Week 9. Final project: COVID-19 and government response

I decided to conduct my Week 9 research project on exploring COVID statistics and its correlation with new indicators of government response or measures taken against COVID virus spreading.

Research questions

- is there a significant correlation between cumulative deaths from COVID-19 and maximum government response index?
- is there a significant correlation between government response index and government funding healthcare in previous years?
- is there strong positive correlation between confirmed COVID-19 cases and government response index over time?

Hypothesis to check

- yes, there is significant positive correlation between deaths from COVID-19 and maximum government response;
- no, strict response to COVID-19 in 2020 and better funding healthcare systems in 2019 are not significantly correlated.
- yes, correlation between confirmed cases and government response over time is strong.

To explore COVID statistics I use COVID-19 Data Repository by the Center for Systems Science and Engineering at Johns Hopkins University, or JHU CSSE COVID-19 Dataset (here - Dataset 1) accessing it via API.

To explore new indicators of government response (measures taken against COVID virus spreading) I use Oxford Covid-19 Government Response Tracker (here - Dataset 2).

That is why Part I of this project presented as chapter "Dataset 1: Johns Hopkins University & Medicine (JHU)" and its paragraphs, and Part II is presented as "Dataset 2: Oxford Covid-19 Government Response Tracker".

Dataset 1: Johns Hopkins University & Medicine (JHU)

Timeseries from January 22, 2020 to August 20, 2020 are available for downloading: https://github.com/CSSEGISandData/COVID-19/tree/master/csse_covid_19_data/csse_covid_19_time_series (https://github.com/CSSEGISandData/COVID-19/tree/master/csse_covid_19_data/csse_covid_19_time_series)

Raw data links to cumulative data:

Confirmed cases: https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_time_series/time_series_covid19_confirmed_global.csv (https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_time_series/time_series_covid19_confirmed_global.csv)

Deaths: https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_time_series/time_series_covid19_deaths_global.csv (https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_time_series/time_series_covid19_deaths_global.csv)

Recovered: https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_time_series/time_series_covid19_recovered_global.csv (https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_time_series/time_series_covid19_recovered_global.csv)

1. Confirmed cases statistics

§1.1. Cleaning and preparing for animation

The GIS technologies have played an important role in many aspects, including the data integration, and geospatial visualization of epidemic information, spatial tracking of confirmed cases, prediction of regional transmission, and many more. These provide support information for government sectors to fight against the COVID-19 spreading.

The Center for Systems Science and Engineering (CSSE) at Johns Hopkins University & Medicine (JHU) had provided the dashboard created with ESRI ArcGIS operation dashboard (https://www.arcgis.com/apps/opsdashboard/index.html#/bda7594740fd40299423467b48e9ecf6 (https://www.arcgis.com/apps/opsdashboard/index.html#/bda7594740fd40299423467b48e9ecf6)). But feature for visualizing the change of data overtime on the map is missing. Later JHU created animated map on confirmed cases here https://coronavirus.jhu.edu/data/animated-world-map (https://coronavirus.jhu.edu/data/animated-world-map) , separately from the dashboard, but the users can only observe the map changing colors, they have no access to view the actual numbers or zoom in the map, as it is not interactive and does not show the actual data.

So I decided to create animated maps to explore data changes over time. In order to do that my current dataset structure should be changed. Now the data structure is that every day's statistics is a separate column, so the values are "scattered" in unique cells for each day and country; I will move all the values to a single "value" column, and move all days labels from columns names to single "Date" column. It will transform the dataset to its long variation with repeating country rows and date rows.

In [1]:
```
import pandas as pd #to work with tabular data
import pycountry #to get the three-letter country codes ISO 3166-1 for each country

df_cases=pd.read_csv("https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/

# Aggregate the dataset
df_cases = df_cases.drop(columns=['Province/State','Lat','Long'])
df_cases = df_cases.groupby('Country/Region').agg('sum')
date_list = list(df_cases.columns)

# Get the country codes for each country
#list(pycountry.countries) #uncomment to load the list of available data

def get_country_code(name):
    """
    Return ISO-3 letter code for country by its name;
    Return None if name is not found in the pycountry.countries
    """
    try:
        return pycountry.countries.lookup(name).alpha_3
    except:
        return None

df_cases['Country'] = df_cases.index
```

Several countries' names are written differently than pycountry expects, so I change their names to match and get the code

In [2]:
```
df_cases.loc[df_cases.Country=="Burma",'Country']='Myanmar'
df_cases.loc[df_cases.Country=="Brunei",'Country']='Brunei Darussalam'
df_cases.loc[df_cases.Country=="Iran",'Country']='Iran, Islamic Republic of'
df_cases.loc[df_cases.Country=="Congo (Brazzaville)",'Country']='Congo, The Democratic Republic of the'
df_cases.loc[df_cases.Country=="Congo (Kinshasa)",'Country']='Republic of the Congo'
df_cases.loc[df_cases.Country=="Cote d'Ivoire",'Country']="Côte d'Ivoire"
df_cases.loc[df_cases.Country=="Korea, South",'Country']="Korea, Republic of"
df_cases.loc[df_cases.Country=="Syria",'Country']="Syrian Arab Republic"
df_cases.loc[df_cases.Country=="Taiwan*",'Country']="Taiwan, Province of China"
df_cases.loc[df_cases.Country=="Russia",'Country']='Russian Federation'
df_cases.loc[df_cases.Country=="West Bank and Gaza",'Country']='Palestine, State of'
df_cases.loc[df_cases.Country=="Venezuela",'Country']='Venezuela, Bolivarian Republic of'
df_cases.loc[df_cases.Country=="US",'Country']='United States'
```

As soon as names are unified, I can add their ISO-3 codes.

In [3]:
```
```

188

In [4]:
```
```

In [5]:
```
# Transform the dataset in a long format
```

In [6]:
```
# Rename columns
```

In [7]:
```
```

There is one "None" value left in the df_confirmed_long[0:60] slice ("Diamond Princess").

In [8]:
```
```

In [9]:
```
#df_cases[120:160] #checking the slice 3 in the dataset on confirmed cases, uncomment to load
```

There are "None" values for "MS Zaandam", "Holy See" (Vatican) and "Kosovo" left in the df_confirmed_long[60:120] slice. First is not a country, second is too small and excessive to dataset (population is 809 people), but Kosovo is important to show on the map as this European country has population more than 1.8 mln people and 11 thousands of confirmed cases.

The problem is, that "Kosovo" is not listed in pycountry dictionary (although the World Bank added XKX code to Kosovo in June 2017 according to archives https://libraries.acm.org/binaries/content/assets/libraries/archive/world-bank-list-of-economies.pdf (https://libraries.acm.org/binaries/content/assets/libraries/archive/world-bank-list-of-economies.pdf)), that is why I need to "fix Kosovo" after adding all other codes with apply(get_country_code).

In [10]:
```
#add ISO-3 code manually as it is not listed in pycountry dictionary
```

In [11]:
```
#check ISO-3 for "Kosovo" to make sure the code is applied
```

Out[11]:

	Country	ISO-3	Date	Value
92	Kosovo	XKX	1/22/20	0
280	Kosovo	XKX	1/23/20	0

	Country	ISO-3	Date	Value
468	Kosovo	XKX	1/24/20	0
656	Kosovo	XKX	1/25/20	0

Now it is safe to drop "None" values.

In [12]:
```
df_cases = df_cases.dropna()
```

Out[12]:

	Country	ISO-3	Date	Value
47	Denmark	DNK	1/22/20	0
49	Djibouti	DJI	1/22/20	0

In [13]:

Out[13]:

	Country	ISO-3	Date	Value
103	Luxembourg	LUX	1/22/20	0
105	Madagascar	MDG	1/22/20	0
106	Malawi	MWI	1/22/20	0

Dataset is cleaned and has data on 185 countries over January - August 2020.

In [14]:
```
print(len(df_cases['ISO-3'].unique().tolist()))
print(len(df_cases['ISO-3']))
```

```
185
39590
True
```

In [15]:

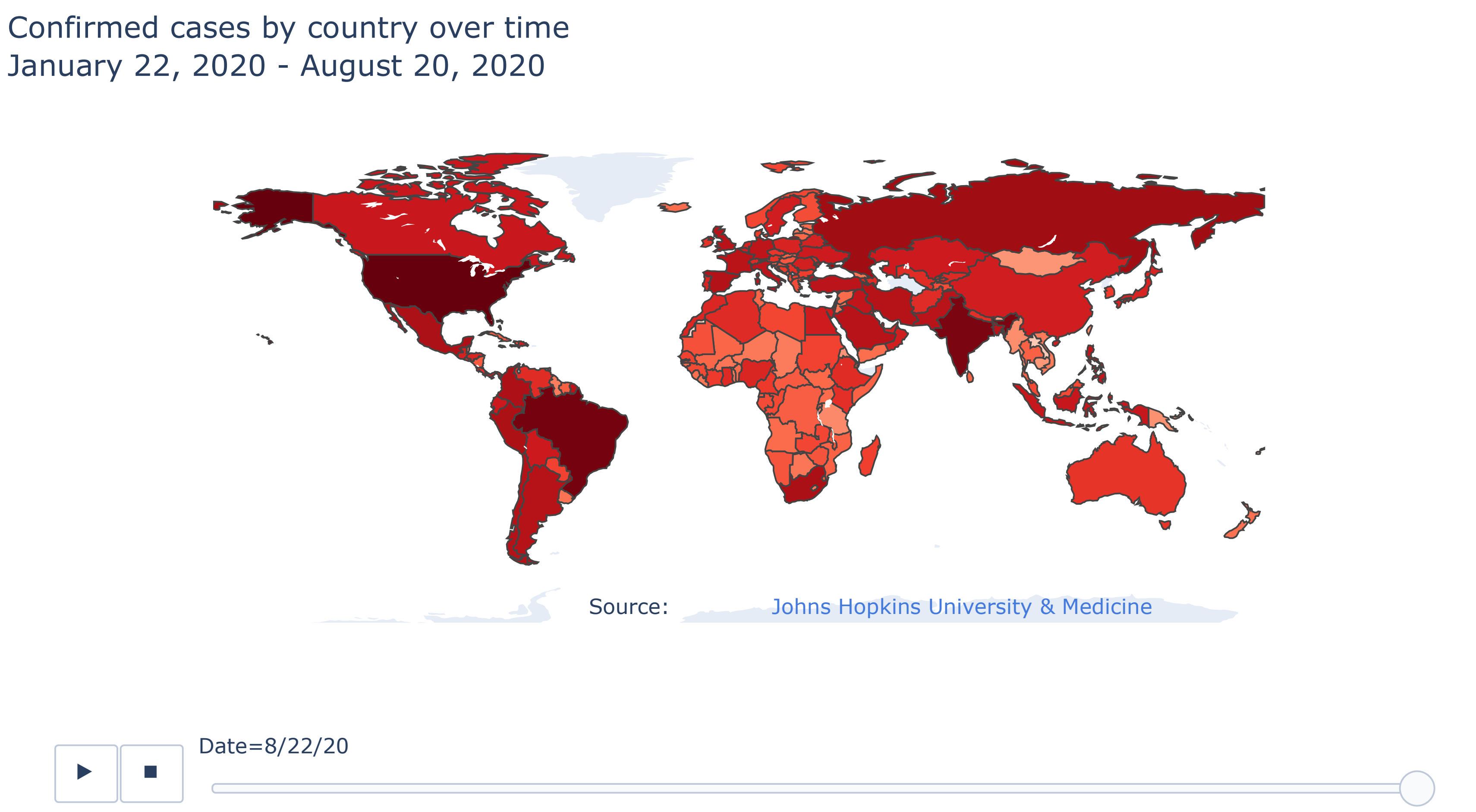
Out[15]:
```
Country    False
ISO-3      False
Date       False
Value      False
dtype: bool
```

§1.2. Animation of the map over time: cases

Now I can use Plotly Express to create animated map. The cumulative cases animation shows the total number of cases reported in each country at each point in time, regardless of how many people have recovered. Visualizing cumulative cases demonstrates the overall toll of coronavirus on a country over time.

In [103]:
```
import plotly.express as px
import numpy as np
df = df_cases
fig = px.choropleth(df,                                 # input dataframe
                    locationmode='ISO-3',               # set of locations used to map 'locations'
                    locations="ISO-3",                  # identify country by code
                    color=np.log10(df['Value']),        # identify values and replace linear scale with logari
                    hover_name="Country",               # identify column to add as name to hover information
                    animation_frame="Date",             # identify date column
                    projection="equirectangular",       # select projection
                    hover_data=[df['Value']],           # hover text
                    center = {"lat": 14.883333, "lon": 5.266667},# set map center
                    color_continuous_scale=px.colors.sequential.Reds,      # set color scale, "_r" to rever
                    range_color=[0,round(np.log10(df['Value']).max(),2)], # set the range of dataset
                   )

#customize layout
fig.update_layout(
    title_text='Confirmed cases by country over time<br>January 22, 2020 - August 20, 2020',
    geo=dict(showframe=False, showcoastlines=False, projection_type='equirectangular'),

    annotations = [dict(
        x=0.8,
        y=0.0,
        xref='paper',
        yref='paper',
        text='Source: <a href="https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid
             Johns Hopkins University & Medicine</a>',
        showarrow = False
    )],

    #customize colorbar
    coloraxis_colorbar=dict(
                            title='Confirmed',
                            tickvals=[0, 1, 2, 3, 4, 5, 6, 6.7], #customize colorbar title and ticks value
```

```
                                ticktext = ['1', '10','100', '1K', '10K', '100K', '1M', '6M'] #replace log10 
                                )

                    )
fig.show()
fig.write_html("Confirmed_map.html")
```



The map has "Play" and "Stop" buttons near the Date of observation mark, and allows zooming and observing the number of cases, ISO-3 codes and dates of observation in hover info for each country.

The map reveals later some interesting details. JHU CSSE COVID-19 Dataset does not contain information on confirmed cases in Somaliland (part of Somalia, ISO-3 code of Somalia is "SOM"), North Korea (ISO-3 code is "PRK") and Turkmenistan ("TKM").

Somaliland has declared independence, but is not recognized internationally (hence not in the ISO list), so choropleth module has to particular code to use to map the data. Plotting by country name is also not possible because Somaliland borders are not interationally set and recognized.

North Korea escalates coronavirus response, but extent of outbreak is unclear; there are no confirmed cases of COVID-19 in North Korea, the government has taken extensive measures, including quarantines and travel restrictions. North Korea didn't admit to its 1st case until July, although city of Kaesong has been focus of quarantines. Since the end of December till August, according to unofficial data North Korea has quarantined and released 25,905 people, 382 of them foreigners.

Lack of information is not surprising in the first and the second case, but Turkmenistan is missing for different reasons. There is no official statistics on COVID-19 spread in Turkmenistan at all. The state-controlled media are not allowed to use the word "coronavirus" and it has even been removed from health information brochures distributed in schools, hospitals and workplaces (according to Turkmenistan Chronicle, one of the few sources of independent news, whose site is blocked within the country). Turkmenistan 2020 population is estimated at 6.0 mln people at mid year according to UN data.

In [17]:

```
Max confirmed cases: 5667112
```

As of August 20, 2020 maximum number of cases - 5.6 mln - were confirmed in USA, and there were performed about 69.6 mln tests there. Testing has covered every 208 out of 1000 people in the country.

§1.3. Structure by region and income level: cases

I would like to see bigger picture for data, not only by country, but also by region and by income level. To make this happen I add region and income level columns to all countries. I use the World Bank data to create dataframe-converter and merge additional columns to my dataset.

In [18]:

```
df_convert=pd.read_csv("iso3_region_income_country.csv")
```

Out[18]:

	ISO-3	Region	IncomeLevel	Country_WB
0	AFG	South Asia	Low income	Afghanistan
1	AGO	Sub-Saharan Africa	Lower middle income	Angola
2	ALB	Europe & Central Asia	Upper middle income	Albania

In [19]:
```
df_cases=df_cases.merge(df_convert,on='ISO-3')
```

Out[19]:

	Country	ISO-3	Date	Value	Region	IncomeLevel	Country_WB
0	Afghanistan	AFG	1/22/20	0	South Asia	Low income	Afghanistan

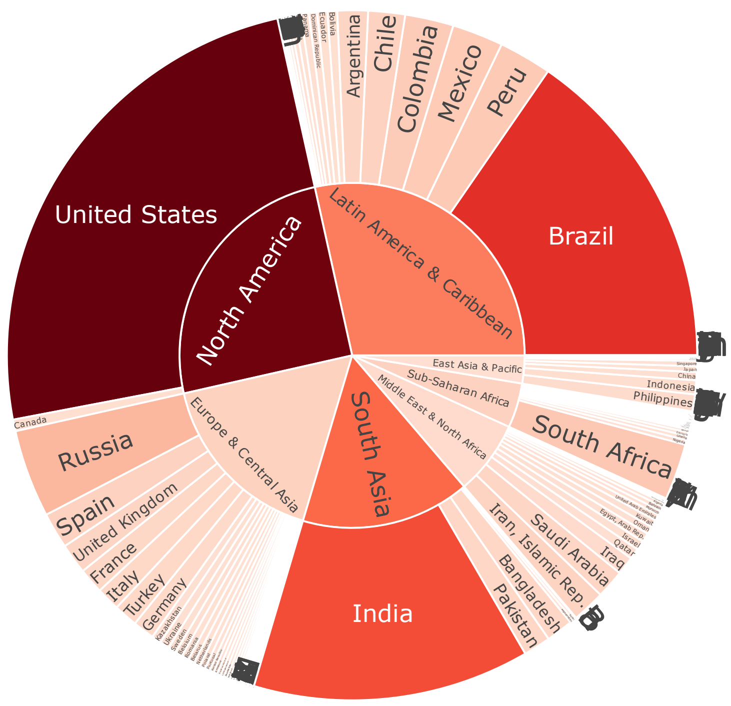
In [20]:

Out[20]:
```
Country       False
ISO-3         False
Date          False
Value         False
Region        False
IncomeLevel   False
Country_WB    False
dtype: bool
```

Interactive sunburst plot represents hierarchial data as sectors laid out over several levels of concentric rings. Next sunburst graph shows countries within world's regions where the most cases of virus were confirmed. It is United States and Brazil in Americas, India - in South Asia, Russia - in Europe and Central Asia, and South Africa in African continent.

In [21]:
```
import numpy as np
import plotly.express as px
df = df_cases[df_cases['Date']=='8/21/20'] # take the last day of observation so cumulative values are ma
fig = px.sunburst(df, path=['Region', 'Country_WB'], values=df.Value,
                  color=df.Value, color_continuous_scale='Reds',
                  title = 'Confirmed cases by regions and countries<br>by August 21, 2020',
                  color_continuous_midpoint=np.average(df.Value,weights=df.Value))
fig.show()
```

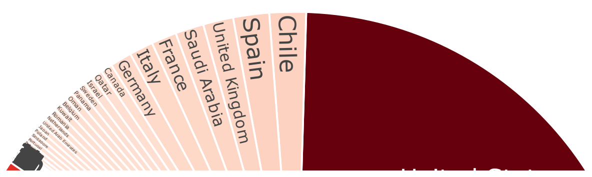
Confirmed cases by regions and countries
by August 21, 2020



Is there a pattern in terms of virus spread between different regions of income? The next sunburst graph shows that 'High income' countries and 'Upper medium income' countries cover 41% and 39% of total COVID-19 cases respectively; "Lower middle income" countries cover less than 19.5% of total COVID-19 cases, and share of cases confirmed in low income countries is about 0.5%.

In [22]:
```
df = df_cases[df_cases['Date']=='8/21/20'] # take the last day of observation so cumulative values are ma
fig = px.sunburst(df, path=['IncomeLevel', 'Country_WB'], values=df['Value'],
                  color=df['Value'], color_continuous_scale='Reds',
                  color_continuous_midpoint=np.average(df['Value'], weights=df['Value']),
                  title = 'Confirmed cases by income level and countries<br>by August 21, 2020')
fig.show()

```

Confirmed cases by income level and countries
by August 21, 2020



At first it could look like there is a correlation, as 80% of cases are confirmed in countries where income level is higher than medium. But it is importnant to note, that the number of confirmed cases is lower than the number of actual cases at all times, the main reason for that is limited testing. On one hand, this especially could make effect on COVID-19 statistics in lower income countries where the virus is harder to diagnosed due to various limitations. On the other hand, low income countries population is less globaly mobile and this factor is probably slowing down the spreading of virus there in comparison with high income countries.

In any case there are lots of controversial effects from different groups of factors and it is too early to make conclusions at this stage given the available statistics.

2. Deaths statistics analysis

§2.1. Cleaning and preparing for animation: deaths

In [23]:
```
import pandas as pd
import pycountry

df_deaths=pd.read_csv("https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data

# Aggregate the dataset
df_deaths = df_deaths.drop(columns=['Province/State','Lat','Long'])
df_deaths = df_deaths.groupby('Country/Region').agg('sum')
date_list = list(df_deaths.columns)

df_deaths['Country'] = df_deaths.index
```

Several countries' names are written differently than pycountry expects, so I change their names to match and get the code

In [24]:
```
df_deaths.loc[df_deaths.Country=="Burma",'Country']='Myanmar'
df_deaths.loc[df_deaths.Country=="Brunei",'Country']='Brunei Darussalam'
df_deaths.loc[df_deaths.Country=="Iran",'Country']='Iran, Islamic Republic of'
df_deaths.loc[df_deaths.Country=="Congo (Brazzaville)",'Country']='Congo, The Democratic Republic of the'
df_deaths.loc[df_deaths.Country=="Congo (Kinshasa)",'Country']='Republic of the Congo'
df_deaths.loc[df_deaths.Country=="Cote d'Ivoire",'Country']="Côte d'Ivoire"
df_deaths.loc[df_deaths.Country=="Korea, South",'Country']="Korea, Republic of"
df_deaths.loc[df_deaths.Country=="Syria",'Country']="Syrian Arab Republic"
df_deaths.loc[df_deaths.Country=="Taiwan*",'Country']="Taiwan, Province of China"
df_deaths.loc[df_deaths.Country=="Russia",'Country']='Russian Federation'
df_deaths.loc[df_deaths.Country=="West Bank and Gaza",'Country']='Palestine, State of'
df_deaths.loc[df_deaths.Country=="Venezuela",'Country']='Venezuela, Bolivarian Republic of'
df_deaths.loc[df_deaths.Country=="US",'Country']='United States'
```

As soon as names are unified, I can add their ISO-3 codes.

In [25]:

In [26]:
```
# Transform the dataset in a long format
df_deaths = pd.melt(df_deaths, id_vars=['Country','ISO-3'], value_vars=date_list)
```

Out[26]:

Country	ISO-3	variable	value

In [27]:
```
df_deaths = df_deaths.rename(columns={"variable": "Date","value": "Value"})
```

Out[27]:

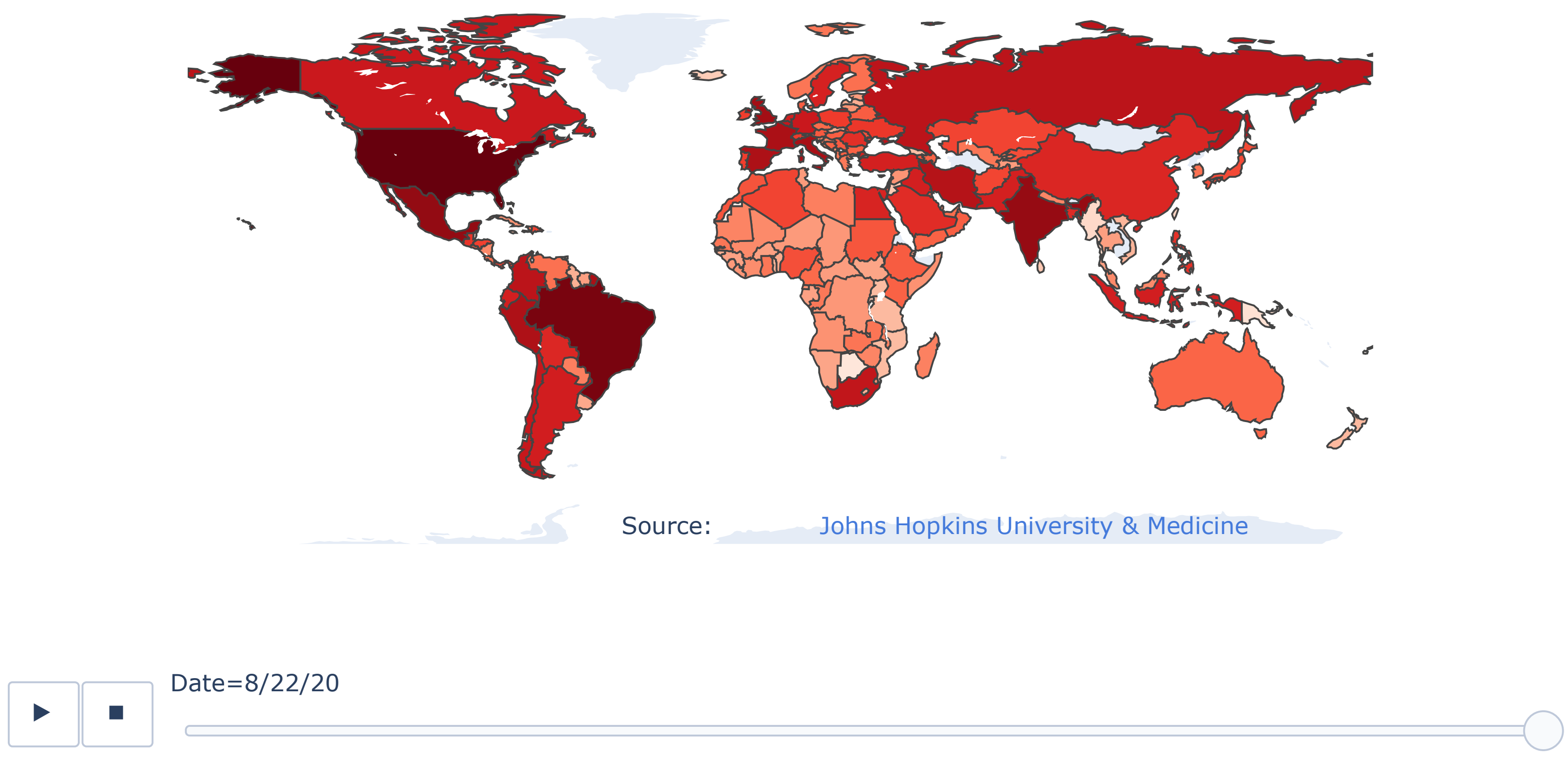
Country	ISO-3	Date	Value

In [28]:
```
#add Kosovo ISO-3 code as it is not listed in pycountry dictionary
df_deaths.loc[df_deaths.Country=="Kosovo",'ISO-3']="XKX"
```

§2.2. Animation of the map over time: deaths

In [104]:
```
import plotly.express as px
df = df_deaths
fig = px.choropleth(df,                                  # input dataframe
                    locationmode='ISO-3',                # set of locations used to map 'locations'
                    locations="ISO-3",                   # identify country by code
                    color=np.log10(df['Value']),         # identify values and replace linear scale with logar
                    hover_name="Country",                # identify column to add as name to hover information
                    animation_frame="Date",              # identify date column
                    projection="equirectangular",        # select projection
                    hover_data=[df['Value']],              # hover text
                    center = {"lat": 14.883333, "lon": 5.266667},# set map center
                    color_continuous_scale=px.colors.sequential.Reds,     # set color scale, "_r" to rever
                    range_color=[0,round(np.log10(df['Value']).max(),2)], # set the range of dataset
                   )

#customize layout
fig.update_layout(
    title_text='Deaths from COVID-19 by country over time<br>January 22, 2020 - August 21, 2020',
    geo=dict(showframe=False, showcoastlines=False, projection_type='equirectangular'),

    annotations = [dict(
        x=0.8,
        y=0.0,
        xref='paper',
        yref='paper',
        text='Source: <a href="https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covic
            Johns Hopkins University & Medicine</a>',
        showarrow = False
    )],

    #customize colorbar
    coloraxis_colorbar=dict(
                        title='Deaths',
                        tickvals=[0, 1, 2, 3, 4, 5, 5.3010299957], #customize colorbar title and ticks
                        ticktext = ['1', '10', '100', '1K', '10K', '100K', '200K'] #replace log10 colc
                        )
                    )
fig.show()
fig.write_html("Deaths_map.html")
```

Deaths from COVID-19 by country over time
January 22, 2020 - August 21, 2020



As of August 2020 maximum number of lethal end cases were registered in USA. Mongolia has not reported COVID-19 deaths.

In [30]:

```
Max number of deaths: 176353
```

In [31]:
```
df_deaths = df_deaths.merge(df_convert,on='ISO-3')
```
Out[31]:

	Country	ISO-3	Date	Value	Region	IncomeLevel	Country_WB
0	Afghanistan	AFG	1/22/20	0	South Asia	Low income	Afghanistan

In [32]:

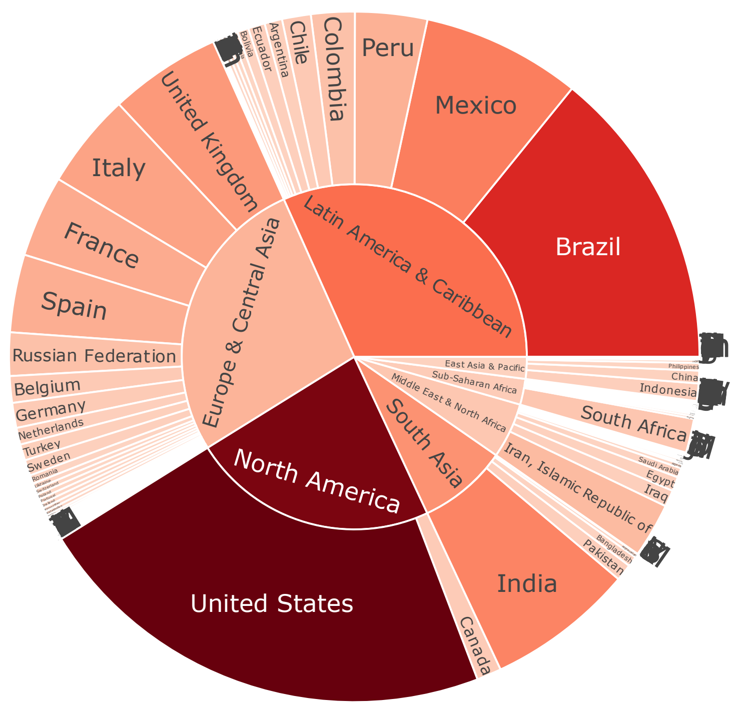
Out[32]:
```
Country        False
ISO-3          False
Date           False
Value          False
Region         False
IncomeLevel    False
Country_WB     False
dtype: bool
```

§2.3. Structure by region and income level: deaths

Interactive sunburst graph shows countries within world's regions where the lethal end cases of COVID-19 were reported by August 20, 2020.

In [33]:
```
import numpy as np
import plotly.express as px

# filter the rows for the last day of observation so cumulative values would be maximum and latest
df = df_deaths[df_deaths['Date'] == '8/20/20']

#exclude rows with zero values of 12 countries with no deaths reported
df = df[df['Value'] != 0]

fig = px.sunburst(df, path = ['Region', 'Country'], values = df.Value,
                  color = df.Value, color_continuous_scale='Reds',
                  title = 'Deaths from COVID-19 by regions and countries<br>by August 20, 2020',
                  color_continuous_midpoint=round(np.average(df.Value, weights = df.Value),1))
fig.show()
```

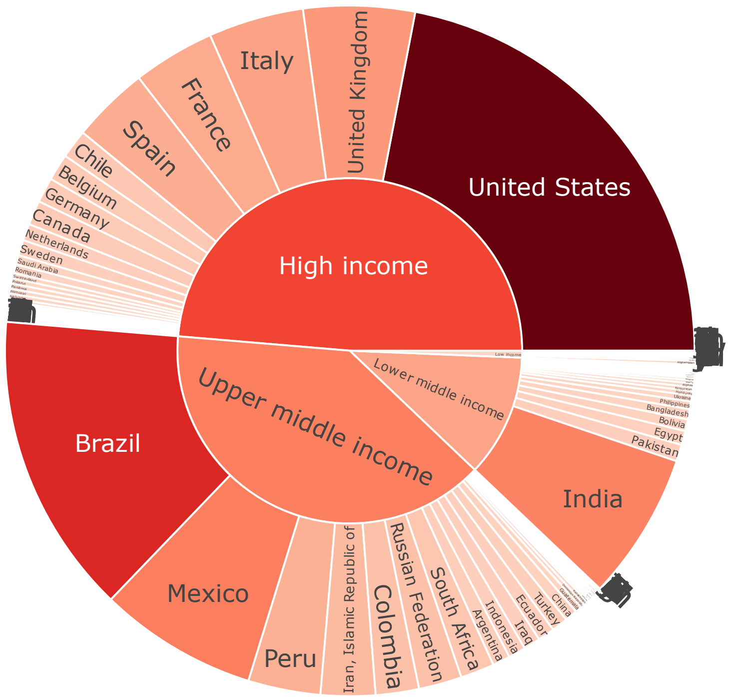
Deaths from COVID-19 by regions and countries
by August 20, 2020



Is there a pattern in terms of virus spread between different regions of income? The sunburst graph shows that 'High income' countries and 'Upper medium income' countries cover 41% and 39% of total COVID-19 cases respectively; Lower middle income countries cover less than 19.5% of total COVID-19 cases; virus spread in low income countries are not that significant yet, or maybe it just was not diagnosed properly.

In [34]:
```
# filter the rows for the last day of observation so cumulative values would be maximum and latest
df = df_deaths[df_deaths['Date'] == '8/20/20']

#exclude rows with zero values of 12 countries with no reported deaths
df = df[df['Value'] != 0]

fig = px.sunburst(df, path=['IncomeLevel', 'Country'], values=df.Value,
                  color = df.Value, color_continuous_scale='Reds',
                  title = 'Deaths from COVID-19 by income level and country<br>by August 20, 2020',
                  color_continuous_midpoint=round(np.average(df.Value, weights = df.Value),1))
fig.show()
```

Deaths from COVID-19 by income level and country
by August 20, 2020



3. Recovered patients

§3.1. Cleaning and preparing for animation: recovery

In [35]:
```
import pandas as pd
import pycountry

df_Recovered=pd.read_csv("https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_d
# Aggregate the dataset
df_Recovered = df_Recovered.drop(columns=['Province/State','Lat','Long'])
df_Recovered = df_Recovered.groupby('Country/Region').agg('sum')
date_list = list(df_Recovered.columns)
df_Recovered['Country'] = df_Recovered.index
```

Several countries' names are written differently than pycountry expects, so I change their names to match and get the code

In [36]:
```
df_Recovered.loc[df_Recovered.Country=="Burma",'Country']='Myanmar'
df_Recovered.loc[df_Recovered.Country=="Brunei",'Country']='Brunei Darussalam'
df_Recovered.loc[df_Recovered.Country=="Iran",'Country']='Iran, Islamic Republic of'
df_Recovered.loc[df_Recovered.Country=="Congo (Brazzaville)",'Country']='Congo, The Democratic Republic of
df_Recovered.loc[df_Recovered.Country=="Congo (Kinshasa)",'Country']='Republic of the Congo'
df_Recovered.loc[df_Recovered.Country=="Cote d'Ivoire",'Country']="Côte d'Ivoire"
df_Recovered.loc[df_Recovered.Country=="Korea, South",'Country']="Korea, Republic of"
df_Recovered.loc[df_Recovered.Country=="Syria",'Country']="Syrian Arab Republic"
df_Recovered.loc[df_Recovered.Country=="Taiwan*",'Country']="Taiwan, Province of China"
df_Recovered.loc[df_Recovered.Country=="Russia",'Country']='Russian Federation'
df_Recovered.loc[df_Recovered.Country=="West Bank and Gaza",'Country']='Palestine, State of'
df_Recovered.loc[df_Recovered.Country=="Venezuela",'Country']='Venezuela, Bolivarian Republic of'
df_Recovered.loc[df_Recovered.Country=="US",'Country']='United States'
```

As soon as names are unified, I can add their ISO-3 codes.

In [37]:

In [38]:
```
# View data structure
```

Out[38]:

	1/22/20	1/23/20	1/24/20	1/25/20	1/26/20	1/27/20	1/28/20	1/29/20	1/30/20	1/31/20	...	8/15/20	8/16/20	8/17/20	8/18/20	8/19/20
Country/Region																
Afghanistan	0	0	0	0	0	0	0	0	0	0	...	27166	27166	27166	27166	27166
Albania	0	0	0	0	0	0	0	0	0	0	...	3746	3794	3816	3871	3928

2 rows × 216 columns

In [39]:
```
# Transform the dataset in a long format
df_Recovered = pd.melt(df_Recovered, id_vars=['Country','ISO-3'], value_vars=date_list)
df_Recovered.head(0)
```

Out[39]:

Country	ISO-3	variable	value

In [40]:

In [41]:
```
#add Kosovo ISO-3 code as it is not listed in pycountry dictionary
df_Recovered.loc[df_Recovered.Country=="Kosovo",'ISO-3']="XKX"
```

In [42]:
```
# View data shape
```
Out[42]: (39590, 4)

As of August 20, 2020 maximum number of recovered patients in one country were registered in Brazil.

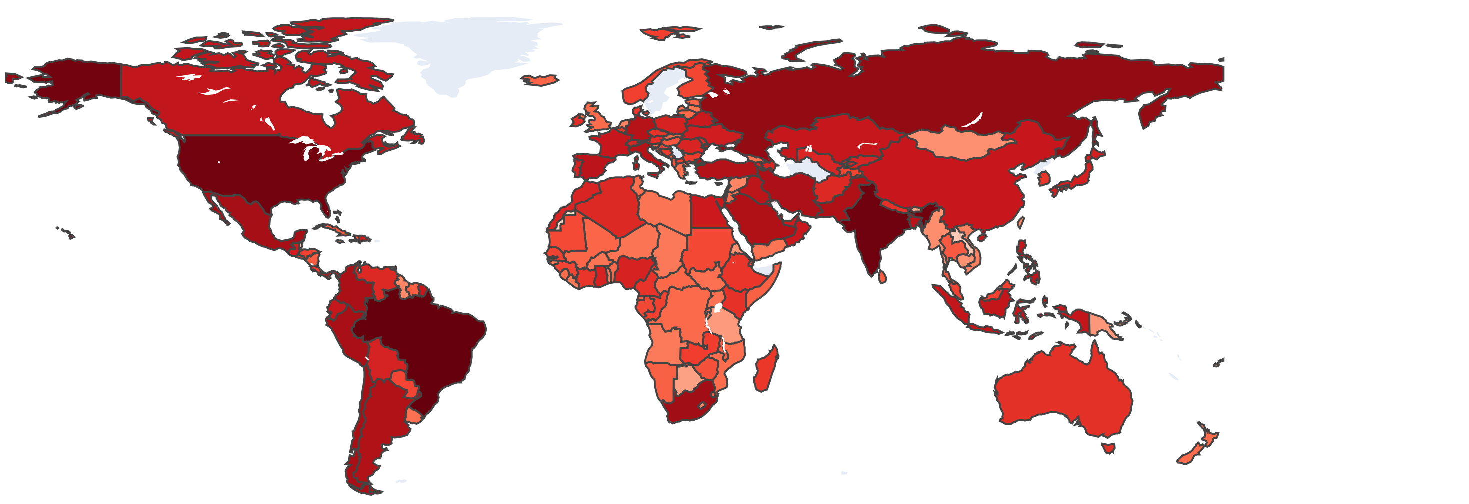
In [43]:

```
Max number of recovered: 2913966
```

§3.2. Animation of the map over time: recovery

In [105]:
```
import plotly.express as px
df = df_Recovered
fig = px.choropleth(df,                                # input dataframe
                    locationmode='ISO-3',              # set of locations used to map 'locations'
                    locations="ISO-3",                 # identify country by code
                    color=np.log10(df['Value']),       # identify values and replace linear scale with logari
                    hover_name="Country",              # identify column to add as name to hover information
                    animation_frame="Date",            # identify date column
                    projection="equirectangular",      # select projection
                    hover_data=[df['Value']],           # hover text
                    center = {"lat": 14.883333, "lon": 5.266667},# set map center
                    color_continuous_scale=px.colors.sequential.Reds,      # set color scale, "_r" to rever
                    range_color=[0,round(np.log10(df['Value']).max(),2)], # set the range of dataset
                   )

#customize layout
fig.update_layout(
    title_text='Recovered from COVID-19 by country over time<br>January 22, 2020 - August 21, 2020',
    geo=dict(showframe=False, showcoastlines=False, projection_type='equirectangular'),

    annotations = [dict(
        x=0.8,
        y=0.0,
        xref='paper',
        yref='paper',
        text='Source: <a href="https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covic
            Johns Hopkins University & Medicine</a>',
        showarrow = False
    )],

    #customize colorbar
    coloraxis_colorbar=dict(
                        title='Recovered',
                        tickvals=[0,1,2, 3, 4, 5, 6, 6.5], #customize colorbar title and ticks values
                        ticktext = ['1','10','100', '1K', '10K', '100K', '1M', '3M'] #replace log10 cc
                        )
                    )
fig.show()
fig.write_html("Recovered_map.html")
```

Recovered from COVID-19 by country over time
January 22, 2020 - August 21, 2020



Source: Johns Hopkins University & Medicine

Interactive map shows where are located the most number of recovery cases. It is Brazil (2.6 mln), India (1.9 mln), United States (1.8 mln), Russia (0.7 mln) and South Africa (0.5 mln).

In [45]:

```
df_Recovered = df_Recovered.merge(df_convert,on='ISO-3')
```

Out[45]:

	Country	ISO-3	Date	Value	Region	IncomeLevel	Country_WB
0	Afghanistan	AFG	1/22/20	0	South Asia	Low income	Afghanistan

In [46]:

Out[46]:

```
Country        False
ISO-3          False
Date           False
Value          False
Region         False
IncomeLevel    False
Country_WB     False
dtype: bool
```

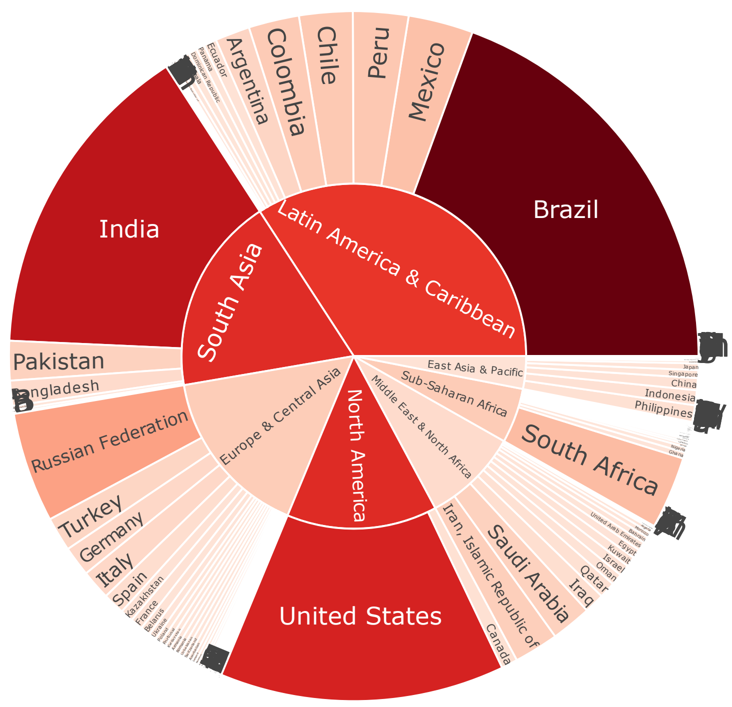
§3.3. Structure by region and income level: recovery

Interactive sunburst graph shows countries within world's regions where the recovery cases of COVID-19 were reported by August 21, 2020.

In [47]:

```
import numpy as np
import plotly.express as px

# filter the rows for the last day of observation so cumulative values would be maximum and latest
df = df_Recovered[df_Recovered['Date'] == '8/21/20']

#exclude rows with zero values of 12 countries with no deaths reported
df = df[df['Value'] != 0]

fig = px.sunburst(df, path = ['Region', 'Country'], values = df.Value,
                  color = df.Value, color_continuous_scale='Reds',
                  title = 'Recovered from COVID-19 by regions and countries<br>by August 21, 2020',
                  color_continuous_midpoint=round(np.average(df.Value, weights = df.Value),1))
fig.show()
```

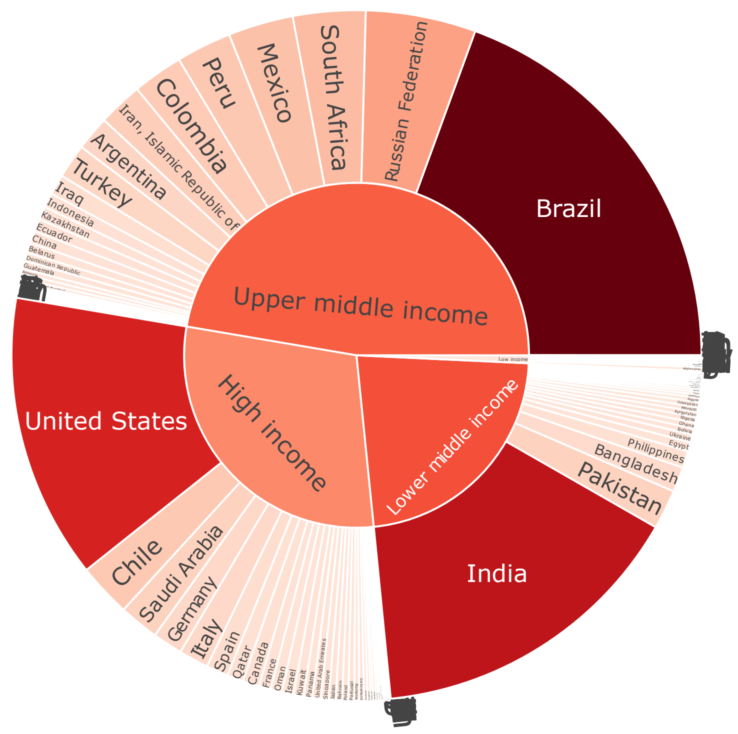
Recovered from COVID-19 by regions and countries
by August 21, 2020



Is there a pattern in terms of virus spread between different regions of income? The sunburst graph shows that 'High income' countries and 'Upper medium income' countries cover 41% and 39% of total COVID-19 cases respectively; Lower middle income countries cover less than 19.5% of total COVID-19 cases; virus spread in low income countries are not that significant yet, or maybe it just was not diagnosed properly.

In [48]:
```python
# filter the rows for the last day of observation so cumulative values would be maximum and latest
df = df_Recovered[df_Recovered['Date'] == '8/21/20']

#exclude rows with zero values of 12 countries with no reported deaths
df = df[df['Value'] != 0]

fig = px.sunburst(df, path=['IncomeLevel', 'Country'], values=df.Value,
                  color = df.Value, color_continuous_scale='Reds',
                  title = 'Recovered from COVID-19 by income level and country<br>by August 21, 2020',
                  color_continuous_midpoint=round(np.average(df.Value, weights = df.Value),1))
fig.show()
```

Recovered from COVID-19 by income level and country
by August 21, 2020



4. Calculating additional data

§4.1. Importing, cleaning and calculating

In order to make meaningful comparison of deaths from virus between the countries I need to import population data first, and calculate deaths per million ratio.

In [49]:
```python
import pandas as pd #to work with tabular data
import requests #to access the csv from the url string
import io #to read the csv directly from the url string

url_pop = "https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/UID_ISO_FI
raw=requests.get(url_pop).content
df_pop=pd.read_csv(io.StringIO(raw.decode('utf-8')))

# View data shape
df_pop.shape
```

Out[49]: (4153, 12)

I expected to see about 200 rows in the dataset (1 row per country), but its shape contains 4153 rows. The reason for that is, as we can see below, that it contains not only population of the countries, but mostly population of provinces, states and cities for some countries.

In [50]:
```python
# View a slice of loaded data
```

Out[50]:

	UID	iso2	iso3	code3	FIPS	Admin2	Province_State	Country_Region	Lat	Long_	Combined_Key	Population
78	15217	CL	CHL	152.0	NaN	NaN	Unknown	Chile	NaN	NaN	Unknown, Chile	NaN
79	170	CO	COL	170.0	NaN	NaN	NaN	Colombia	4.5709	-74.2973	Colombia	50882884.0
80	17001	CO	COL	170.0	NaN	NaN	Amazonas	Colombia	-1.4429	-71.5724	Amazonas, Colombia	76589.0
81	17002	CO	COL	170.0	NaN	NaN	Antioquia	Colombia	7.1986	-75.3412	Antioquia, Colombia	6407102.0
82	17003	CO	COL	170.0	NaN	NaN	Arauca	Colombia	7.0762	-70.7105	Arauca, Colombia	262174.0
83	17004	CO	COL	170.0	NaN	NaN	Atlantico	Colombia	10.6966	-74.8741	Atlantico, Colombia	2535517.0
84	17005	CO	COL	170.0	NaN	NaN	Bolivar	Colombia	8.6704	-74.0300	Bolivar, Colombia	2070110.0

	UID	iso2	iso3	code3	FIPS	Admin2	Province_State	Country_Region	Lat	Long_	Combined_Key	Population
85	17006	CO	COL	170.0	NaN	NaN	Boyaca	Colombia	5.4545	-73.3620	Boyaca, Colombia	1217376.0
86	17007	CO	COL	170.0	NaN	NaN	Caldas	Colombia	5.2983	-75.2479	Caldas, Colombia	998255.0
87	17008	CO	COL	170.0	NaN	NaN	Capital District	Colombia	4.7110	-74.0721	Capital District, Colombia	7412566.0
88	17009	CO	COL	170.0	NaN	NaN	Caqueta	Colombia	0.8699	-73.8419	Caqueta, Colombia	401489.0
89	17010	CO	COL	170.0	NaN	NaN	Casanare	Colombia	5.7589	-71.5724	Casanare, Colombia	420504.0
90	17011	CO	COL	170.0	NaN	NaN	Cauca	Colombia	2.7050	-76.8260	Cauca, Colombia	1464488.0
91	17012	CO	COL	170.0	NaN	NaN	Cesar	Colombia	9.3373	-73.6536	Cesar, Colombia	1200574.0
92	17013	CO	COL	170.0	NaN	NaN	Choco	Colombia	5.2528	-76.8260	Choco, Colombia	534826.0
93	17014	CO	COL	170.0	NaN	NaN	Cordoba	Colombia	8.0493	-75.5740	Cordoba, Colombia	1784783.0
94	17015	CO	COL	170.0	NaN	NaN	Cundinamarca	Colombia	5.0260	-74.0300	Cundinamarca, Colombia	2919060.0
95	17016	CO	COL	170.0	NaN	NaN	Guainia	Colombia	2.5854	-68.5247	Guainia, Colombia	48114.0
96	17017	CO	COL	170.0	NaN	NaN	Guaviare	Colombia	1.0654	-73.2603	Guaviare, Colombia	82767.0
97	17018	CO	COL	170.0	NaN	NaN	Huila	Colombia	2.5359	-75.5277	Huila, Colombia	1100386.0
98	17019	CO	COL	170.0	NaN	NaN	La Guajira	Colombia	11.3548	-72.5205	La Guajira, Colombia	880560.0

To avoid double counting I need to clean the dataset from excessive information. To do that, I filter only the rows with values NaN in column "Province_State" using the isnull function. After that I can simply drop "UID", "iso2", "code3", "FIPS", "Admin2", "Province_State", "Combined_Key" columns.

In [51]:
```
df_pop = df_pop[pd.isnull(df_pop.Province_State)]
df_pop = df_pop.drop("UID", axis=1)
df_pop = df_pop.drop("iso2", axis=1)
df_pop = df_pop.drop("FIPS", axis=1)
df_pop = df_pop.drop("Admin2", axis=1)
df_pop = df_pop.drop("Province_State", axis=1)
df_pop = df_pop.drop("Combined_Key", axis=1)
```

Now the dataset contains only population of 188 countries, and 5 columns instead of 12.

In [52]:

Out[52]: (188, 5)

There were 2 empty cells in population column of the dataframe, they contained data on "MS Zaandam" and "Diamond Princess".

In [53]:
```
df_pop = df_pop[df_pop.Country_Region !='MS Zaandam']
df_pop = df_pop[df_pop.Country_Region !='Diamond Princess']
```

Out[53]: (186, 5)

In [54]:

If I check maximum population, I see population of China.

In [55]:

Out[55]: 1404676330.0

Minimum population is in Vatican ("Holy See").

In [56]:

Out[56]: 809.0

Vatican population is too small to be visible on the map or graph, so it is better to get rid of it in dataset.

In [57]:
```
df_pop = df_pop[df_pop.Country_Region !='Holy See']
```

Out[57]: (185, 5)

Now the dataset has the same number of countries, as the datasets of COVID-19 statistics. In order to use population data for calculation, I unify the "iso3"/"ISO-3" names of columns and merge "Population" column to df_deaths dataframe.

In [58]:

In [59]:

Out[59]:

	Country	ISO-3	Date	Value	Region	IncomeLevel	Country_WB
0	Afghanistan	AFG	1/22/20	0	South Asia	Low income	Afghanistan

Both dataframes based on the same list of 185 countries.

In [60]:

Out[60]: True

In [61]:

In [62]:

In [65]:

```
df_deaths_pop["Death per mln"] = round((df_deaths_pop.Value / df_deaths_pop.Population * 1000000),2)
```

Out[65]:

	Country	ISO-3	Date	Value	Region	IncomeLevel	Country_WB	Population	Death_per_mln	Death per mln
39371	Zimbabwe	ZWE	8/18/20	141	Sub-Saharan Africa	Lower middle income	Zimbabwe	14862927.0	9.49	9.49
39372	Zimbabwe	ZWE	8/19/20	150	Sub-Saharan Africa	Lower middle income	Zimbabwe	14862927.0	10.09	10.09
39373	Zimbabwe	ZWE	8/20/20	151	Sub-Saharan Africa	Lower middle income	Zimbabwe	14862927.0	10.16	10.16
39374	Zimbabwe	ZWE	8/21/20	152	Sub-Saharan Africa	Lower middle income	Zimbabwe	14862927.0	10.23	10.23
39375	Zimbabwe	ZWE	8/22/20	153	Sub-Saharan Africa	Lower middle income	Zimbabwe	14862927.0	10.29	10.29

In [66]:

```
print(df_deaths_pop["Death per mln"].min())
print(df_deaths_pop["Death per mln"].max())
```

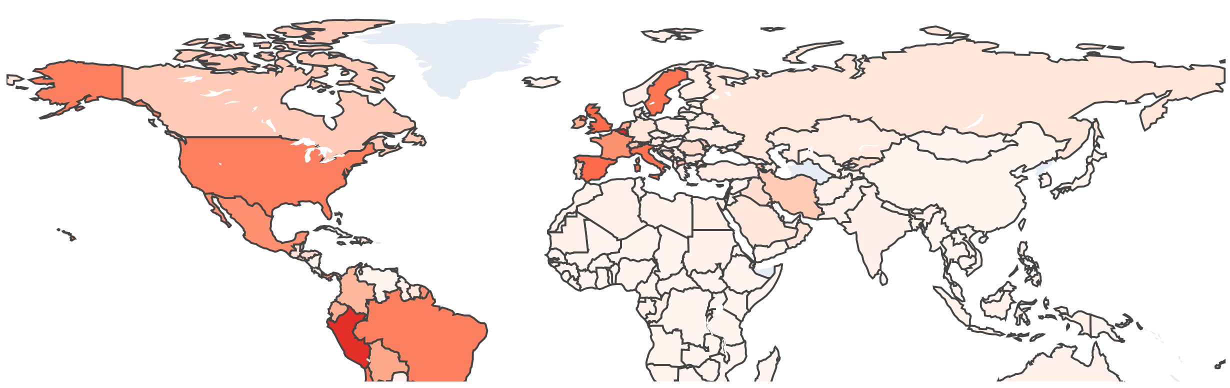
```
0.0
1237.55
40.3
```

§2.2. Animation of the map over time: deaths per million

In [67]:

```
import plotly.express as px
df = df_deaths_pop
fig = px.choropleth(df,                                 # input dataframe
                    locationmode='ISO-3',               # set of locations used to map 'locations'
                    locations="ISO-3",                  # identify country by code
                    color=df['Death per mln'],         # identify values
                    hover_name="Country",               # identify column to add as name to hover information
                    animation_frame="Date",             # identify date column
                    projection="equirectangular",       # select projection
                    hover_data=[df['Death per mln']],               # hover text
                    center = {"lat": 14.883333, "lon": 5.266667},# set map center
                    color_continuous_scale=px.colors.sequential.Reds,    # set color scale, "_r" to reverse
                    range_color=[0,round(df["Death per mln"].max(),0)] # set the range of dataset
                    )

#customize layout
fig.update_layout(
    title_text='Deaths from COVID-19 per million people by country over time<br>January 22, 2020 - August
    geo=dict(showframe=False, showcoastlines=False, projection_type='equirectangular'),

    annotations = [dict(
        x=0.8,
        y=0.0,
        xref='paper',
        yref='paper',
        text='Source: <a href="https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covic
            Johns Hopkins University & Medicine</a>',
        showarrow = False
    )],

    #customize colorbar
    coloraxis_colorbar=dict(title='Deaths<br>per M')
                    )
fig.show()
fig.write_html("Deaths_per_mln_map.html")
```

Deaths from COVID-19 per million people by country over time
January 22, 2020 - August 21, 2020



Dataset 2: Oxford Covid-19 Government Response Tracker

This part of the project is based on ideas "Variation in governmentresponses to COVID-19" paper by Oxford:

"As governments continue to respond to COVID-19, it is imperative to study what measures are effective and which are not. While the data presented here do, of course, not measure effectiveness directly, they can be useful input to studies that analyse factors affecting disease progression. OxCGRT seeks to contribute to this knowledge gap by providing comparable measures of individual policy actions, as well as several comparable aggregate indices. We find significant variation in both the measures that governments adopt and when they adopt them. Going forward, governments will benefit from adopting an evidence-based approach to the measures they deploy."

Paper: [https://www.bsg.ox.ac.uk/sites/default/files/2020-05/BSG-WP-2020-032-v6.0.pdf (https://www.bsg.ox.ac.uk/sites/default/files/2020-05/BSG-WP-2020-032-v6.0.pdf)](https://www.bsg.ox.ac.uk/sites/default/files/2020-05/BSG-WP-2020-032-v6.0.pdf)

Data: [https://github.com/OxCGRT/covid-policy-tracker (https://github.com/OxCGRT/covid-policy-tracker)](https://github.com/OxCGRT/covid-policy-tracker)

5. Government response index

§5.1. Cleaning and preparing for plotting

The Oxford Covid-19 Government Response Tracker (OxCGRT) collects systematic information on which governments have taken which measures, and when. The data is daily published at project's GitHub page [https://github.com/OxCGRT/covid-policy-tracker (https://github.com/OxCGRT/covid-policy-tracker)](https://github.com/OxCGRT/covid-policy-tracker)

In [68]:

```
import requests #to access the csv from the url string
import io #to read the csv directly from the url string
import pandas as pd #to work with tabular data
import pycountry #to get the three-letter country codes ISO 3166-1 for each country

url_OxCGRT = "https://raw.githubusercontent.com/OxCGRT/covid-policy-tracker/master/data/OxCGRT_latest.csv'
raw=requests.get(url_OxCGRT).content
df_OxCGRT=pd.read_csv(io.StringIO(raw.decode('utf-8')))
```

Out[68]:

```
(43660, 42)
```

In [69]:

```
print("Minimum date in YYYYMMDD format is", df_OxCGRT.Date.min())
```

```
Minimum date in YYYYMMDD format is 20200101
Maximum date in YYYYMMDD format is 20200823
```

In [70]:

Out[70]:

	CountryName	CountryCode	Date	C1_School closing	C1_Flag	C2_Workplace closing	C2_Flag	C3_Cancel public events	C3_Flag	C4_Restrictions on gatherings	...	StringencyInd
181	Aruba	ABW	20200630	0.0	NaN	1.0	1.0	0.0	NaN	0.0	...	32.
417	Afghanistan	AFG	20200630	3.0	1.0	3.0	0.0	2.0	1.0	4.0	...	78.
653	Angola	AGO	20200630	3.0	1.0	2.0	0.0	2.0	1.0	3.0	...	75.

3 rows × 42 columns

In [71]:

```
# drop excessive columns
df_OxCGRT_indexes = df_OxCGRT.drop(axis=1, columns = ['C1_School closing', 'C1_Flag',
                                                      'C2_Workplace closing', 'C2_Flag',
                                                      'C3_Cancel public events', 'C3_Flag',
                                                      'C4_Restrictions on gatherings', 'C4_Flag',
                                                      'C5_Close public transport', 'C5_Flag',
                                                      'C6_Stay at home requirements', 'C6_Flag',
                                                      'C7_Restrictions on internal movement', 'C7_
                                                      'C8_International travel controls',
                                                      'E1_Income support', 'E1_Flag',
                                                      'E2_Debt/contract relief', 'E3_Fiscal measu
```

```
                                                          'E4_International support',
                                                          'H1_Public information campaigns', 'H1_Flag'
                                                          'H2_Testing policy', 'H3_Contact tracing',
                                                          'H4_Emergency investment in healthcare',
                                                          'H5_Investment in vaccines',
                                                          'M1_Wildcard',
                                                          'StringencyIndex',
                                                          'StringencyLegacyIndex',
                                                          'StringencyLegacyIndexForDisplay',
                                                          'GovernmentResponseIndex',
                                                          'ContainmentHealthIndex',
                                                          'EconomicSupportIndex'
                                                         ])
df_OxCGRT_indexes[df_OxCGRT_indexes.Date == 20200820]
```

Out[71]:

	CountryName	CountryCode	Date	ConfirmedCases	ConfirmedDeaths	StringencyIndexForDisplay	GovernmentResponseIndexForDisplay
232	Aruba	ABW	20200820	1296.0	5.0	47.22	51.28
468	Afghanistan	AFG	20200820	37759.0	1383.0	62.04	49.36
704	Angola	AGO	20200820	1966.0	90.0	79.17	67.63
940	Albania	ALB	20200820	7812.0	234.0	53.70	55.13
1176	Andorra	AND	20200820	1024.0	53.0	44.44	58.97
...	...	...	...	...	...	...	...
42712	Kosovo	RKS	20200820	11545.0	390.0	69.44	64.10
42948	Anguilla	AIA	20200820	3.0	0.0	26.85	36.54
43184	Falkland Islands	FLK	20200820	13.0	0.0	27.78	39.74
43420	Montserrat	MSR	20200820	NaN	NaN	64.81	65.38
43656	Pitcairn Islands	PCN	20200820	NaN	NaN	11.11	NaN

185 rows × 9 columns

The Oxford Covid-19 Government Response Tracker tracks individual policy measures across 17 indicators and calculate several indices to give an overall impression of government activity.

I am particularly interested in 2 aggregated indexes calculated by Oxford:

- GovernmentResponseIndexForDisplay (all 17 indicators)
- EconomicSupportIndexForDisplay (2 indicators)

Each of these indices report a number between 0 to 100 that reflects the level of the governments response along certain dimensions. This is a measure of how many of the relevant indicators a government has acted upon, and to what degree. The index cannot say whether a government's policy has been implemented effectively.

Each index dataframe could be downloaded as separate .csv file from [https://github.com/OxCGRT/covid-policy-tracker/tree/master/data/timeseries (https://github.com/OxCGRT/covid-policy-tracker/tree/master/data/timeseries)](https://github.com/OxCGRT/covid-policy-tracker/tree/master/data/timeseries)

In [72]:

```
df_iGovResp=pd.read_csv("index_governmentresponse.csv")
```

172

I will plot the data of United States, Canada, China, Brazil and Russia, and compare those countries' indexes to each other and world's average (calculated as mean of 172 countries available in this dataset). First, identify parts of dataframe I need to plot:

In [73]:

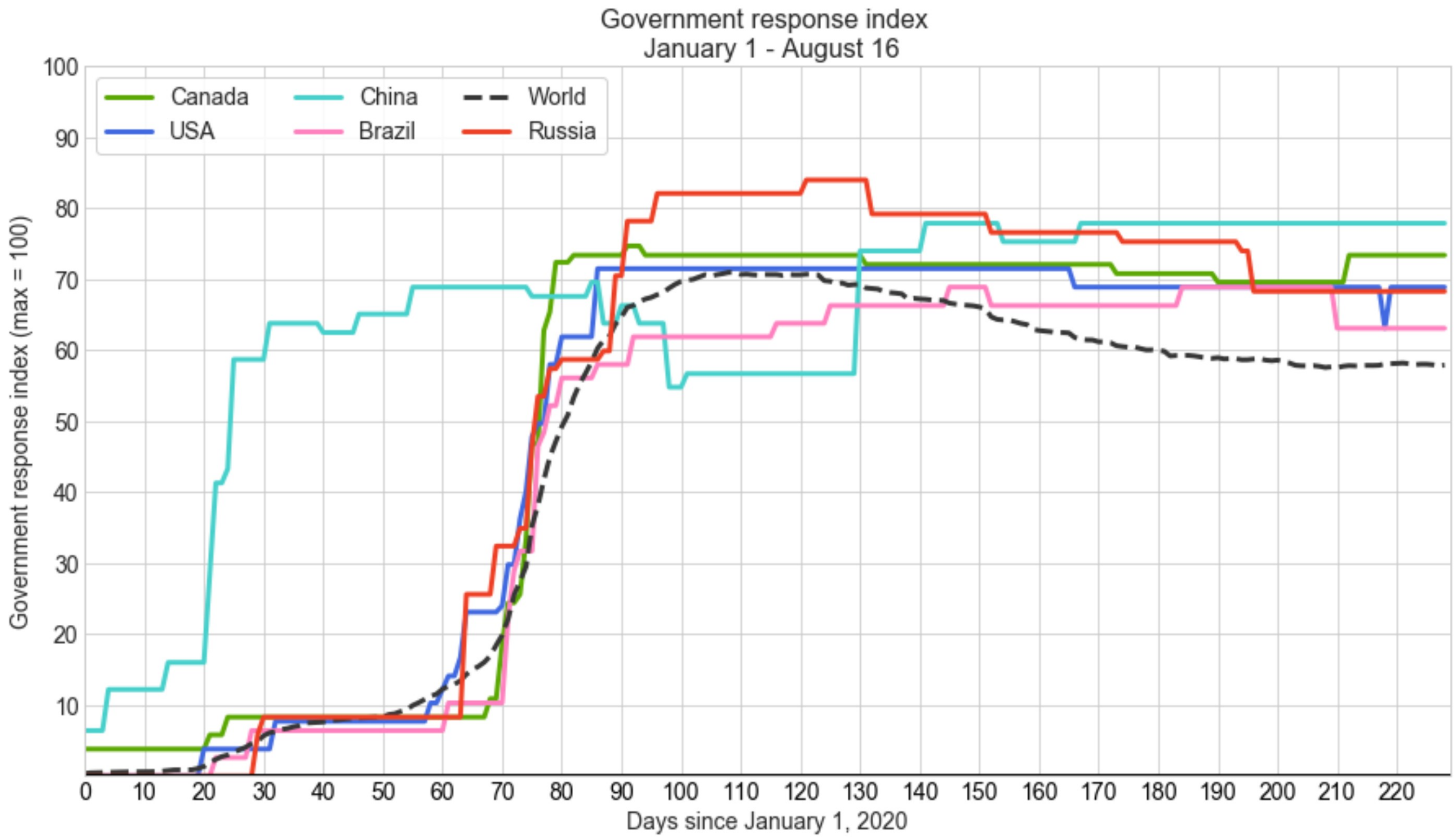
```
ca_iGovResp = df_iGovResp[df_iGovResp.CountryName == 'Canada'].drop(axis=1, columns = ['CountryCode']) #Ca
ru_iGovResp = df_iGovResp[df_iGovResp.CountryName == 'Russia'].drop(axis=1, columns = ['CountryCode']) #Ru
wld_iGovResp = df_iGovResp.mean(axis=0, skipna=True).drop(axis=1, columns = ['CountryCode']) #"World", 172
cn_iGovResp = df_iGovResp[df_iGovResp.CountryName == 'China'].drop(axis=1, columns = ['CountryCode']) #Ch
br_iGovResp = df_iGovResp[df_iGovResp.CountryName == 'Brazil'].drop(axis=1, columns = ['CountryCode']) #B
us_iGovResp = df_iGovResp[df_iGovResp.CountryName == 'United States'].drop(axis=1, columns = ['CountryCode
```

In [74]:

```
import matplotlib.pyplot as plt
import matplotlib.style as style
import numpy as np
import seaborn as sns
%matplotlib inline

SMALL_SIZE = 12
MEDIUM_SIZE = 14
BIGGER_SIZE = 16

plt.rc('font', size=MEDIUM_SIZE)          # controls default text sizes
plt.rc('axes', titlesize=BIGGER_SIZE)     # fontsize of the axes title
plt.rc('axes', labelsize=MEDIUM_SIZE)     # fontsize of the x and y labels
plt.rc('xtick', labelsize=MEDIUM_SIZE)    # fontsize of the tick labels
plt.rc('ytick', labelsize=MEDIUM_SIZE)    # fontsize of the tick labels
plt.rc('legend', fontsize=MEDIUM_SIZE)    # legend fontsize
plt.rc('figure', titlesize=BIGGER_SIZE)   # fontsize of the figure title
```

In [75]:

```
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation

plt.figure(figsize=(15,8))
plt.style.use('seaborn-whitegrid')

plt.title('Government response index \n January 1 - August 16')
plt.axhline(y = 0, color = 'black', linewidth = 1.3, alpha = .7)
plt.tick_params(axis = 'both', which = 'major')

x = np.linspace(0, 228, 229) # x axis start point, end point and number of intervals
xlim = (0, 229) # y axis start point, end point
y_pos = np.arange(len(wld_iGovResp))


# use the plt.xticks function to custom labels
plt.xticks(y_pos, color='black', rotation=False)
plt.xticks(np.arange(0, 230, 10.0))
plt.yticks(np.arange(10, 110, 10))
plt.xlabel('Days since January 1, 2020')
plt.ylabel('Government response index (max = 100)')

y1 = np.array(ca_iGovResp.drop(axis=1, columns = 'CountryName').values.tolist()[0])
y2 = np.array(us_iGovResp.drop(axis=1, columns = 'CountryName').values.tolist()[0])
y3 = np.array(cn_iGovResp.drop(axis=1, columns = 'CountryName').values.tolist()[0])
y4 = np.array(br_iGovResp.drop(axis=1, columns = 'CountryName').values.tolist()[0])
y5 = np.array(wld_iGovResp)
y6 = np.array(ru_iGovResp.drop(axis=1, columns = 'CountryName').values.tolist()[0])

ca = plt.plot(x, y1, label='Canada',c="xkcd:leaf green", lw=3, animated =True)
us = plt.plot(x, y2, label='USA', c="royalblue", lw=3, animated =True)
cn = plt.plot(x, y3, label='China', c="mediumturquoise", lw=3, animated =True)
br = plt.plot(x, y4, label='Brazil', c="xkcd:pink", lw=3, animated =True)
wld = plt.plot(x, y5, label='World', ls='--',c='xkcd:dark grey', lw=3, animated =True)
ru = plt.plot(x, y6, label='Russia', c="xkcd:tomato", lw=3, animated =True)

plt.axis([0, 229, 0, 100])
plt.axhline(0, c='black', ls='-', lw=2)
plt.axvline(0, c='black', ls='-', lw=2)
plt.legend(loc='upper left', frameon=True, fancybox=True, framealpha=0.9, facecolor='white', ncol=3)
```



The graph shows government response index to COVID-19 for period since January 1st till August 16th.

Government of China was the first one that responded to COVID-19, dealing with the outbreak first identified in Wuhan in December 2019. It took actions 50-70 days ahead of the rest of the world. Measures taken were more strict in terms of methods, to prevent more damage from virus. Both developing countries (like Brazil and Russia) and developed countries (United States and Canada) prefered not to make unpopular and costly decisions and not enforce strict measures to prevent virus spread till mid-March.

Brazil government reaction was relatively mild in comparison with both all of the rest plotted countries and world's average. Russian government took more strict measures and keeps that level above world's average since March.

§5.2. Aggregating data for correlation analysis

In [76]:
```
data_all = df_deaths_pop #start aggregating data to single dataframe
data_all = data_all[data_all.Date == '8/21/20'] #filter the latest day to research
print(data_all.shape)
data_all = data_all.rename(columns = {'Value':'Deaths',
                            'IncomeLevel':'Income',
                            'Death_per_mln':'Deaths per mln'})
```

```
(184, 10)
(184, 10)
```

In [77]:
```
#merge Cases to data_all
Cases = df_cases[df_cases['Date']=='8/21/20']
Cases = Cases[Cases['Value']>0]
Cases = Cases.drop(axis=1, columns = ['Country','Date','Region','IncomeLevel','Country_WB'])
Cases = Cases.rename(columns = {'Value':'Cases'})
data_all = data_all.merge(Cases, on = "ISO-3")

#Calculate Cases per mln of population
data_all['Cases per mln'] = data_all.Cases / data_all.Population *1000000
```

```
(184, 12)
```

§5.3. Correlation analysis

Correlation analysis: Cases and Deaths

In [78]:

Out[78]:

	Country	ISO-3	Date	Deaths	Region	Income	Country_WB	Population	Deaths per mln	Death per mln	Cases	Cases per mln
0	Afghanistan	AFG	8/21/20	1385	South Asia	Low income	Afghanistan	38928341.0	35.58	35.58	37894	973.429615

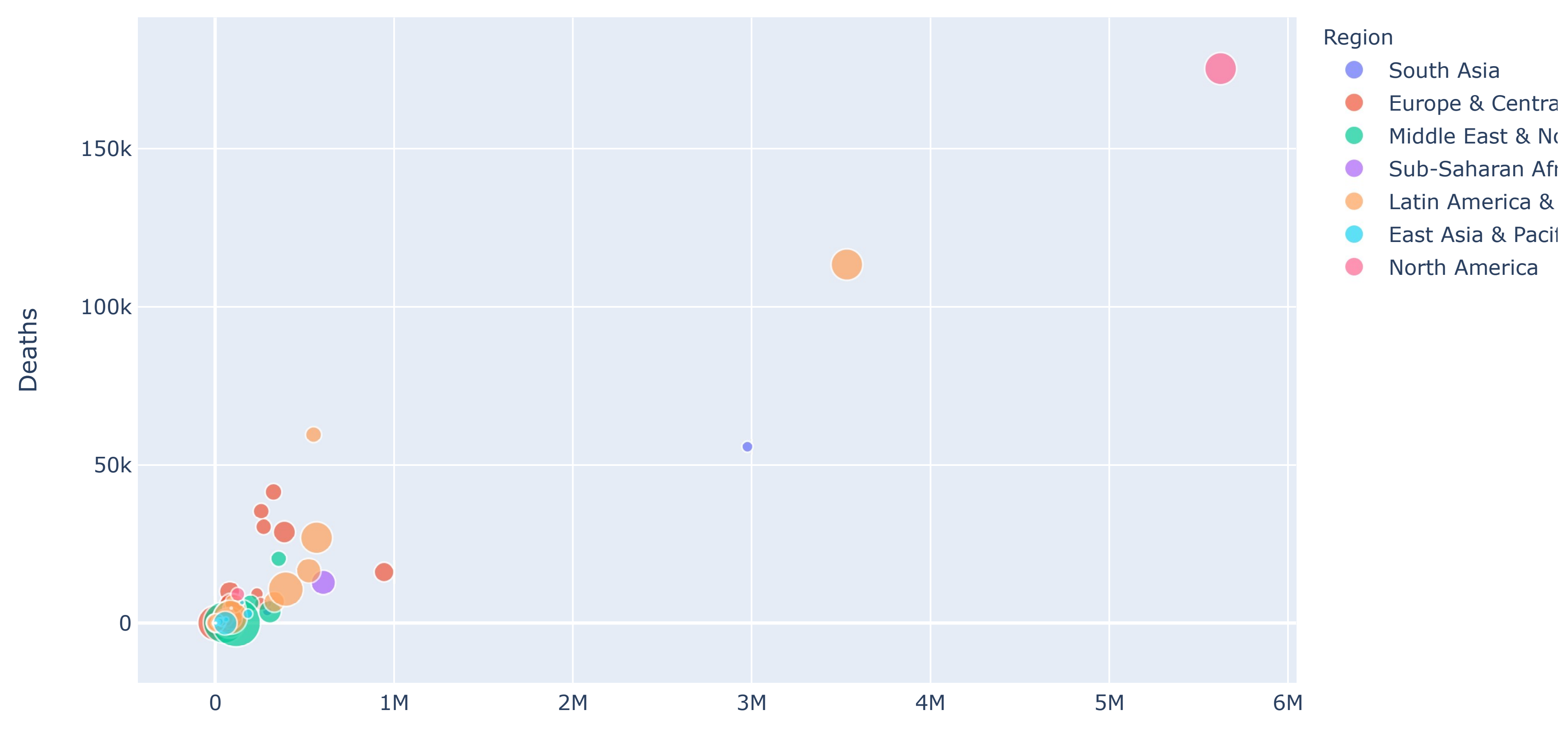
I explore the correlation between columns.

In [79]:
```
print("Q: Do dumulative cases correlate with cumulative deaths?")
print("A: Yes, correlation is positive and strong, standard correlation coefficient is 0.94")
data_all_nonulls = data_all[data_all.Deaths != 0] #filter out 12 countries not reported deaths
r = data_all_nonulls.Deaths.corr(data_all_nonulls.Cases)
```

```
Q: Do dumulative cases correlate with cumulative deaths?
A: Yes, correlation is positive and strong, standard correlation coefficient is 0.94
```

Out[79]: 0.94

In [80]:
```
# Cases vs Deaths plot
import plotly.express as px
df = data_all_nonulls
fig = px.scatter(df, x=df['Cases'], y=df['Deaths'],
                 color='Region', size=round(df['Cases per mln'],1),
                 hover_data=['Country'],
                 labels={'x':'Cases', 'y':'Deaths', 'size':'Cases per mln'}
                )
```

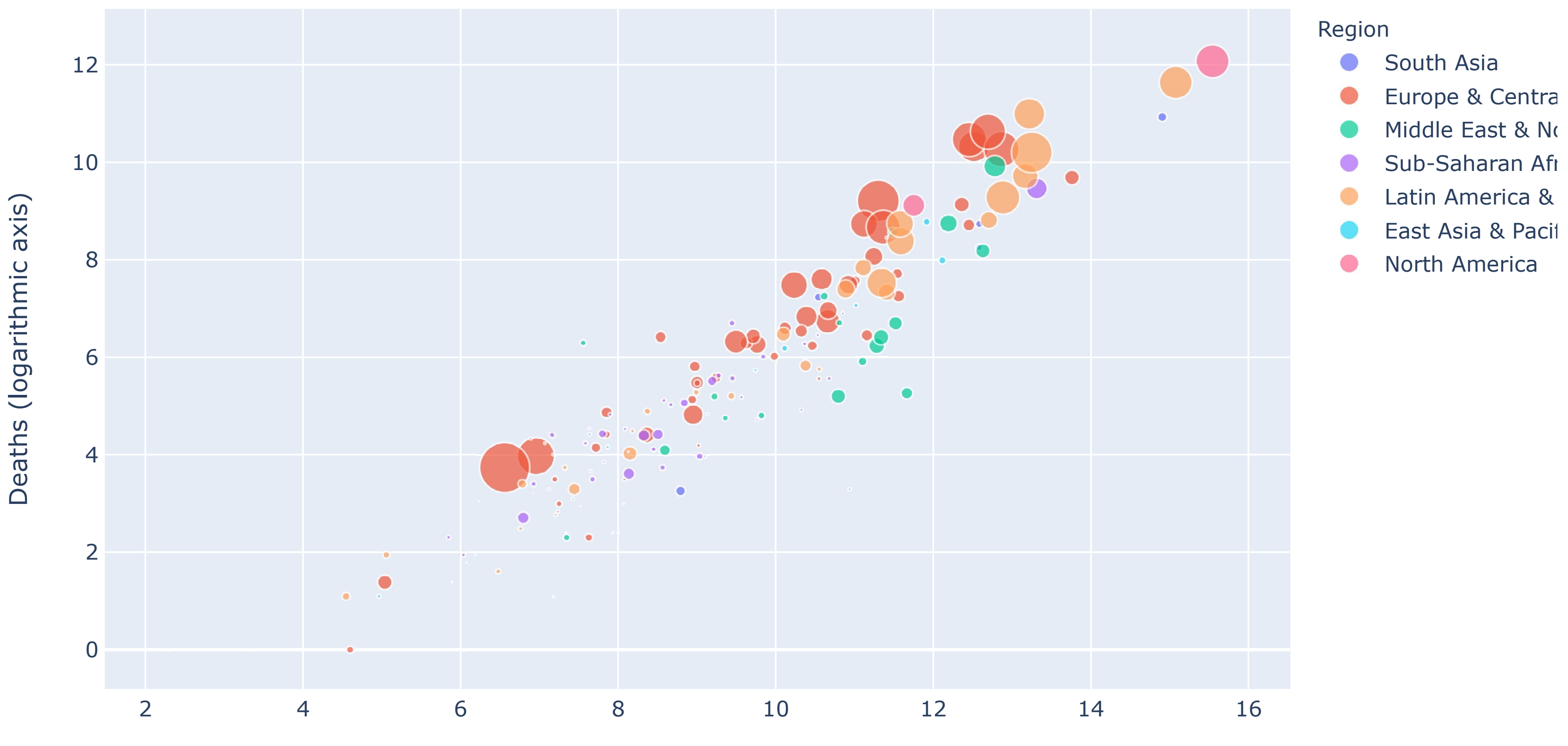


In [81]:
```
print("Q: log(Deaths) correlate with log(Cases)?")
print("A: Yes, correlation is positive and very strong, r = 0.93")
r = (np.log(data_all_nonulls.Deaths).corr(np.log(data_all_nonulls.Cases), method = "pearson"))
```

```
Q: log(Deaths) correlate with log(Cases)?
A: Yes, correlation is positive and very strong, r = 0.93
```

Out[81]: 0.94

In [82]:
```
# log(Cases) correlation with (Deaths) plot
import plotly.express as px
import plotly.io as pio
pio.templates
for template in ["plotly_dark"]:
    df = data_all_nonulls
    fig = px.scatter(df,
                     x=np.log(df['Cases']),
                     y=np.log(df['Deaths']),
                     color='Region',
                     size=df['Deaths per mln'],
                     hover_data=['Country'],
                     labels={'x':'Confirmed cases (logarithmic axis)', 'y':'Deaths (logarithmic axis)'},

                     )
```



In [83]:
```
print("Q: Deaths correlate with Deaths per mln?")
print("A: Yes, but positive correlation is weak and insignificant, r = 0.44")
r = data_all_nonulls.Deaths.corr(data_all_nonulls['Deaths per mln'], method = "pearson")
```

```
Q: Deaths correlate with Deaths per mln?
A: Yes, but positive correlation is weak and insignificant, r = 0.44
```

Out[83]: 0.44

In [84]:
```
print("Q: log(Deaths) correlate with log(Deaths per mln)?")
print("A: Yes, positive correlation is quite strong, r = 0.7")
r = np.log(data_all_nonulls.Deaths).corr(np.log(data_all_nonulls['Deaths per mln']),method = "pearson")
```

```
Q: log(Deaths) correlate with log(Deaths per mln)?
A: Yes, positive correlation is quite strong, r = 0.7
```

Out[84]: 0.68

Correlation analysis: Cases, Deaths and Response

In [85]:
```
#Creating dataframe on maximum government response index
resp = pd.DataFrame(df_iGovResp.max(axis=1))
ind = pd.DataFrame(df_iGovResp.CountryCode)
Response = (resp.merge(ind,
                       left_index=True,
                       right_index=True)).rename(columns={"CountryCode": "ISO-3",
                                                          0: "Response"})
```

```
(185, 2)
```

In [86]:
```
# Merging maximum government response index into data_all, ignoring 0 deaths
data_all_nonulls = data_all[data_all.Deaths != 0] #filter out 12 countries not reported deaths
data_all_nonulls = data_all_nonulls.merge(Response, on = "ISO-3")
```

```
(158, 13)
```

In [87]:
```
print("Q: Deaths correlates with maximum government response?")
print("A: No, r = 0.2")
r = (np.log(data_all_nonulls.Deaths).corr(data_all_nonulls.Response, method = "pearson"))
```

```
Q: Deaths correlates with maximum government response?
A: No, r = 0.2
```

Out[87]: 0.19

Total deaths are not correlated to maximum government response, as many countries started to take measures before epidemic led to deaths number grouth in order to prevent the losses, and many of them keep high level of restrictions after the deaths number starts to decrease.

In [88]:
```
print("Q: Deaths per mln correlate with maximum government response index?")
print("A: No, r = 0.1")
```

```
Q: Deaths per mln correlate with maximum government response index?
A: No, r = 0.1
```

Out[88]: 0.12

In [89]:
```
print("Q: Cases correlate with maximum government response?")
print("A: No, r = 0.3")
r = (np.log(data_all_nonulls['Cases']).corr(data_all_nonulls.Response, method = "pearson"))
```

```
Q: Cases correlate with maximum government response?
A: No, r = 0.3
```

Out[89]: 0.26

Correlation analysis: government response and government health expenditure

In [95]:
```
# Merging Response and Government health expenditure per capita 2017, international $
import world_bank_data as wb
df_govHealth = pd.DataFrame(wb.get_series('SH.XPD.GHED.PP.CD',date='2017',id_or_value="id",simplify_index=
df_govHealth['ISO-3'] = df_govHealth.index
Response_Funding = Response.merge(df_govHealth, on = 'ISO-3')
```

In [96]:
```
print("Q: Government response index correlates to government health expenditure?")
print("A: No, there is no correlation (corr coeff = 0.06)")
```

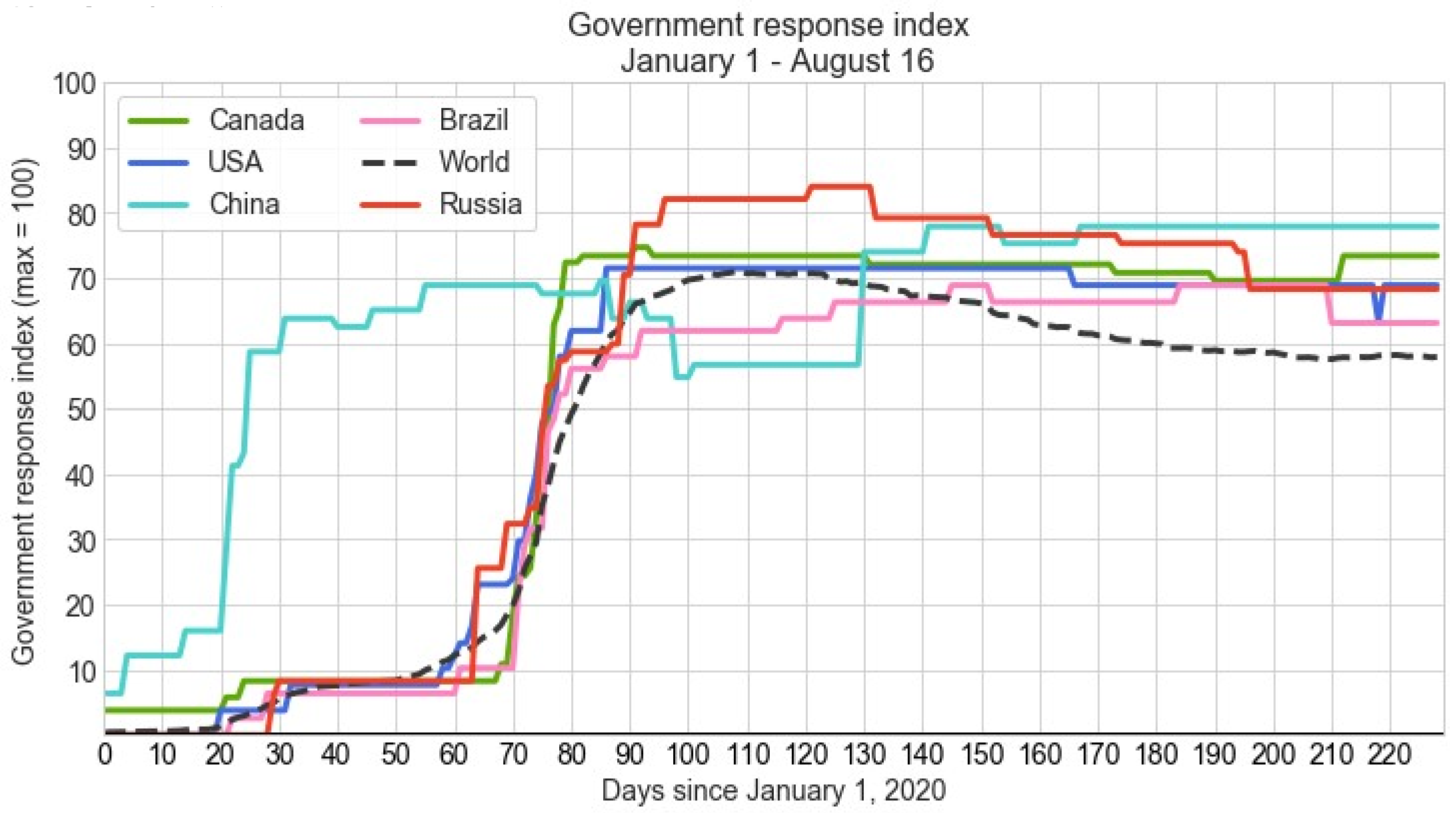
```
Q: Government response index correlates to government health expenditure?
A: No, there is no correlation (corr coeff = 0.06)
0.06
```

Plot: government response index by country over time

Strong correlation between number of confirmed cases and government response index over time is easy to see at the next plot.

In [97]:
```
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation

plt.figure(figsize=(12,6))
plt.style.use('seaborn-whitegrid')

plt.title('Government response index \n January 1 - August 16')
plt.axhline(y = 0, color = 'black', linewidth = 1.3, alpha = .7)
plt.tick_params(axis = 'both', which = 'major')

x = np.linspace(0, 228, 229) # x axis start point, end point and number of intervals
xlim = (0, 229) # y axis start point, end point
y_pos = np.arange(len(wld_iGovResp))

# use the plt.xticks function to custom labels
plt.xticks(y_pos, color='black', rotation=False)
plt.xticks(np.arange(0, 230, 10.0))
plt.yticks(np.arange(10, 110, 10))
```

```
plt.xlabel('Days since January 1, 2020')
plt.ylabel('Government response index (max = 100)')

y1 = np.array(ca_iGovResp.drop(axis=1, columns = 'CountryName').values.tolist()[0])
y2 = np.array(us_iGovResp.drop(axis=1, columns = 'CountryName').values.tolist()[0])
y3 = np.array(cn_iGovResp.drop(axis=1, columns = 'CountryName').values.tolist()[0])
y4 = np.array(br_iGovResp.drop(axis=1, columns = 'CountryName').values.tolist()[0])
y5 = np.array(wld_iGovResp)
y6 = np.array(ru_iGovResp.drop(axis=1, columns = 'CountryName').values.tolist()[0])

ca = plt.plot(x, y1, label='Canada',c="xkcd:leaf green", lw=3, animated =True)
us = plt.plot(x, y2, label='USA', c="royalblue", lw=3, animated =True)
cn = plt.plot(x, y3, label='China', c="mediumturquoise", lw=3, animated =True)
br = plt.plot(x, y4, label='Brazil', c="xkcd:pink", lw=3, animated =True)
wld = plt.plot(x, y5, label='World', ls='--',c='xkcd:dark grey', lw=3, animated =True)
ru = plt.plot(x, y6, label='Russia', c="xkcd:tomato", lw=3, animated =True)

plt.axis([0, 229, 0, 100])
plt.axhline(0, c='black', ls='-', lw=2)
plt.axvline(0, c='black', ls='-', lw=2)
plt.legend(loc='best', frameon=True, fancybox=True, framealpha=0.9, facecolor='white', ncol=2)
```



Correlation analysis: confirmed cases and government response index over time

In [98]:

```
# log(Cases) correlation with Response over time
#RESPONSE Jan 22-Aug 16
y1[21:] #ca_iGovResp Jan 22-Aug 16
y2[21:] #us_iGovResp Jan 22-Aug 16
y3[21:] #cn_iGovResp Jan 22-Aug 16
y4[21:] #br_iGovResp Jan 22-Aug 16
#y5[21:] #wld_iGovResp Jan 22-Aug 16
y6[21:] #ru_iGovResp Jan 22-Aug 16
#print(len(y1[21:]))

#CASES Jan 22-Aug 16
cases_part = pd.read_csv("df_cases.csv")
#print(len(cases_part.Date.unique()))
x1 = np.array(cases_part.Value[cases_part['ISO-3']=='CAN'])
x2 = np.array(cases_part.Value[cases_part['ISO-3']=='USA'])
x3 = np.array(cases_part.Value[cases_part['ISO-3']=='CHN'])
x4 = np.array(cases_part.Value[cases_part['ISO-3']=='BRA'])
#x5 = np.array(cases_part.groupby('Date')['Value'].mean()) #world's average, !sorted
```

In [99]:

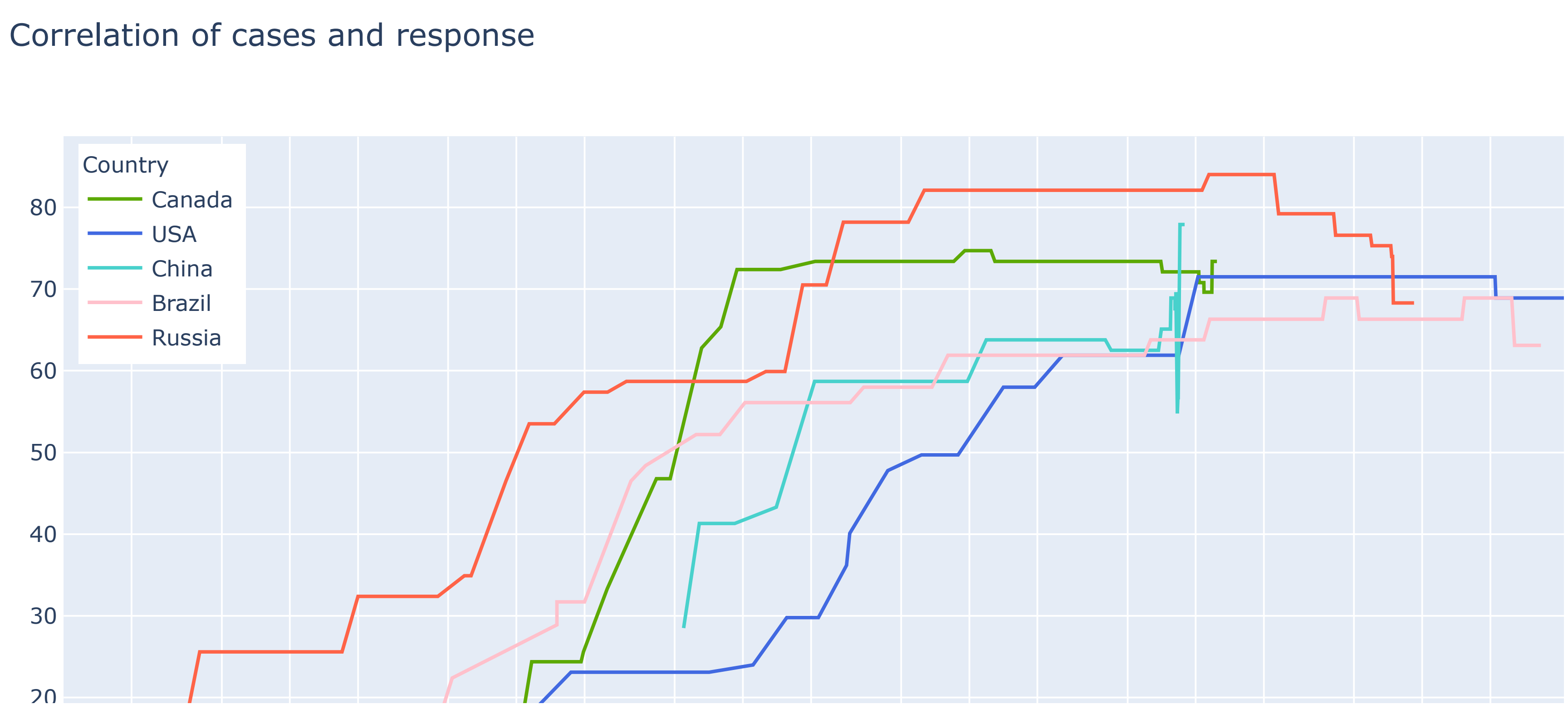
```
Canada = pd.DataFrame(x1,y1[21:])
Canada['Response'] = Canada.index
Canada['Country'] = "Canada"
Canada = Canada.rename(columns={0: "Cases"}).reset_index(drop=True)

USA = pd.DataFrame(x2,y2[21:])
USA['Response'] = USA.index
USA['Country'] = "USA"
USA = USA.rename(columns={0: "Cases"}).reset_index(drop=True)

China = pd.DataFrame(x3,y3[21:])
China['Response'] = China.index
China['Country'] = "China"
China = China.rename(columns={0: "Cases"}).reset_index(drop=True)

Brazil = pd.DataFrame(x4,y4[21:])
Brazil['Response'] = Brazil.index
Brazil['Country'] = "Brazil"
```

```
Brazil = Brazil.rename(columns={0: "Cases"}).reset_index(drop=True)

Russia = pd.DataFrame(x6,y6[21:])
Russia['Response'] = Russia.index
Russia['Country'] = 'Russia'
```

To find HEX codes for this colors (Canada 'leaf green', USA 'royalblue', China 'mediumturquoise", Brazil 'pink', Russia 'tomato'), view the list of colors from mathplotlib:

In [100]:
```python
import matplotlib
colorname = []
colorid = []

for name, hex in matplotlib.colors.cnames.items():
    colorname.append(name)
    colorid.append(hex)
zippedcolors = list(zip(colorname, colorid))
zippedcolors = sorted(zippedcolors, key=lambda x: x[1])
```

In [101]:
```python
# log(Cases) correlation with (Government response index) over time
import plotly.express as px
from plotly.subplots import make_subplots
import plotly.graph_objects as go

fig = make_subplots(rows=1, cols=1)

fig.append_trace(go.Scatter(x = Canada['Cases'],
              y = Canada['Response'], name="Canada",
              mode="lines", line=dict(color='#5ca904')),
              row=1, col=1)

fig.append_trace(go.Scatter(x = USA['Cases'],
              y = USA['Response'], name="USA",
              mode="lines", line=dict(color='#4169E1')),
              row=1, col=1)

fig.append_trace(go.Scatter(x = China['Cases'],
              y = China['Response'], name="China",
              mode="lines", line=dict(color='#48D1CC')),
              row=1, col=1)

fig.append_trace(go.Scatter(x = Brazil['Cases'],
              y = Brazil['Response'], name="Brazil",
              mode="lines", line=dict(color='#FFC0CB')),
              row=1, col=1)

fig.append_trace(go.Scatter(x = Russia['Cases'],
              y = Russia['Response'], name="Russia",
              mode="lines", line=dict(color='#FF6347')),
              row=1, col=1)

fig.layout
fig.update_layout(xaxis_type="log",height=600, width=1000,
                  title_text="Correlation of cases and response",
                  showlegend=True,
                  legend=dict(yanchor="top",
                              y=0.99,
                              xanchor="left",x=0.01),
                  legend_title_text='Country'
                             )
fig.show()

```



Research questions

- is there a significant correlation between cumulative deaths from COVID-19 and maximum government response index?
- is there a significant correlation between government response index and government funding healthcare in previous years?
- is there strong positive correlation between confirmed COVID-19 cases and government response index over time?

Research results

- No, there is no significant correlation between deaths from COVID-19 and government response to COVID. Probably because such government measures are being globally implemented in order to prevent further virus spread and it's damage.
- No, there is no correlation between government response index and government funding healthcare in previous years. The governments that show higher response index to COVID-19 in 2020 were not necesseraly better in funding their healthcare systems in previous years.
- Correlation between confirmed cases and government response over time is very strong (coefficient is equal to 0.94). But correlation between total cumulative number of cases and maximum government response response is very weak (only 0.26), and correlation between deaths and maximum government response is (0.19). It means, that time factor is crucial to government response in order for measures taken have positive effect on the situation with virus spread.

In any case, it is too early to make final conclusions as desease statistics database is only growing.