active cases started dropping after around 10 days from the lockdown.
- Since the 13th of February Death Cases has started to decrease significantly.
- Since the 4th of February -11 days after the lockdown- Recovery cases has started increasing and doubling.
- China had a very effective and robust Lockdown, and a very effective healthcare system, which resulted in very low death rate, very high recovery rate and minimum active cases rate globally.

Germany National Trends

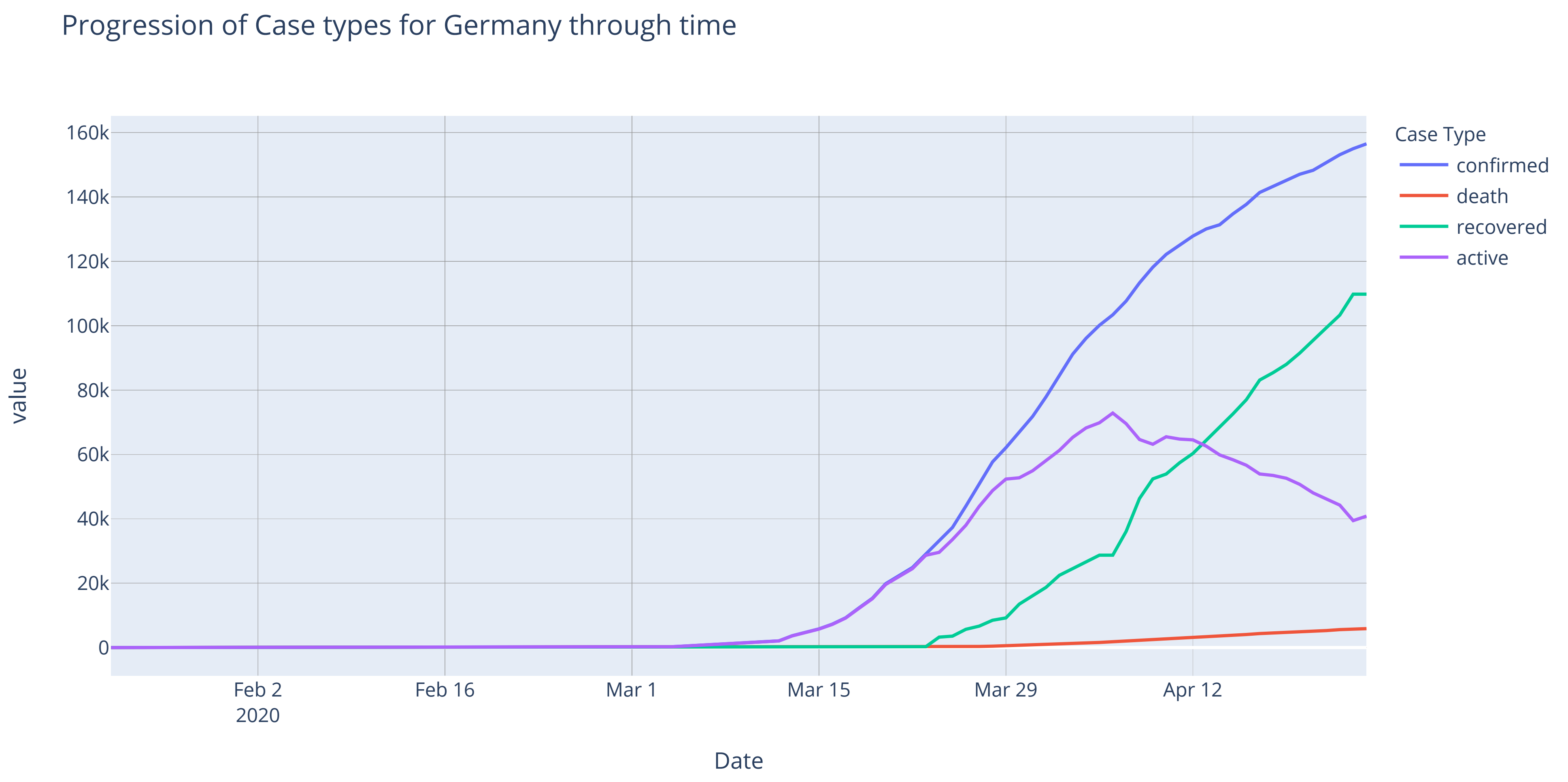
Starting analysis with progression from the 1st of March 2020.

In [43]:
```
gplotbar(full_df,daily=False,countryname='Germany',cols=['confirmed','active','recovered','death'],title='Germany Cases, Deaths, Recovered and Active cases from 3/1/2020',startdate='3/1/2020')
```

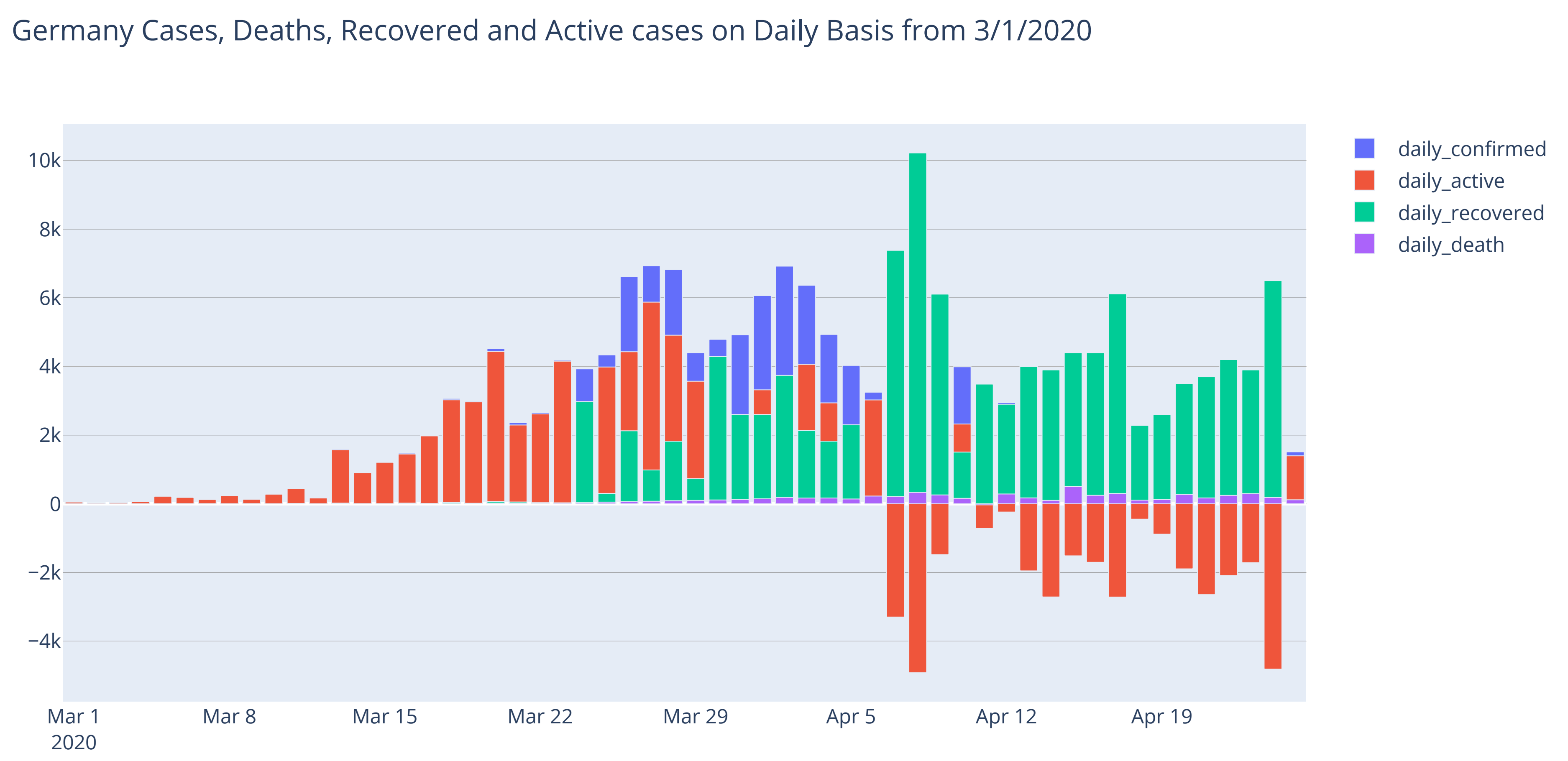


In [44]:
```
#Creating a reshaped df with Case Type as one column
country_all = full_df.melt(id_vars = ['country','Date','ISO','confirmed%','mortality%','pop','confirmed_cat','death_cat','recov
ered_cat','density pop/km2','continent','active_cat','active%'],var_name = 'Case Type')


countryname= 'Germany'
fig = px.line(country_all[(country_all['country'] == countryname)],x = 'Date', y = 'value',color = 'Case Type',title=f'Progress
ion of Case types for {countryname} through time')
fig.show()
```



In [45]:
```
gplotbar(full_df,daily=True,countryname='Germany',cols=['confirmed','active','recovered','death'],title='Germany Cases, Deaths,
Recovered and Active cases on Daily Basis from 3/1/2020',startdate='3/1/2020')
```



Conclusions :

- Germany Started Lockdown on the 22rd of March.
- Active cases started dropping rapidly ever since.
- Since the 13th of April Recovery count surpassed the active cases count.
- Death Cases in Germany never reached critical points.
- Germany had a very effective and robust Lockdown, and a very effective healthcare system, which resulted in very low death rate, very high recovery rate and active cases count are in steady decrease.

Exploring Mortality Rates

The mortality rate is a critical indicator in such crises as it translates the percentage of deaths to confirmed cases and describes how aggressive the pandemic is to individual countries. In this section, Mortality rate is explored to find trends and patterns as well as analyse which countries are in more critical condition.

Finding most affected countries and most mortality rates

Sorting countries by mortality rates calculated from confirmed cases and deaths.

In [46]:
```
df_temp = full_df[(full_df.Date == full_df.Date.max()) & (full_df.confirmed > 1000)][['country','confirmed','death','recovered'
,'density pop/km2','mortality%','confirmed%']].sort_values(by = 'mortality%',ascending = False)
df_temp.style.background_gradient(cmap='Blues',subset=["confirmed"])\
                        .background_gradient(cmap='Reds',subset=["death"])\
                        .background_gradient(cmap='Greens',subset=["recovered"])\
                        .background_gradient(cmap='Purples',subset=["density pop/km2"])\
                        .background_gradient(cmap='YlOrBr',subset=["mortality%"])\
                        .background_gradient(cmap='bone_r',subset=["confirmed%"])
```

Out[46]:

	country	confirmed	death	recovered	density pop/km2	mortality%	confirmed%
1614	Belgium	45325	6917	10417	376	15.2609	0.3933
5794	France	161644	22648	45372	123	14.011	0.241
16434	United Kingdom	149569	20381	774	274	13.6265	0.2251
7884	Italy	195351	26384	63120	200	13.5059	0.3242
284	Algeria	3256	419	1479	18	12.8686	0.0076
15009	Sweden	18177	2192	1005	23	12.0592	0.1754
11304	Netherlands	37384	4424	102	420	11.8339	0.2143
7124	Hungary	2443	262	458	105	10.7245	0.025
14629	Spain	223759	22902	95708	93	10.2351	0.4767
10449	Mexico	13842	1305	7149	64	9.42783	0.0109
7409	Indonesia	8607	720	1042	141	8.36528	0.0032
4939	Egypt	4319	307	1114	100	7.10813	0.0043
2279	Brazil	59324	4057	29160	25	6.83872	0.0281
12539	Philippines	7294	494	792	361	6.77269	0.0067
7504	Iran	89328	5650	68193	51	6.325	0.1072
14249	Slovenia	1388	81	219	103	5.83573	0.0666
16054	US	938154	53755	100372	34	5.72987	0.2847
7694	Ireland	18561	1063	9233	70	5.72706	0.3771
12919	Romania	10635	601	2890	81	5.65115	0.0548
3039	Canada	45491	2547	16013	4	5.59891	0.1198
15104	Switzerland	28894	1599	21300	208	5.53402	0.3365
3419	China	83909	4636	78175	145	5.52503	0.006
6364	Greece	2506	130	577	81	5.18755	0.0234
664	Argentina	3780	185	1030	16	4.89418	0.0084
7599	Iraq	1763	86	1224	90	4.87805	0.0045
4464	Denmark	8643	418	5858	135	4.83628	0.1486
12634	Poland	11273	524	2126	123	4.64827	0.0294
4749	Dominican Republic	5926	273	822	216	4.60682	0.0572
3514	Colombia	5142	233	1067	40	4.53131	0.0111
2469	Bulgaria	1247	55	197	63	4.41059	0.0178
11779	North Macedonia	1367	59	374	81	4.31602	0.0658
5699	Finland	4475	186	2500	16	4.15642	0.081
10924	Morocco	3897	159	537	80	4.08006	0.0109
2089	Bosnia and Herzegovina	1486	57	592	69	3.8358	0.0423
4179	Cuba	1337	51	437	102	3.81451	0.0119
12729	Portugal	23392	880	1277	112	3.76197	0.2276
6174	Germany	156513	5877	109800	233	3.75496	0.1882
949	Austria	15148	536	12103	106	3.53842	0.1702
2944	Cameroon	1518	53	697	52	3.49144	0.0062
94	Afghanistan	1463	47	188	49	3.21258	0.0046
7314	India	26283	825	5939	414	3.13891	0.0019
4369	Czechia	7352	218	2453	135	2.96518	0.0688
11684	Nigeria	1182	35	222	218	2.96108	0.0006
9499	Lithuania	1426	41	460	43	2.87518	0.051
12159	Panama	5538	159	338	56	2.87107	0.1332
10544	Moldova	3304	94	825	79	2.84504	0.1232
5319	Estonia	1635	46	228	29	2.81346	0.1234
1329	Bangladesh	4998	140	113	1169	2.80112	0.003
12444	Peru	25331	700	7797	25	2.76341	0.0788
8074	Japan	13231	360	1656	333	2.72088	0.0105
11874	Norway	7499	201	32	17	2.68036	0.1397
4084	Croatia	2016	54	1034	72	2.67857	0.0493
4844	Ecuador	22719	576	1366	63	2.53532	0.1302
15959	Turkey	107773	2706	25582	106	2.51083	0.1296
16244	Ukraine	8125	201	782	69	2.47385	0.0194
9594	Luxembourg	3711	85	3088	237	2.29049	0.6045
8454	Korea, South	10728	242	8717	517	2.25578	0.0207

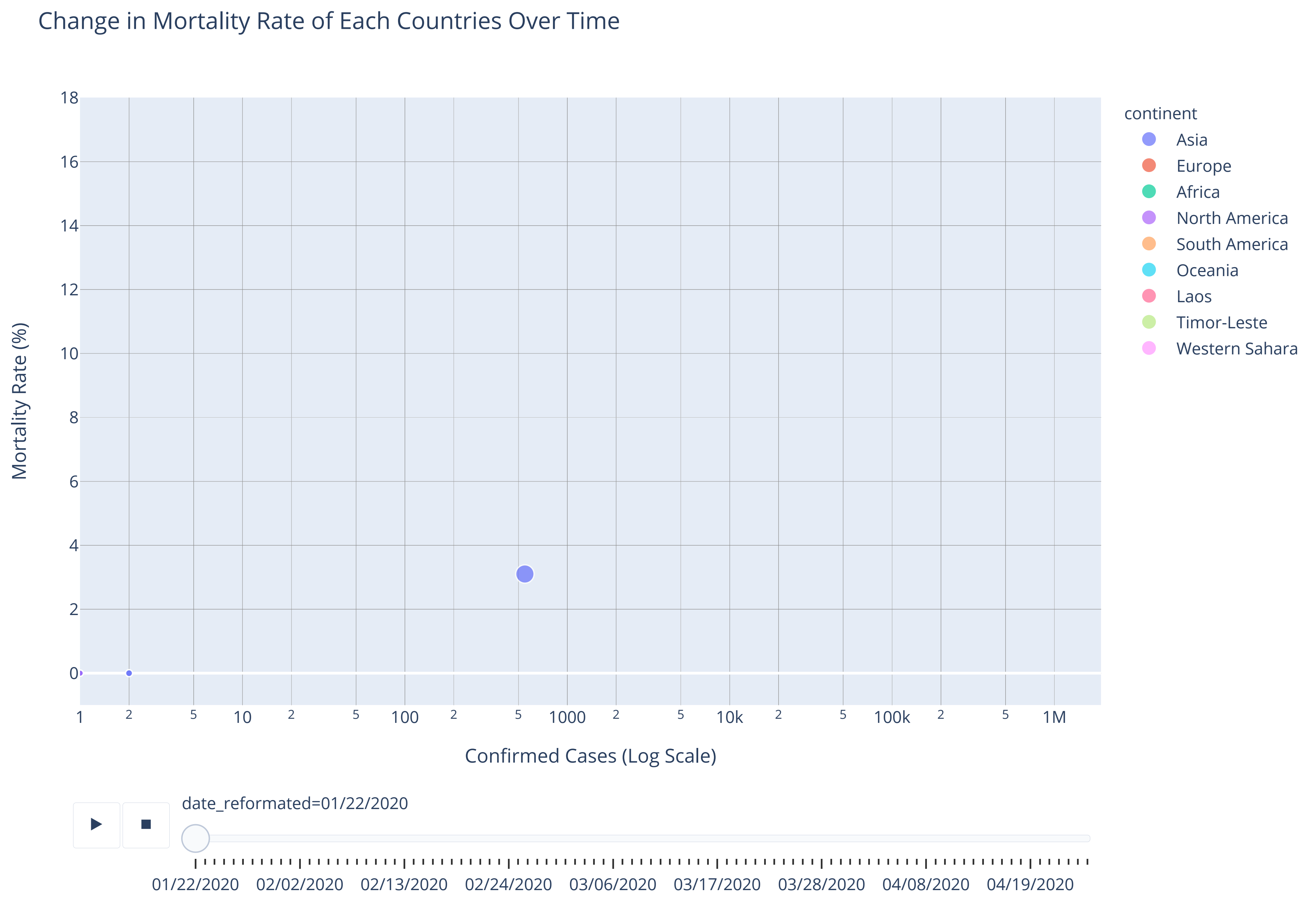
	country	confirmed	death	recovered	density pop/km2	mortality%	confirmed%
12064	Pakistan	12723	269	2866	272	2.11428	0.0058
14439	South Africa	4361	86	1473	48	1.97202	0.0074
8644	Serbia	6630	125	870	165	1.88537	0.3692
15484	Thailand	2907	51	2547	130	1.75439	0.0044
9879	Malaysia	5742	98	3762	99	1.70672	0.0175
759	Armenia	1677	28	803	99	1.66965	0.0567
3324	Chile	12858	181	6746	23	1.40768	0.074
7789	Israel	15298	199	6435	416	1.30082	0.1668
1044	Azerbaijan	1617	21	1080	116	1.2987	0.0161
14154	Slovakia	1373	17	386	111	1.23816	0.0252
11399	New Zealand	1470	18	1142	18	1.22449	0.0296
854	Australia	6694	80	5376	3	1.1951	0.0261
8264	Kazakhstan	2601	25	646	7	0.961169	0.014
13014	Russia	74588	681	6250	9	0.913015	0.0508
13679	Saudi Arabia	16299	136	2215	16	0.834407	0.0476
6269	Ghana	1279	10	134	127	0.781861	0.0042
16339	United Arab Emirates	9813	71	1887	117	0.72353	0.1004
1519	Belarus	9590	67	1573	46	0.698644	0.1013
8739	Kuwait	2892	19	656	248	0.656985	0.0654
7219	Iceland	1790	10	1570	3.5	0.558659	0.4967
11969	Oman	1905	10	329	14	0.524934	0.0455
16624	Uzbekistan	1862	8	707	73	0.429646	0.0057
1234	Bahrain	2588	8	1160	1983	0.309119	0.1677
4559	Djibouti	1008	2	373	47	0.198413	0.0935
12824	Qatar	9358	10	929	237	0.10686	0.3415
14059	Singapore	12693	12	1002	7894	0.0945403	0.2225

Visualizing rate of change of mortality rate through time

Exploring the rate of change of the mortality percentage through time as well as continents of each country and its case count.

```
In [47]: df_data = full_df.groupby(['Date', 'country'])['confirmed', 'death','continent','mortality%'].max().reset_index()
df_data["date_reformated"] = pd.to_datetime( df_data["Date"]).dt.strftime('%m/%d/%Y')


fig = px.scatter(df_data, y='mortality%',
                 x= df_data["confirmed"],
                 range_y = [-1,18],
                 range_x = [1,df_data["confirmed"].max()+1000000],
                 color= "continent",
                 hover_name="country",
                 hover_data=["confirmed","death"],
                 range_color= [0, max(np.power(df_data["confirmed"],0.3))],
                 animation_frame="date_reformated",
                 animation_group="country",
                 color_continuous_scale=px.colors.sequential.Plasma,
                 title='Change in Mortality Rate of Each Countries Over Time',
                 size = np.power(df_data["confirmed"]+1,0.3)-0.5,
                 size_max = 30,
                 log_x=True,
                 height =700,
                 )
fig.update_coloraxes(colorscale="hot")
fig.update(layout_coloraxis_showscale=False)
fig.update_xaxes(title_text="Confirmed Cases (Log Scale)")
fig.update_yaxes(title_text="Mortality Rate (%)")
fig.show()
```



Conclusions :

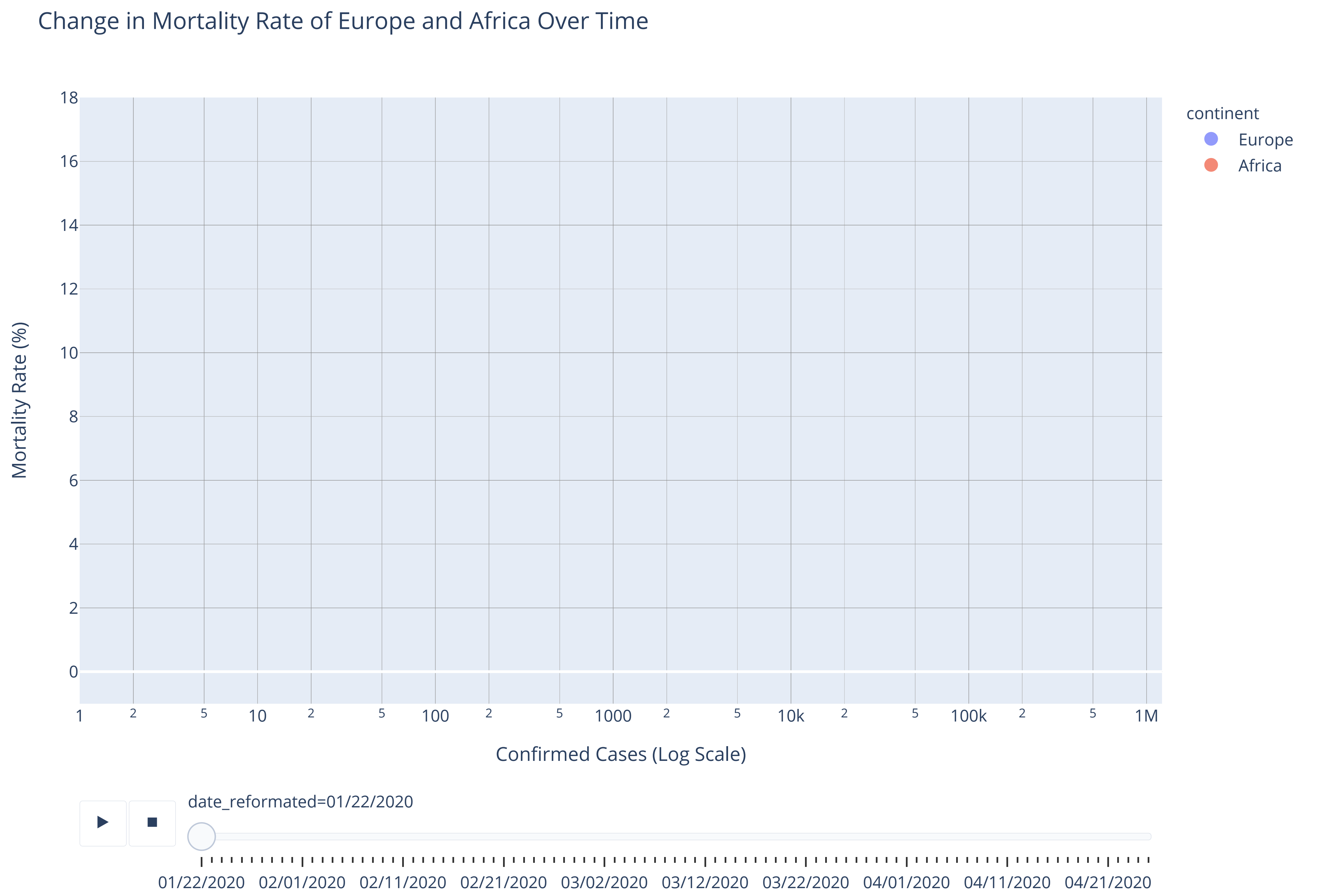
- Mortality Rate indicates how health systems perform during crisis.
- There are many Europain countries with high mortality rate and high case counts which is considered critical point.
- Up till the 1st of March no country passed a mortality rate above 7%.
- Up till the 1st of March no country passed a case count more than 4000 cases (excluding China).
- Mortality rates reached over 15% in some countries in less than a month.
- Virus is progressing fast and is in critical phase.

Exploring Mortality rate between Continents

Exploring mortality rates between continents and their ordinary least squares to find insights on how they interact with each other through time.

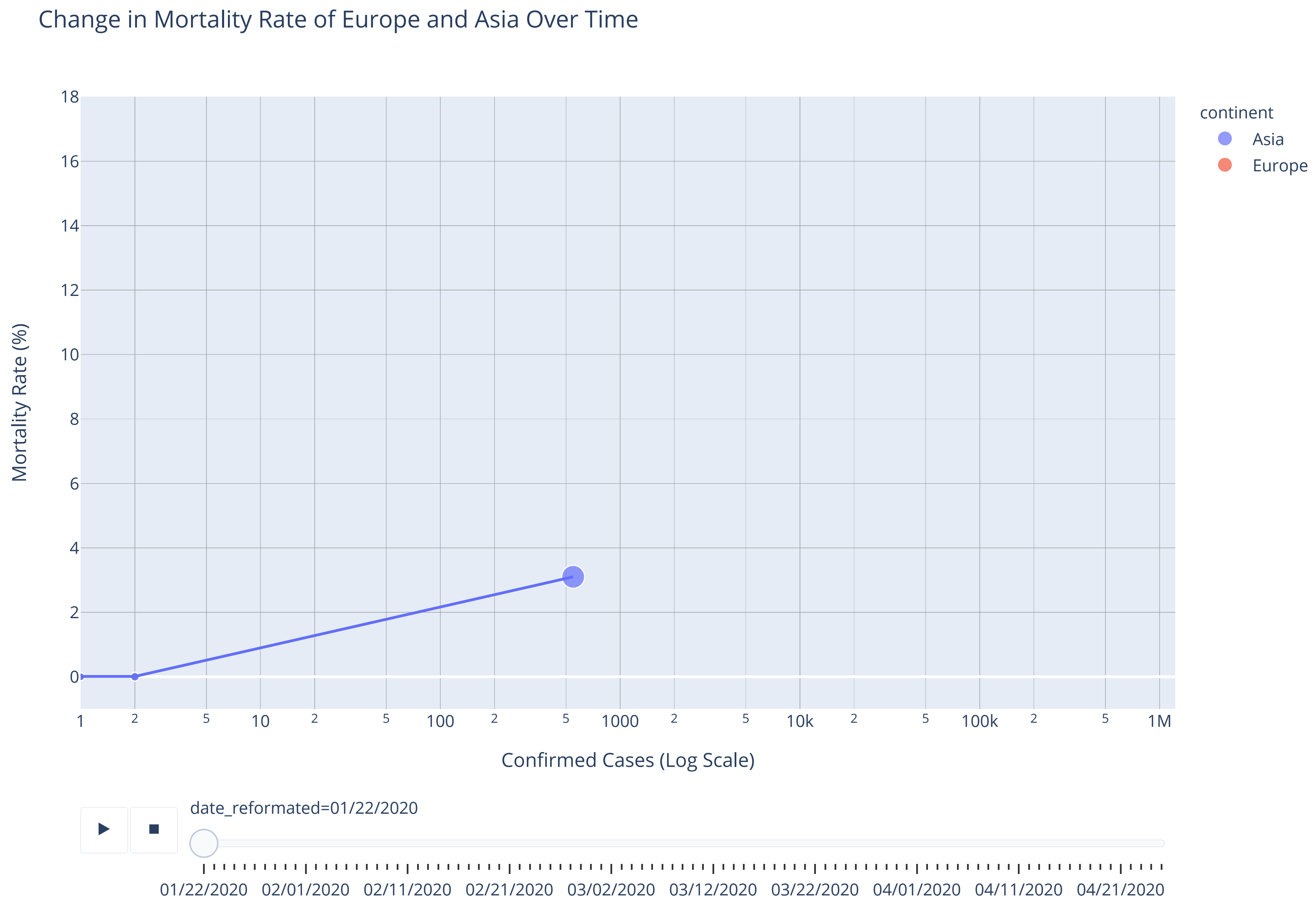
In [48]:
```
df_data = full_df.groupby(['Date', 'country'])['confirmed', 'death','continent','mortality%'].max().reset_index()
df_data["date_reformated"] = pd.to_datetime( df_data["Date"]).dt.strftime('%m/%d/%Y')
df_data = df_data[(df_data.continent == 'Europe') | (df_data.continent == 'Africa')]

fig = px.scatter(df_data, trendline = 'ols', y='mortality%',
                    x= df_data["confirmed"],
                    range_y = [-1,18],
                    range_x = [1,df_data["confirmed"].max()+1000000],
                    color= "continent",
                    hover_name="country",
                    hover_data=["confirmed","death"],
                    range_color= [0, max(np.power(df_data["confirmed"],0.3))],
                    animation_frame="date_reformated",
                    animation_group="country",
                    color_continuous_scale=px.colors.sequential.Plasma,
                    title='Change in Mortality Rate of Europe and Africa Over Time',
                    size = np.power(df_data["confirmed"]+1,0.3)-0.5,
                    size_max = 30,
                    log_x=True,
                    height =700,
                    )
fig.update_coloraxes(colorscale="hot")
fig.update(layout_coloraxis_showscale=False)
fig.update_xaxes(title_text="Confirmed Cases (Log Scale)")
fig.update_yaxes(title_text="Mortality Rate (%)")
fig.show()
```



```
In [49]: df_data = full_df.groupby(['Date', 'country'])['confirmed', 'death','continent','mortality%'].max().reset_index()
df_data["date_reformated"] = pd.to_datetime( df_data["Date"]).dt.strftime('%m/%d/%Y')
df_data = df_data[(df_data.continent == 'Europe') | (df_data.continent == 'Asia')]

fig = px.scatter(df_data, trendline = 'ols', y='mortality%',
                    x= df_data["confirmed"],
                    range_y = [-1,18],
                    range_x = [1,df_data["confirmed"].max()+1000000],
                    color= "continent",
                    hover_name="country",
                    hover_data=["confirmed","death"],
                    range_color= [0, max(np.power(df_data["confirmed"],0.3))],
                    animation_frame="date_reformated",
                    animation_group="country",
                    color_continuous_scale=px.colors.sequential.Plasma,
                    title='Change in Mortality Rate of Europe and Asia Over Time',
                    size = np.power(df_data["confirmed"]+1,0.3)-0.5,
                    size_max = 30,
                    log_x=True,
                    height =700,
                    )
fig.update_coloraxes(colorscale="hot")
fig.update(layout_coloraxis_showscale=False)
fig.update_xaxes(title_text="Confirmed Cases (Log Scale)")
fig.update_yaxes(title_text="Mortality Rate (%)")
fig.show()
```



Conclusions :

- African countries with high confirmed counts tend to have high mortality rate yet with low confirmed cases there is also high mortality rate. Which suggests a weak healthcare system with few exceptions.
- European countries tend to have low mortality rate with low confirmed cases yet as the confirmed cases count increase so does the mortality rate which suggests capable healthcare system yet inefficiant at large numbers.
- Asian countries tend to have low mortality rates regardless of the confirmed cases count yet as the count increases more than 80K they tend to have higher mortality rates yet below the average of European countries.

Investigating countries with highest mortality rates

Investigating high mortality rate in countries with correlation to high case count to find most critical countries affected by the current pandamic.

In [50]:
```
df_temp = full_df[(full_df.confirmed > 2500) & (full_df.Date == full_df.Date.max())][['country','confirmed','death','recovered'
,'density pop/km2','mortality%','confirmed%']].sort_values(by = 'mortality%',ascending = False).nlargest(10,'mortality%')
df_temp.style.background_gradient(cmap='Blues',subset=["confirmed"])\
                        .background_gradient(cmap='Reds',subset=["death"])\
                        .background_gradient(cmap='Greens',subset=["recovered"])\
                        .background_gradient(cmap='Purples',subset=["density pop/km2"])\
                        .background_gradient(cmap='YlOrBr',subset=["mortality%"])\
                        .background_gradient(cmap='bone_r',subset=["confirmed%"])
```

Out[50]:

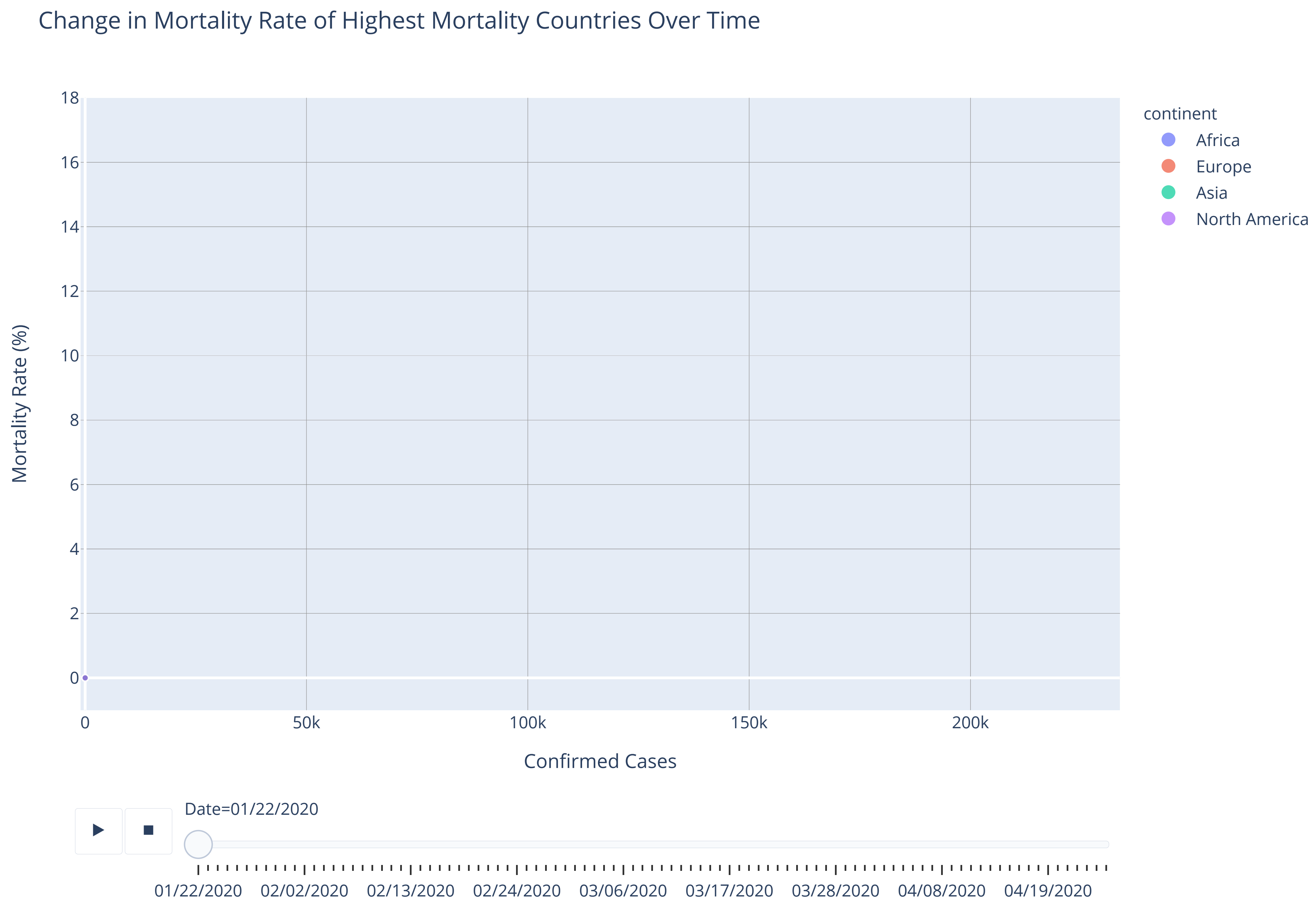
	country	confirmed	death	recovered	density pop/km2	mortality%	confirmed%
1614	Belgium	45325	6917	10417	376	15.2609	0.3933
5794	France	161644	22648	45372	123	14.011	0.241
16434	United Kingdom	149569	20381	774	274	13.6265	0.2251
7884	Italy	195351	26384	63120	200	13.5059	0.3242
284	Algeria	3256	419	1479	18	12.8686	0.0076
15009	Sweden	18177	2192	1005	23	12.0592	0.1754
11304	Netherlands	37384	4424	102	420	11.8339	0.2143
14629	Spain	223759	22902	95708	93	10.2351	0.4767
10449	Mexico	13842	1305	7149	64	9.42783	0.0109
7409	Indonesia	8607	720	1042	141	8.36528	0.0032

Visualizing rate of change of mortality rate through time for top countries

Exploring the rate of change of the mortality percentage through time as well as continents of each country and its case count for the top ten affected countries.

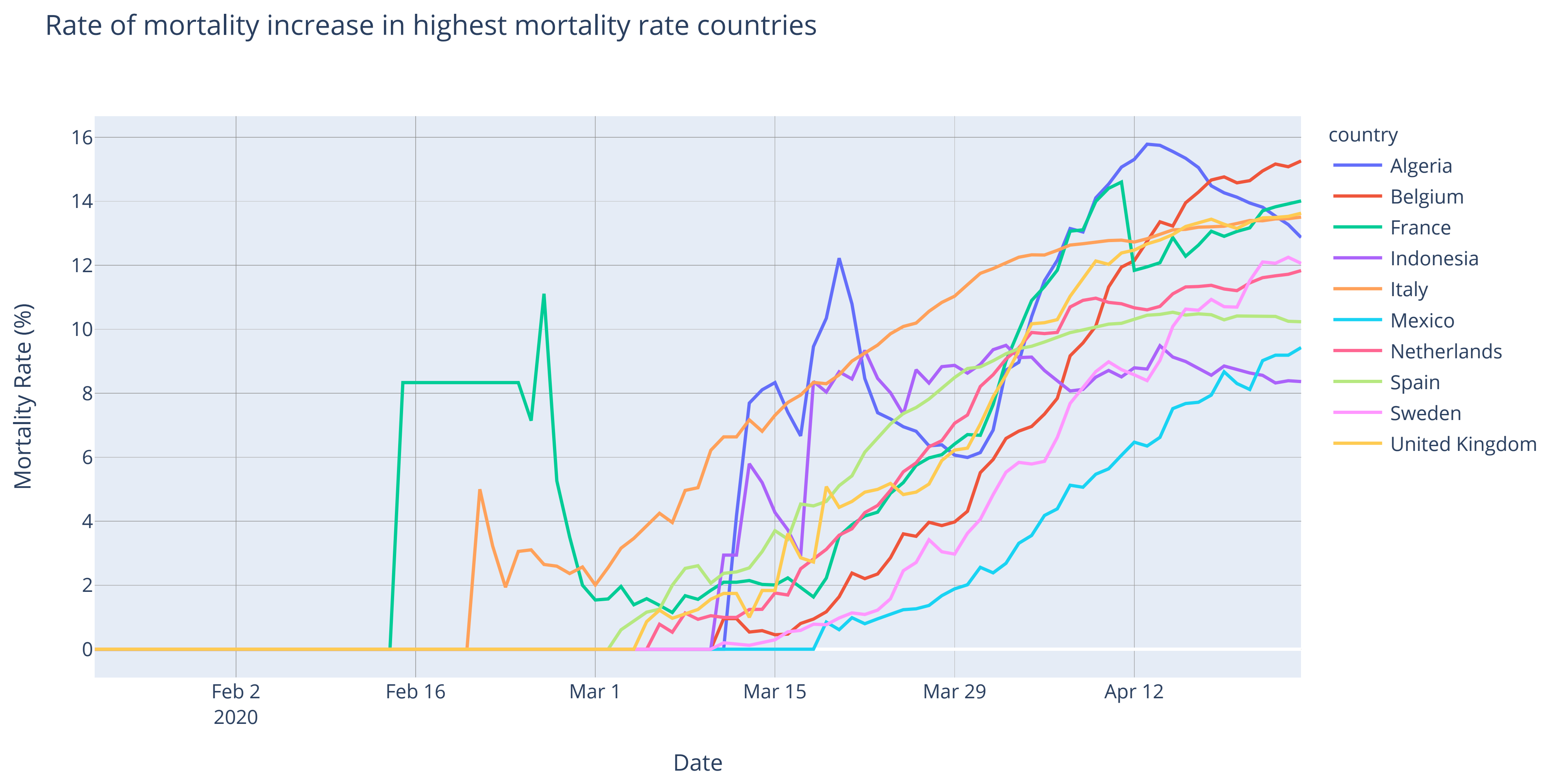
In [51]:
```
filter_df = full_df[(full_df.Date >= full_df.Date.quantile(1).strftime('%m/%d/%Y')) & (full_df.confirmed > 2500) ][['Date','country','continent','confirmed','death','recovered','density pop/km2','mortality%','confirmed%']].sort_values(by = 'mortality%',ascending = False)
choice = filter_df.nlargest(10,'mortality%')['country']
df_temp = full_df.merge(choice,how = 'right',on = 'country')
df_temp['Date'] = pd.to_datetime(df_temp['Date']).dt.strftime('%m/%d/%Y')

fig = px.scatter(df_temp,trendline='ols', y='mortality%',
                    x = df_temp["confirmed"],
                    color= "continent",
                    hover_name="country",
                    hover_data=["confirmed","death"],
                    range_y = [-1,18],
                    range_x = [-1000,df_temp["confirmed"].max()+10000],
                    range_color= [0, max(np.power(df_temp["confirmed"],0.3))],
                    animation_frame=df_temp["Date"],
                    animation_group=df_temp["country"],
                    color_continuous_scale=px.colors.sequential.Plasma,
                    title='Change in Mortality Rate of Highest Mortality Countries Over Time',
                    size = np.power(df_temp["confirmed"]+1,0.3)-0.5,
                    size_max = 30,
                    log_x=False,
                    height =700
                    )
fig.update_coloraxes(colorscale="hot")
fig.update(layout_coloraxis_showscale=False)
fig.update_xaxes(title_text="Confirmed Cases")
fig.update_yaxes(title_text="Mortality Rate (%)")
fig.show()
```



In [52]:
```
df_temp1 = full_df[(full_df.Date == full_df.Date.max()) & (full_df.confirmed > 2500)].sort_values( by = 'mortality%').nlargest(
10,'mortality%')
df_temp = full_df.merge(df_temp1,on = 'country')
fig = px.line(df_temp,x = 'Date_x', y = 'mortality%_x',color = 'country',title='Rate of mortality increase in highest mortality
rate countries')
fig.update_layout(xaxis_title='Date',
                  yaxis_title="Mortality Rate (%)")

fig.show()
```



Conclusions :

- African countries with high confirmed counts tend to have high mortality rate yet with low confirmed cases there is also high mortality rate. Which suggests a weak healthcare system with few exceptions.
- European countries tend to have low mortality rate with low confirmed cases yet as the confirmed cases count increase so does the mortality rate which suggests capable healthcare system yet inefficiant at large numbers.
- Asian countries tend to have low mortality rates regardless of the confirmed cases count yet as the count increases more than 80K they tend to have higher mortality rates yet below the average of European countries.

Countries with the highest confirmed to population ratio

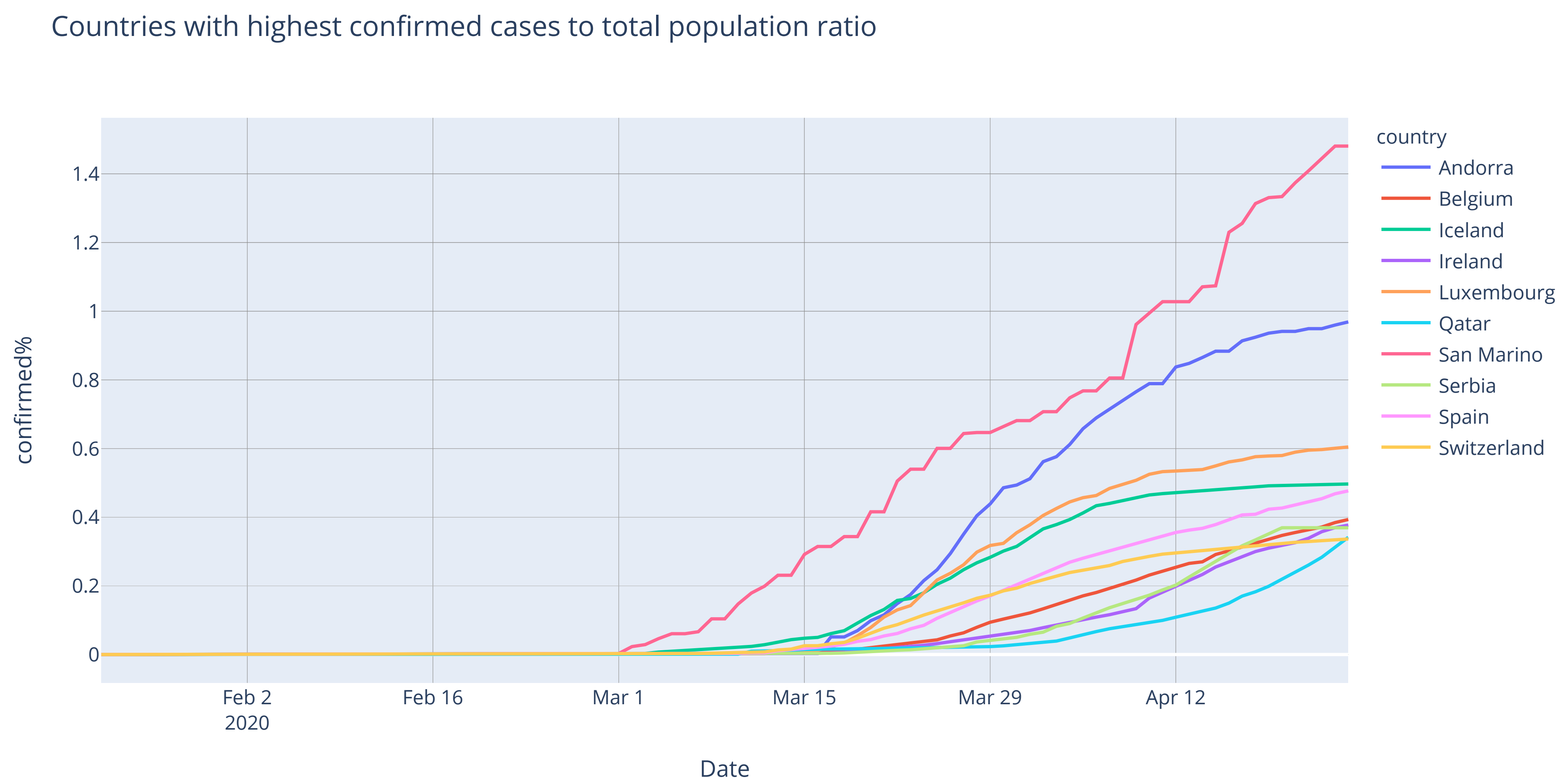
Investigating Critical countries where high percentage of their total population are infected.

In [53]:
```python
df_temp = full_df[(full_df.Date == full_df.Date.max()) & (full_df.confirmed > 200)][['country','continent','confirmed','death',
'recovered','density pop/km2','mortality%','pop','confirmed%']].sort_values(by = 'confirmed%',ascending = False).nlargest(10,'c
onfirmed%')
df_temp.style.background_gradient(cmap='Blues',subset=["confirmed"])\
                          .background_gradient(cmap='Reds',subset=["death"])\
                          .background_gradient(cmap='Greens',subset=["recovered"])\
                          .background_gradient(cmap='Purples',subset=["density pop/km2"])\
                          .background_gradient(cmap='YlOrBr',subset=["mortality%"])\
                          .background_gradient(cmap='bone_r',subset=["confirmed%"])\
                          .background_gradient(cmap='Blues',subset=["pop"])
```

Out[53]:

	country	continent	confirmed	death	recovered	density pop/km2	mortality%	pop	confirmed%
13489	San Marino	Europe	513	40	64	568	7.79727	34641	1.4809
379	Andorra	Europe	738	40	344	164	5.42005	76177	0.9688
9594	Luxembourg	Europe	3711	85	3088	237	2.29049	613894	0.6045
7219	Iceland	Europe	1790	10	1570	3.5	0.558659	360390	0.4967
14629	Spain	Europe	223759	22902	95708	93	10.2351	4.69346e+07	0.4767
1614	Belgium	Europe	45325	6917	10417	376	15.2609	1.15245e+07	0.3933
7694	Ireland	Europe	18561	1063	9233	70	5.72706	4.9215e+06	0.3771
8644	Serbia	Europe	6630	125	870	165	1.88537	1.79567e+06	0.3692
12824	Qatar	Asia	9358	10	929	237	0.10686	2.74048e+06	0.3415
15104	Switzerland	Europe	28894	1599	21300	208	5.53402	8.58655e+06	0.3365

In [54]:
```python
sub_df = full_df[full_df.Date == full_df.Date.max()].nlargest(10,'confirmed%')
sub_df
pxplotline(full_df,sub_df,'confirmed%',x ='Date',title='Countries with highest confirmed cases to total population ratio',hd=[
'pop','death','confirmed'])
```



Conclusions :

- All top ten confirmed cases % to total population are European countries. As of 25th of April.
- San Marino is the country with highest confirmed cases % to total population at almost 1.5% as of 25th of April.
- Spain has the highest confirmed cases on the list and with 5th country in the world with confirmed cases % to total population at almost 0.5%. As of 25th of April.
- Italy with the highest population and second to highest mortality rate comes as the 11th highest country with confirmed cases % to total population at 0.32%. As of 25th of April.
- Beligum with the highest mortality rate on the list (15.3%) comes as the 6th highest country with confirmed cases % to total population at 0.32% which makes it at critical point. As of 25th of April.

Exploring Population density wrt confirmed cases

Investigating countries with highest population density and how the virus spreads over time. As well as exploring the relationship between them.

In [55]:
```
df_temp = full_df[(full_df.Date == full_df.Date.max()) & (full_df.confirmed > 2500) & (full_df['density pop/km2'] < 3500)][['country','continent','confirmed','death','recovered','density pop/km2','mortality%','pop','confirmed%']].sort_values(by = 'density pop/km2',ascending = False).nlargest(20,'density pop/km2')
df_temp.style.background_gradient(cmap='Blues',subset=["confirmed"])\
                        .background_gradient(cmap='Reds',subset=["death"])\
                        .background_gradient(cmap='Greens',subset=["recovered"])\
                        .background_gradient(cmap='Purples',subset=["density pop/km2"])\
                        .background_gradient(cmap='YlOrBr',subset=["mortality%"])\
                        .background_gradient(cmap='bone_r',subset=["confirmed%"])\
                        .background_gradient(cmap='Blues',subset=["pop"])
```

Out[55]:

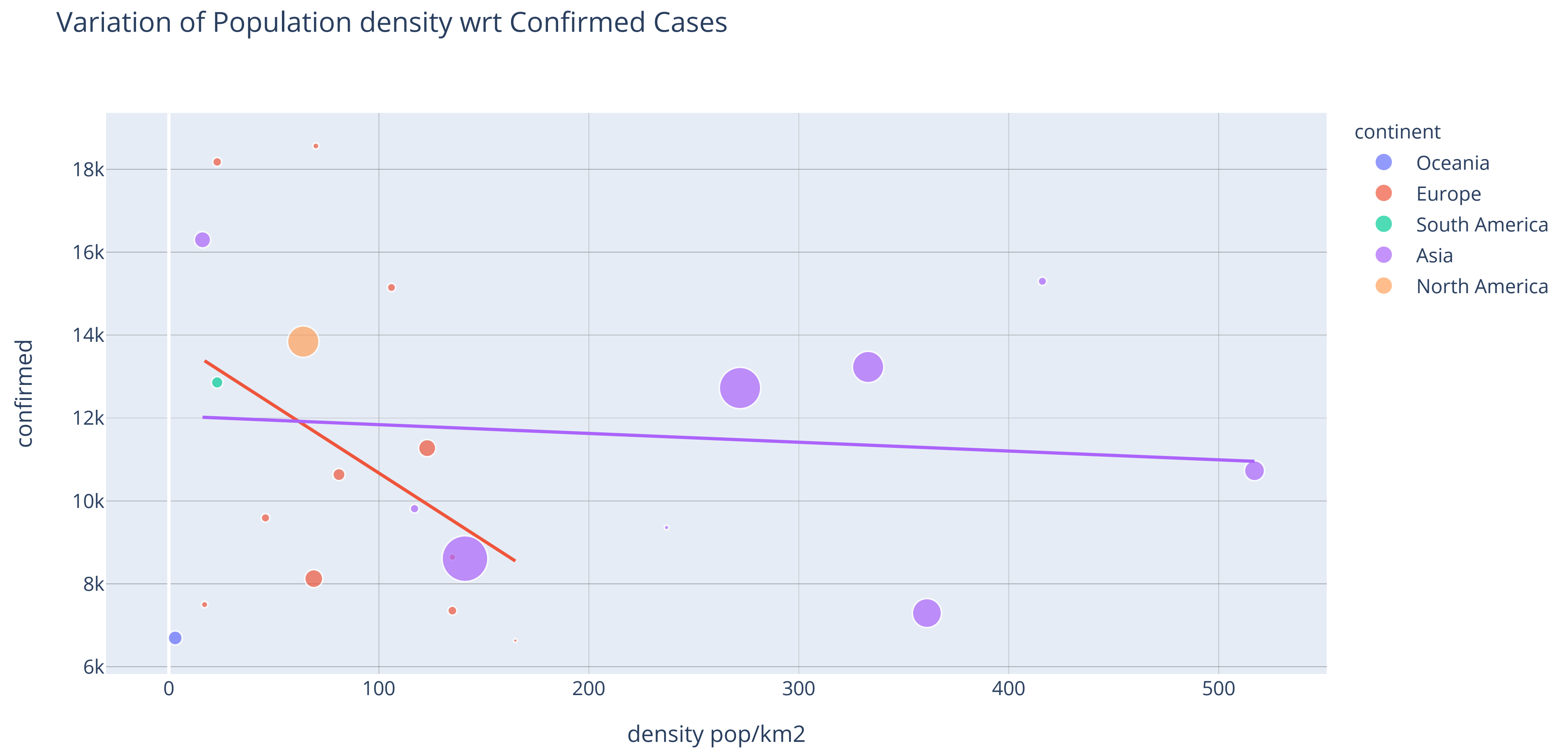
	country	continent	confirmed	death	recovered	density pop/km2	mortality%	pop	confirmed%
1234	Bahrain	Asia	2588	8	1160	1983	0.309119	1.5433e+06	0.1677
1329	Bangladesh	Asia	4998	140	113	1169	2.80112	1.68288e+08	0.003
8454	Korea, South	Asia	10728	242	8717	517	2.25578	5.17806e+07	0.0207
11304	Netherlands	Europe	37384	4424	102	420	11.8339	1.74458e+07	0.2143
7789	Israel	Asia	15298	199	6435	416	1.30082	9.17325e+06	0.1668
7314	India	Asia	26283	825	5939	414	3.13891	1.35993e+09	0.0019
1614	Belgium	Europe	45325	6917	10417	376	15.2609	1.15245e+07	0.3933
12539	Philippines	Asia	7294	494	792	361	6.77269	1.0842e+08	0.0067
8074	Japan	Asia	13231	360	1656	333	2.72088	1.2601e+08	0.0105
16434	United Kingdom	Europe	149569	20381	774	274	13.6265	6.64356e+07	0.2251
12064	Pakistan	Asia	12723	269	2866	272	2.11428	2.18984e+08	0.0058
8739	Kuwait	Asia	2892	19	656	248	0.656985	4.42011e+06	0.0654
12824	Qatar	Asia	9358	10	929	237	0.10686	2.74048e+06	0.3415
9594	Luxembourg	Europe	3711	85	3088	237	2.29049	613894	0.6045
6174	Germany	Europe	156513	5877	109800	233	3.75496	8.31493e+07	0.1882
4749	Dominican Republic	North America	5926	273	822	216	4.60682	1.03583e+07	0.0572
15104	Switzerland	Europe	28894	1599	21300	208	5.53402	8.58655e+06	0.3365
7884	Italy	Europe	195351	26384	63120	200	13.5059	6.02528e+07	0.3242
8644	Serbia	Europe	6630	125	870	165	1.88537	1.79567e+06	0.3692
3419	China	Asia	83909	4636	78175	145	5.52503	1.40181e+09	0.006

Exploring densities of countries with confirmed cases within IQR of 50%

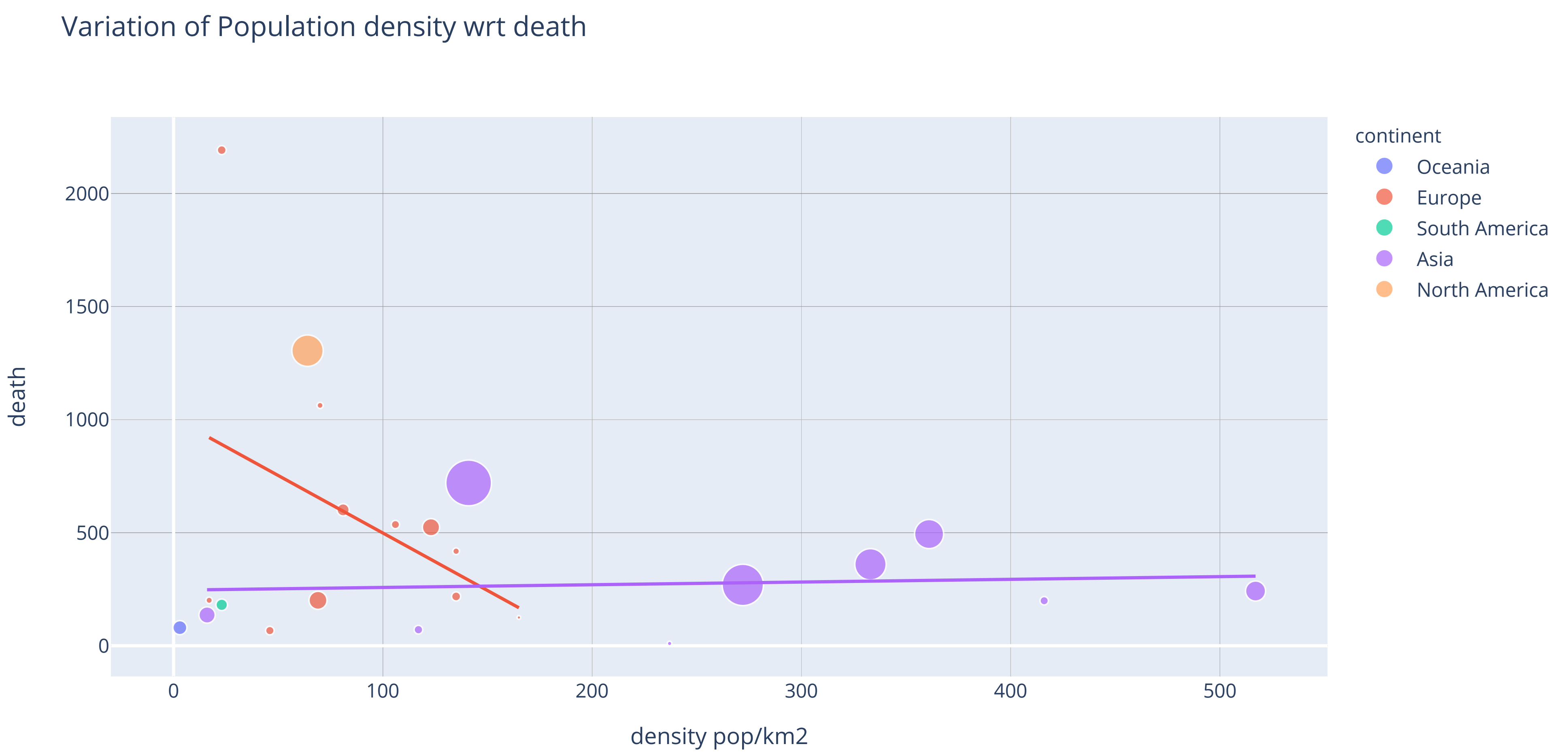
Exploring confirmed and death cases between continents and their countries, and their ordinary least squares to find insights on how they interact with each other through time.

In [56]:
```
df_temp = full_df[full_df['Date']==full_df.Date.max()]
q3 = np.percentile(df_temp.confirmed,75)
q1 = np.percentile(df_temp.confirmed,25)
IQR = q3-q1
low = -q1 + 1.2*IQR
high = q3 + 2.5*IQR
df_temp = df_temp[(df_temp['confirmed']>low) & (df_temp['confirmed']<high)]
df_temp = df_temp[df_temp['density pop/km2'] < 3500]
df_temp = df_temp[df_temp['confirmed'] > 2500]
```

In [57]:
```
px.scatter(df_temp,trendline = 'ols',y='confirmed',x='density pop/km2', size = 'pop', color='continent',hover_data=['country'],
title='Variation of Population density wrt Confirmed Cases')
```



In [58]:
```
px.scatter(df_temp,trendline = 'ols',y='death',x='density pop/km2', size = 'pop', color='continent',hover_data=['country'],titl
e='Variation of Population density wrt death')
```



Conclusions :

- Seven out of the top dense populations are Asian countries.
- Three out of the top dense populations are European countries.
- Asian countries with high density tend to have more confirmed cases with a linear positive relationship yet a weaker linear positive relationship with their death cases.
- European countries on the other hand has a linear strong negative relationship with both their confirmed and death cases wrt their population density.

Exploring Population illiteracy rate wrt confirmed cases

Investigating countries with highest illiterate rate to their supposed educated population wrt confirmed cases, and how the virus spreads over time. As well as exploring the relationship between them.

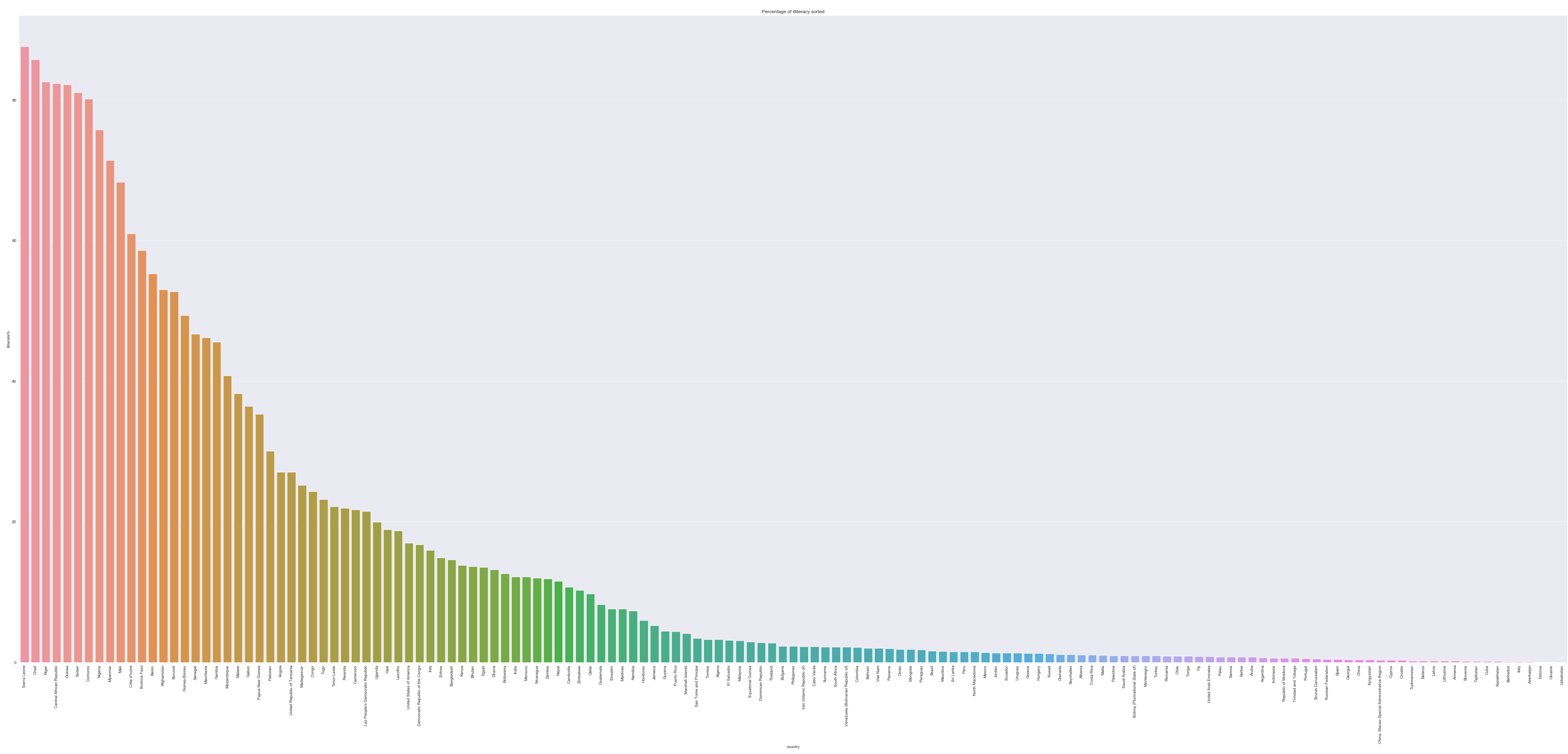
Exploring the countries with the highest illiteracy rate

In [59]:
```python
edu_full_df = full_df.merge(edu_df[['ISO','illiterate%']],on = 'ISO')
edu_full_df_temp = edu_full_df[edu_full_df.Date == edu_full_df.Date.max()].nlargest(20,'illiterate%')
edu_full_df_temp[['country','continent','confirmed','death','recovered','active','illiterate%']].style.background_gradient(cmap
='Blues',subset=["confirmed"])\
                            .background_gradient(cmap='Reds',subset=["death"])\
                            .background_gradient(cmap='Greens',subset=["recovered"])\
                            .background_gradient(cmap='Purples',subset=["active"])\
                            .background_gradient(cmap='YlOrBr',subset=["illiterate%"])\
```

Out[59]:

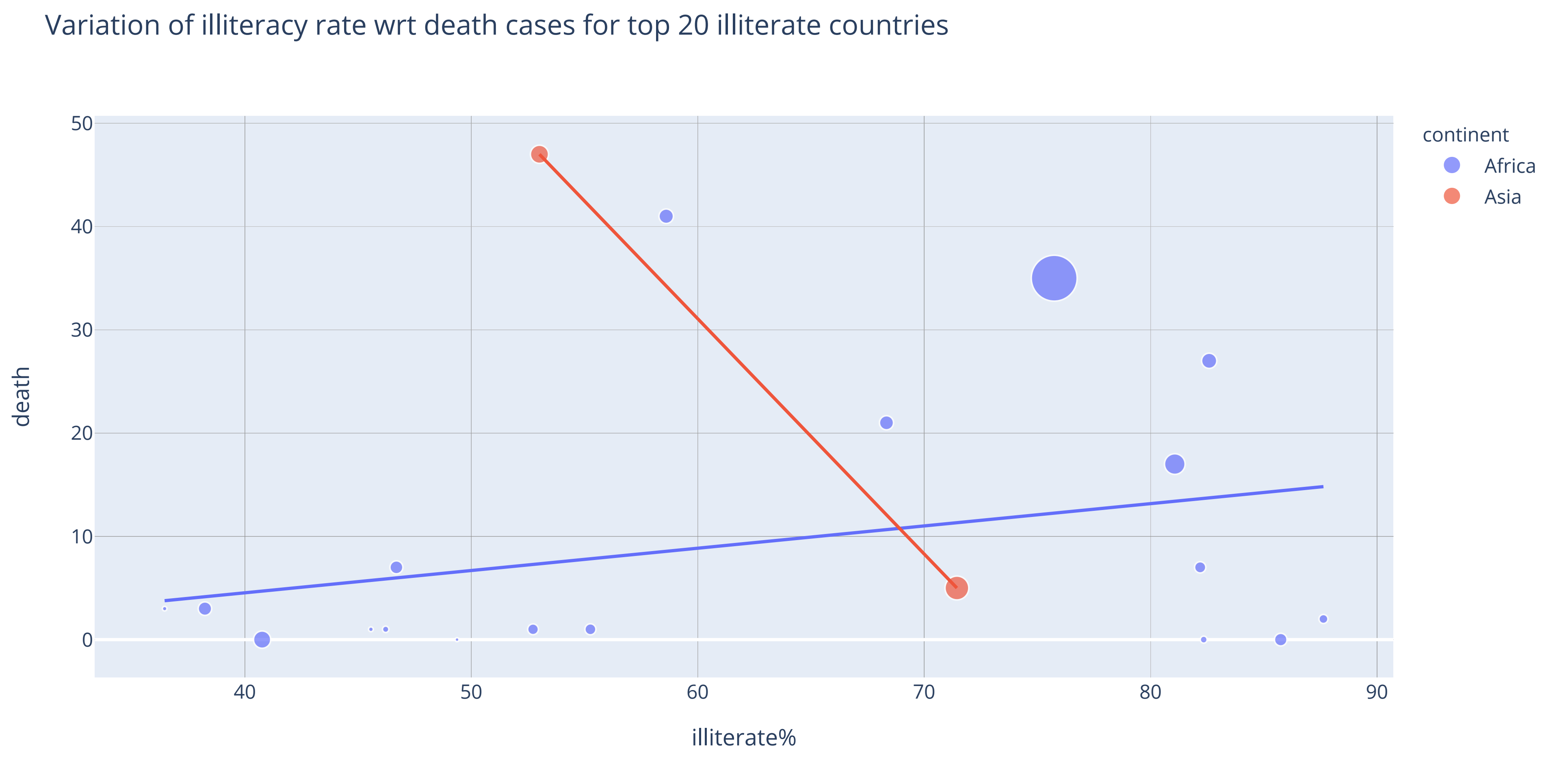
	country	continent	confirmed	death	recovered	active	illiterate%
10639	Sierra Leone	Africa	82	2	10	70	87.6322
2279	Chad	Africa	46	0	15	31	85.7461
8834	Niger	Africa	684	27	325	332	82.5861
2184	Central African Republic	Africa	16	0	10	6	82.3446
5319	Guinea	Africa	996	7	208	781	82.1891
11114	Sudan	Africa	213	17	19	177	81.0677
8929	Nigeria	Africa	1182	35	222	925	75.7421
1804	Burma	Asia	146	5	10	131	71.4451
7599	Mali	Africa	370	21	91	258	68.3345
1709	Burkina Faso	Africa	629	41	442	146	58.6067
1139	Benin	Africa	54	1	27	26	55.2592
94	Afghanistan	Asia	1463	47	188	1228	53.0102
1899	Burundi	Africa	11	1	4	6	52.7257
5414	Guinea-Bissau	Africa	52	0	3	49	49.3701
10449	Senegal	Africa	614	7	276	331	46.6922
7789	Mauritania	Africa	7	1	6	0	46.2173
4749	Gambia	Africa	10	1	8	1	45.5661
8454	Mozambique	Africa	70	0	12	58	40.7616
7314	Malawi	Africa	33	3	4	26	38.2353
4654	Gabon	Africa	176	3	30	143	36.4562

In [60]:
```python
data = edu_df.sort_values('illiterate%',ascending = False)
fig, ax = plt.subplots(figsize = (70,30))
sns.barplot(data = data, x = 'country',y = 'illiterate%')
_=ax.set_xticklabels(labels=data.country, rotation=90)
plt.title('Percentage of illiteracy sorted');
```



In [61]:
```
edu_full_df = full_df.merge(edu_df[['ISO','illiterate%']],on = 'ISO')
edu_full_df_temp = edu_full_df[edu_full_df.Date == edu_full_df.Date.max()].nlargest(20,'illiterate%')
px.scatter(edu_full_df_temp,trendline="ols",x='confirmed',y='illiterate%', size = 'pop', color='continent',hover_data=['country','continent'],title='Variation of illiteracy rate wrt confirmed cases for top 20 illiterate countries')
```

In [62]:
```
edu_full_df = full_df.merge(edu_df[['ISO','illiterate%']],on = 'ISO')
edu_full_df_temp = edu_full_df[edu_full_df.Date == edu_full_df.Date.max()].nlargest(20,'illiterate%')
px.scatter(edu_full_df_temp,trendline="ols",y='death',x='illiterate%', size = 'pop', color='continent',hover_data=['country'],title='Variation of illiteracy rate wrt death cases for top 20 illiterate countries')
```



Conclusions :

- 18 of the top illiterate countries are African countries
- Two of the top illiterate countries are Asian countries
- African countries has a moderate positive relationship between illiteracy rate and confirmed and death cases suggesting that illiteracy may have a slight effect on the spread of virus in Africa.
- Asian countries has a strong negative relationship between illiteracy rate and confirmed and death cases, suggesting that illiteracy has no effect on the spread of virus in Asia.

Final Thoughts

After a lengthy analysis of mentioned data, and several conclusions from varios sections of this project, we could safely assume the following:-

- United states of America is considered the new virus epicenter with most confirmed cases and highest active cases.
- Europe is the most critical continent with very high active cases, confirmed cases and mortality rates in the globe.
- Population density has moderate positive effect on the virus spread in African countries, yet negative effect in European countries.
- Illiteracy percentage has strong effect on the virus spread in African countries, yet negative effect in Asian countries. Suggesting other factors contributing to it and requires further analysis.
- The virus outbreak that started in Asia rapidly shifted its effects to westarn countries and more dense locations.
- Some European countries are on the verge of a catastrophy while some have faced the pandamic very well.
- Lockdown was very effective in all countries in which it was applied correctly.
- African and developing countries are at risk of turning to critical points. With high mortality rates and high case counts as well as low recovery rates.
- Countries with high confirmed cases rate should apply a total Lockdown and enhance thier healthcare system.
- Healthcare systems are the main anchor to this pandamic and combined with an efficient Lockdown are the most promising solution.

And Finally Stay Home and Stay Safe.