Python初級數據分析員證書

（六）數據分析 及可視化專案

13. 數據分析專案
Demo5 – Covid-19 Statistics Visualization

Review

- Statistics
- Hypothesis testing
- Algebra
- Linear regression
- Propositional logic
- Python
- R
- SQL

- Pandas, NumPy, SciPy
- Data Visualization, Matplotlib, Seaborn, Plotly
- Dashboard Visualization, Business Intelligence
- Storytelling

Chapter Summary

- Covid-19 Data Visualization
- Data Import
- Color Palettes
- Create plot function
- px.choropleth with time slider
- pd.melt()
- px.treemap

Data import

In this chapter, we are going to visualise practical Covid-19 statistics.

Source of data: https://github.com/CSSEGISandData/COVID-19

The data was collected from WHO and each countries health department and finalised with students in Johns Hopkins University and enthusiasts on GitHub.

Data Import

There are 6 csv datafile in Covid-19 folder. Firstly we import them and have a glance.

Data Import

```
1 import pandas as pd
2 import numpy as np
3
4 import plotly.express as px
```

Data Import

```
1 full_table = pd.read_csv('../covid-19/covid_19_clean_complete.csv')
2 full_table
```

	Province/State	Country/Region	Lat	Long	Date	Confirmed	Deaths	Recovered	Active	WHO Region
0	NaN	Afghanistan	33.939110	67.709953	2020-01-22	0	0	0	0	Eastern Mediterranean
1	NaN	Albania	41.153300	20.168300	2020-01-22	0	0	0	0	Europe
2	NaN	Algeria	28.033900	1.659600	2020-01-22	0	0	0	0	Africa
3	NaN	Andorra	42.506300	1.521800	2020-01-22	0	0	0	0	Europe
4	NaN	Angola	-11.202700	17.873900	2020-01-22	0	0	0	0	Africa

Data Import

Pd.to_datetime is usually a must for time series data, otherwise Pandas will treat it as string.

```
1 full_grouped = pd.read_csv('../covid-19/full_grouped.csv')
2 full_grouped['Date'] = pd.to_datetime(full_grouped['Date'])
3 full_grouped
```

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

Data Import

```
1 day_wise = pd.read_csv('../covid-19/day_wise.csv')
2 day_wise['Date'] = pd.to_datetime(day_wise['Date'])
3 day_wise
```

	Date	Confirmed	Deaths	Recovered	Active	New cases	New deaths	New recovered	Deaths / 100 Cases	Recovered / 100 Cases	Deaths / 100 Recovered	No. of countries
0	2020-01-22	555	17	28	510	0	0	0	3.06	5.05	60.71	6
1	2020-01-23	654	18	30	606	99	1	2	2.75	4.59	60.00	8
2	2020-01-24	941	26	36	879	287	8	6	2.76	3.83	72.22	9
3	2020-01-25	1434	42	39	1353	493	16	3	2.93	2.72	107.69	11
4	2020-01-26	2118	56	52	2010	684	14	13	2.64	2.46	107.69	13

Data Import

```
country_wise = pd.read_csv('../covid-19/country_wise_latest.csv')
country_wise = country_wise.replace('', np.nan).fillna(0)
country_wise
```

	Country/Region	Confirmed	Deaths	Recovered	Active	New cases	New deaths	New recovered	Deaths / 100 Cases	Recovered / 100 Cases	Deaths / 100 Recovered	Confirmed last week	1 week change	1 week % increase	WHO Re
0	Afghanistan	36263	1269	25198	9796	106	10	18	3.50	69.49	5.04	35526	737	2.07	Ea Mediterra
1	Albania	4880	144	2745	1991	117	6	63	2.95	56.25	5.25	4171	709	17.00	Eu
2	Algeria	27973	1163	18837	7973	616	8	749	4.16	67.34	6.17	23691	4282	18.07	A
3	Andorra	907	52	803	52	10	0	0	5.73	88.53	6.48	884	23	2.60	Eu
4	Angola	950	41	242	667	18	1	0	4.32	25.47	16.94	749	201	26.84	A

Data Import

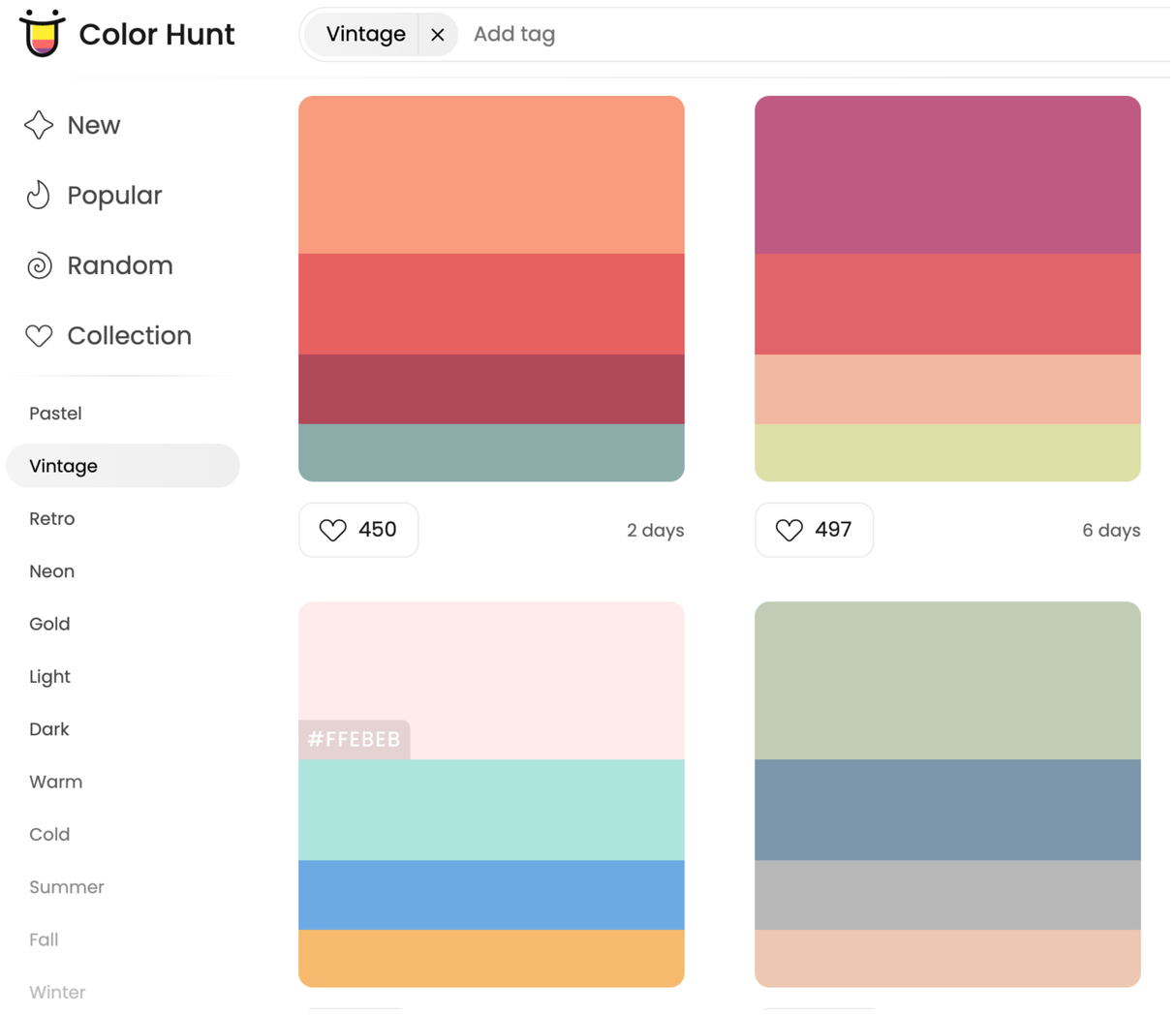
```
worldometer_data = pd.read_csv('../covid-19/worldometer_data.csv')
worldometer_data = worldometer_data.replace('', np.nan).fillna(0)
worldometer_data
```

	Country/Region	Continent	Population	TotalCases	NewCases	TotalDeaths	NewDeaths	TotalRecovered	NewRecovered	ActiveCases	Serious,Critical	C
0	USA	North America	3.311981e+08	5032179	0.0	162804.0	0.0	2576668.0	0.0	2292707.0	18296.0	
1	Brazil	South America	2.127107e+08	2917562	0.0	98644.0	0.0	2047660.0	0.0	771258.0	8318.0	
2	India	Asia	1.381345e+09	2025409	0.0	41638.0	0.0	1377384.0	0.0	606387.0	8944.0	
3	Russia	Europe	1.459409e+08	871894	0.0	14606.0	0.0	676357.0	0.0	180931.0	2300.0	
4	South Africa	Africa	5.938157e+07	538184	0.0	9604.0	0.0	387316.0	0.0	141264.0	539.0	

Color Palettes

Coloring on graph is much important than many people think. Especially when we preparing visual aids for presentation and public report. In general you may follow this simple theory.

Color Palettes



There are many color palettes (with rgb#) online resources for you to pick series of theme color.

https://colorhunt.co/

Colour Palettes

In our Covid-19 visualization, it is relatively simple. We fix these four colour on our decision, in case of picking by plotly default.

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

```
temp = day_wise[['Date','Deaths', 'Recovered', 'Active']].tail(1)
temp = temp.melt(id_vars="Date", value_vars=['Active', 'Deaths', 'Recovered'])
fig = px.treemap(temp, path=["variable"], values="value", height=225,
                 color_discrete_sequence=[act, rec, dth])
fig.data[0].textinfo = 'label+text+value'
fig.show()
```

Create a plot function

DRY, in programming norm, Don't Repeat Yourself. If you need to plot same kind of plots more than once, you should create a function.

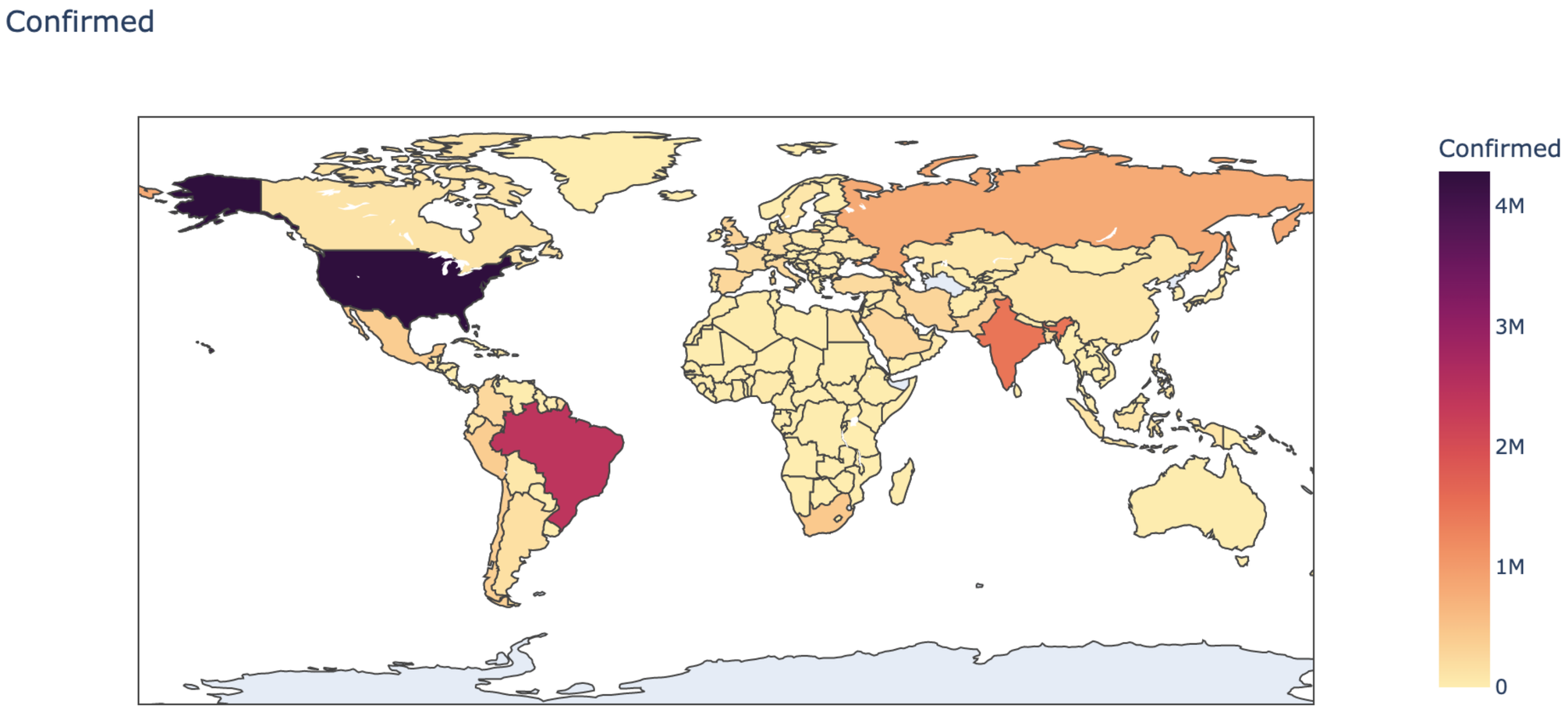
col is column in df; pal is color palettes series

```
1  def plot_map(df, col, pal):
2      df = df[df[col]>0]
3      fig = px.choropleth(df, locations="Country/Region", locationmode='country names',
4                          color=col, hover_name="Country/Region",
5                          title=col, hover_data=[col], color_continuous_scale=pal)
6  #      fig.update_layout(coloraxis_showscale=False)
7      fig.show()
```

For more choice on continuous color series, go to:
https://plotly.com/python/builtin-colorscales/

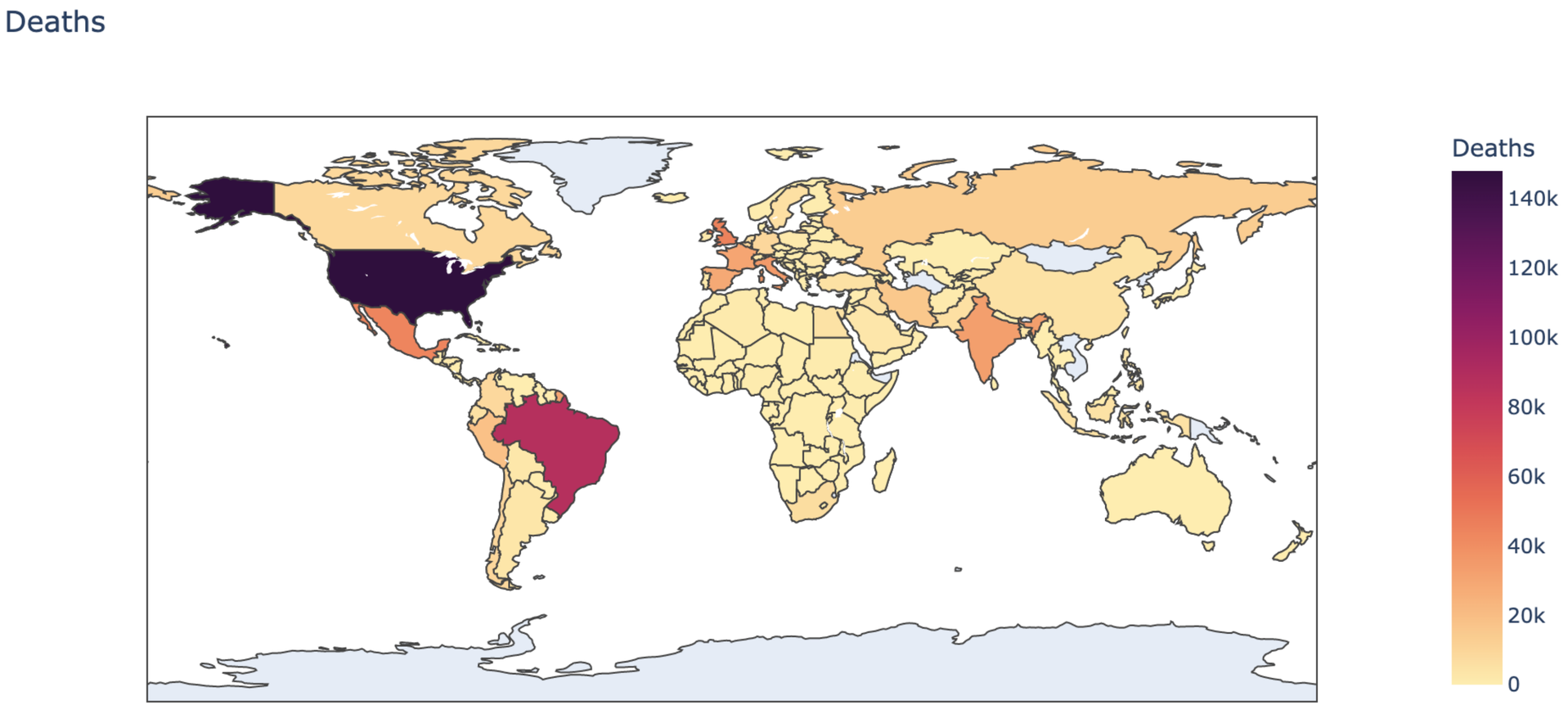
Plotting confirmed case

```
1 plot_map(country_wise, 'Confirmed', 'matter')
```



Plotting death cases

```
1 plot_map(country_wise, 'Deaths', 'matter')
```

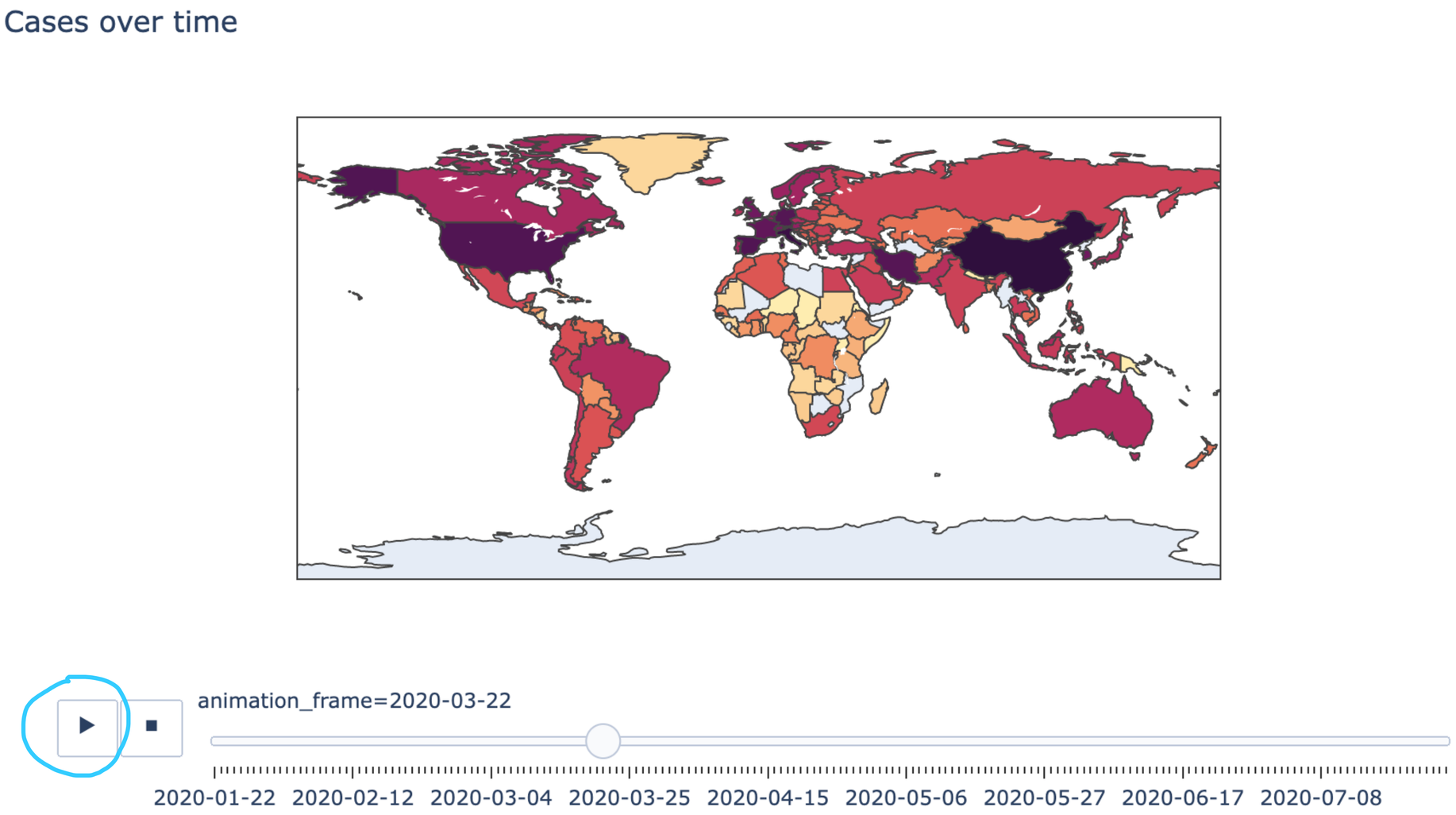


Choropleth with time slider

Covid-19 outbreak spread rapidly, we may plot an interactive graph to review how it did.

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

Choropleth with time slider



Case over time

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

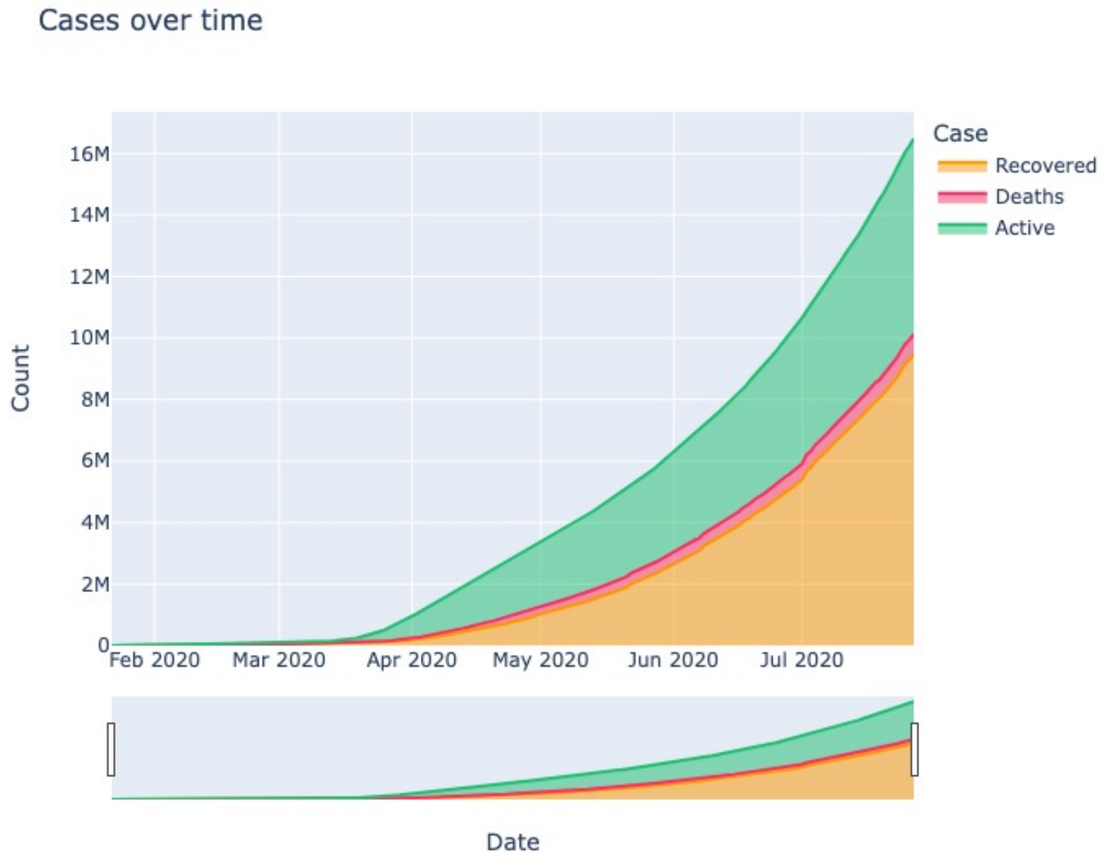
	Date	Case	Count
0	2020-01-22	Recovered	28
1	2020-01-23	Recovered	30
2	2020-01-24	Recovered	36
3	2020-01-25	Recovered	39
4	2020-01-26	Recovered	52

pd.melt(*frame, id_vars=None, value_vars=None, var_name=None, value_name='value', col_level=None, ignore_index=True*)

Unpivot a DataFrame from wide to long format, optionally leaving identifiers set.

Case over time

```
1  fig = px.area(temp, x="Date", y="Count", color='Case', height=600, width=700,
2                title='Cases over time', color_discrete_sequence = [rec, dth, act])
3  fig.update_layout(xaxis_rangeslider_visible=True)
4  fig.show()
```

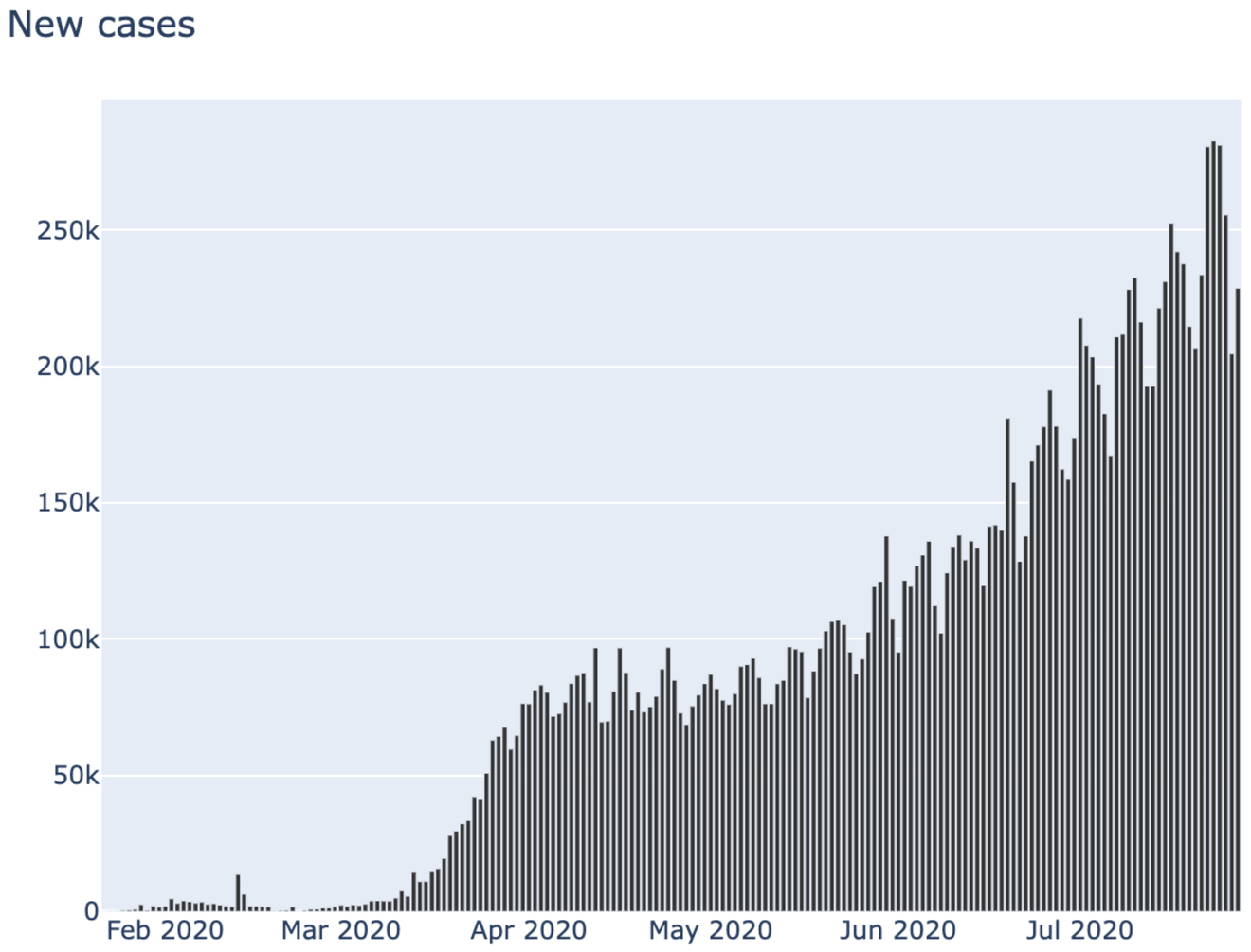


Create function to plot bar and line chart

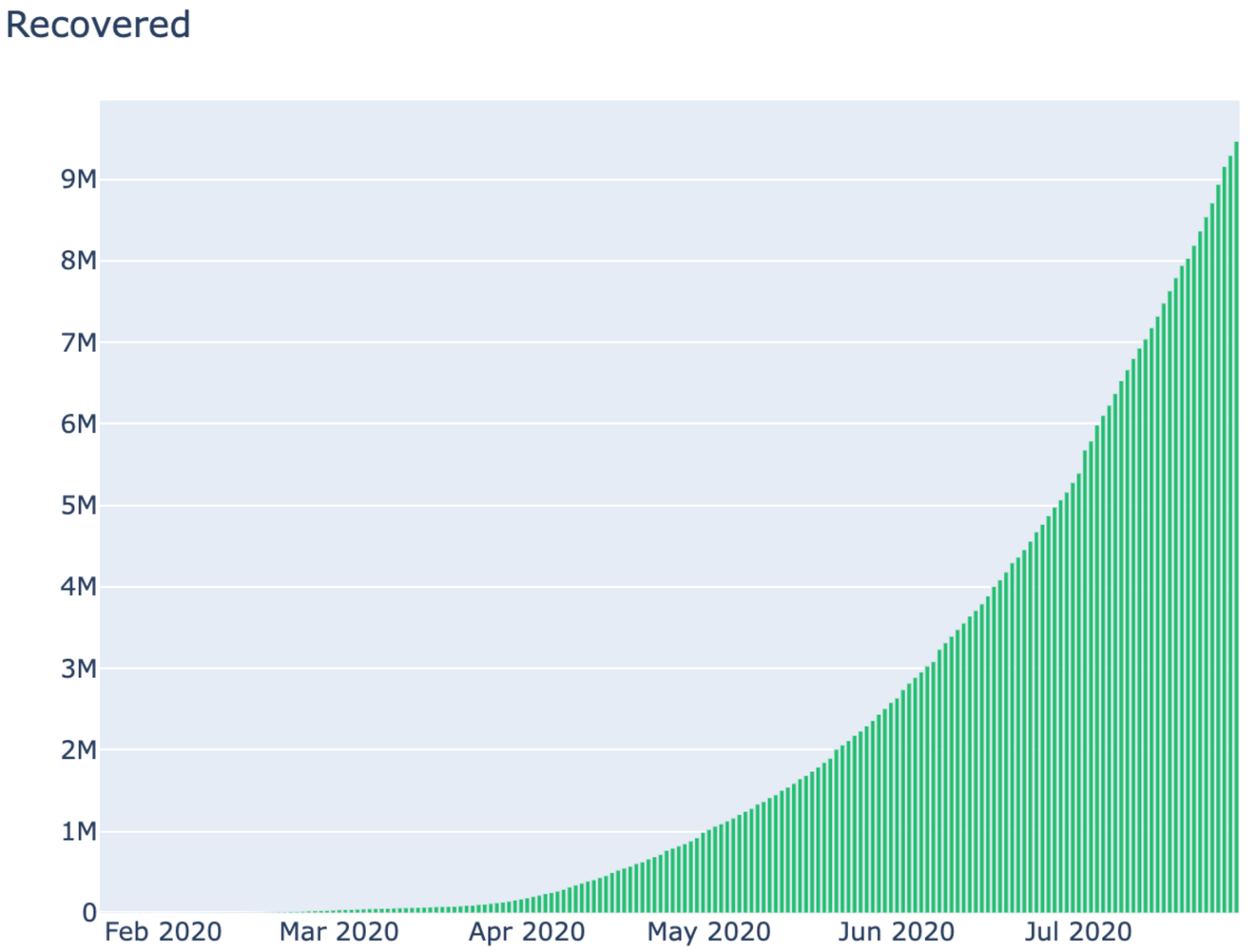
```
1  def plot_daywise(col, hue):
2      fig = px.bar(day_wise, x="Date", y=col, width=700, color_discrete_sequence=[hue])
3      fig.update_layout(title=col, xaxis_title="", yaxis_title="")
4      fig.show()
```

```
1  def plot_daywise_line(col, hue):
2      fig = px.line(day_wise, x="Date", y=col, width=700, color_discrete_sequence=[hue])
3      fig.update_layout(title=col, xaxis_title="", yaxis_title="")
4      fig.show()
```

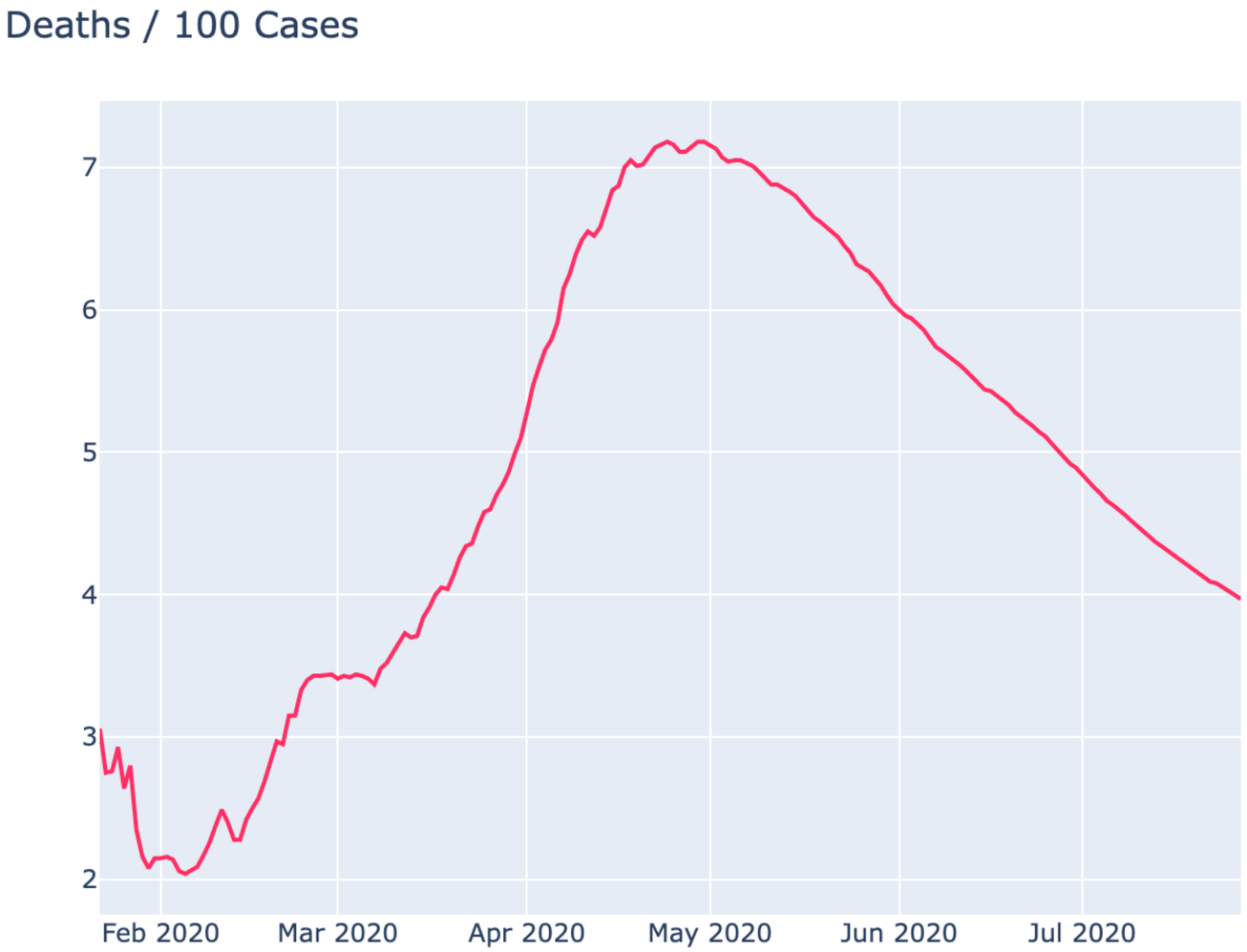
```
1 plot_daywise('New cases', '#333333')
```



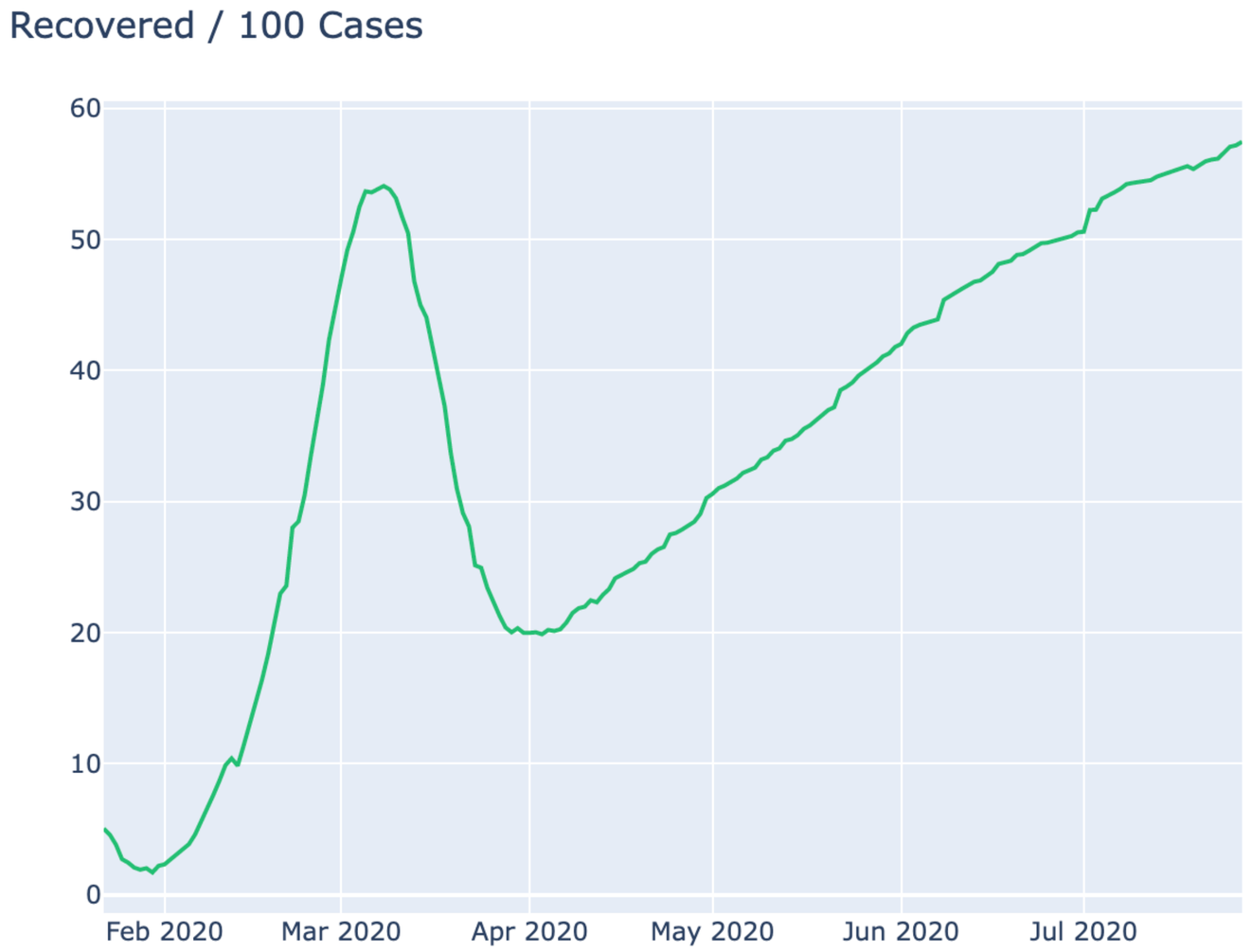
```
1 plot_daywise('Recovered', rec)
```



```
1 plot_daywise_line('Deaths / 100 Cases', dth)
```

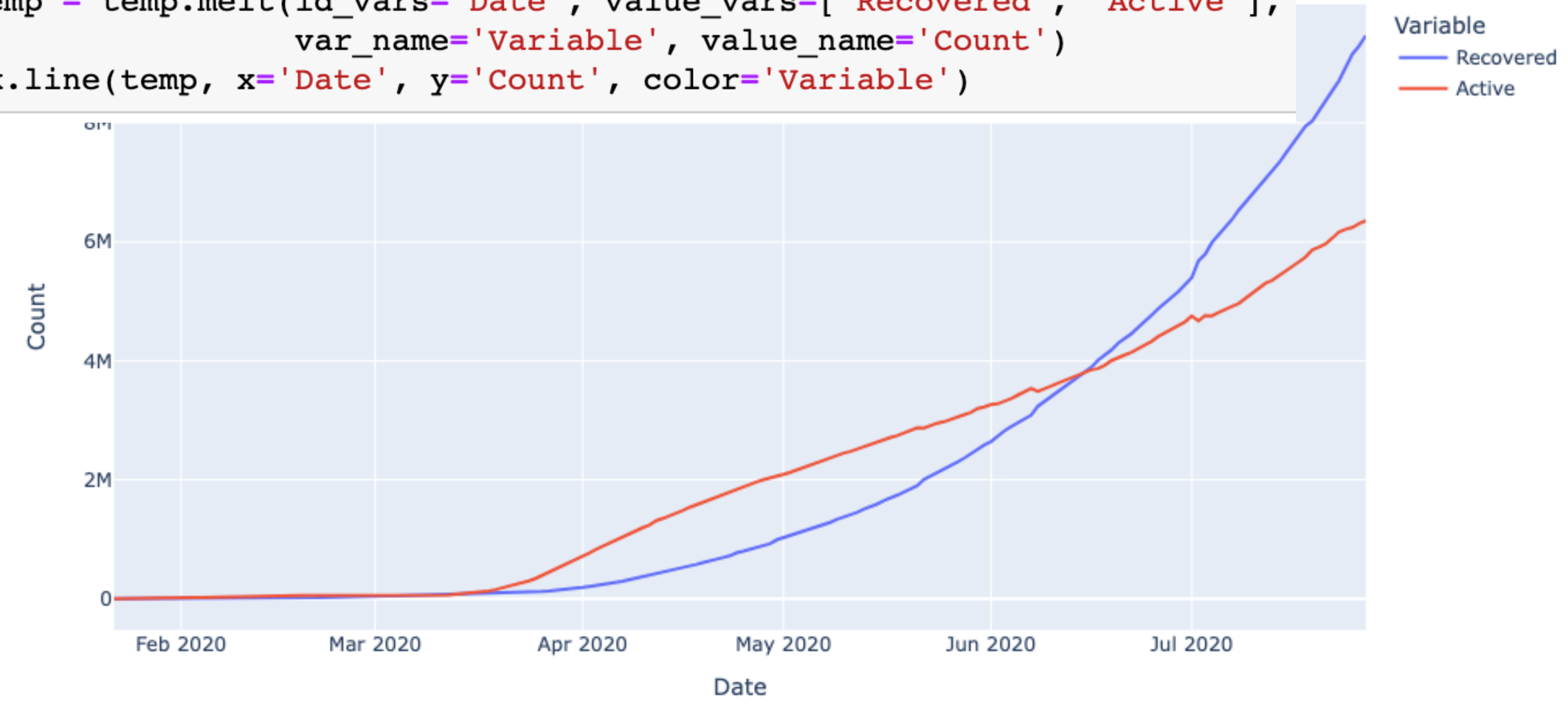


```
1 plot_daywise_line('Recovered / 100 Cases', rec)
```



Plotting recover and active case

```
temp = day_wise[['Date', 'Recovered', 'Active']]
temp = temp.melt(id_vars='Date', value_vars=['Recovered', 'Active'],
                 var_name='Variable', value_name='Count')
px.line(temp, x='Date', y='Count', color='Variable')
```

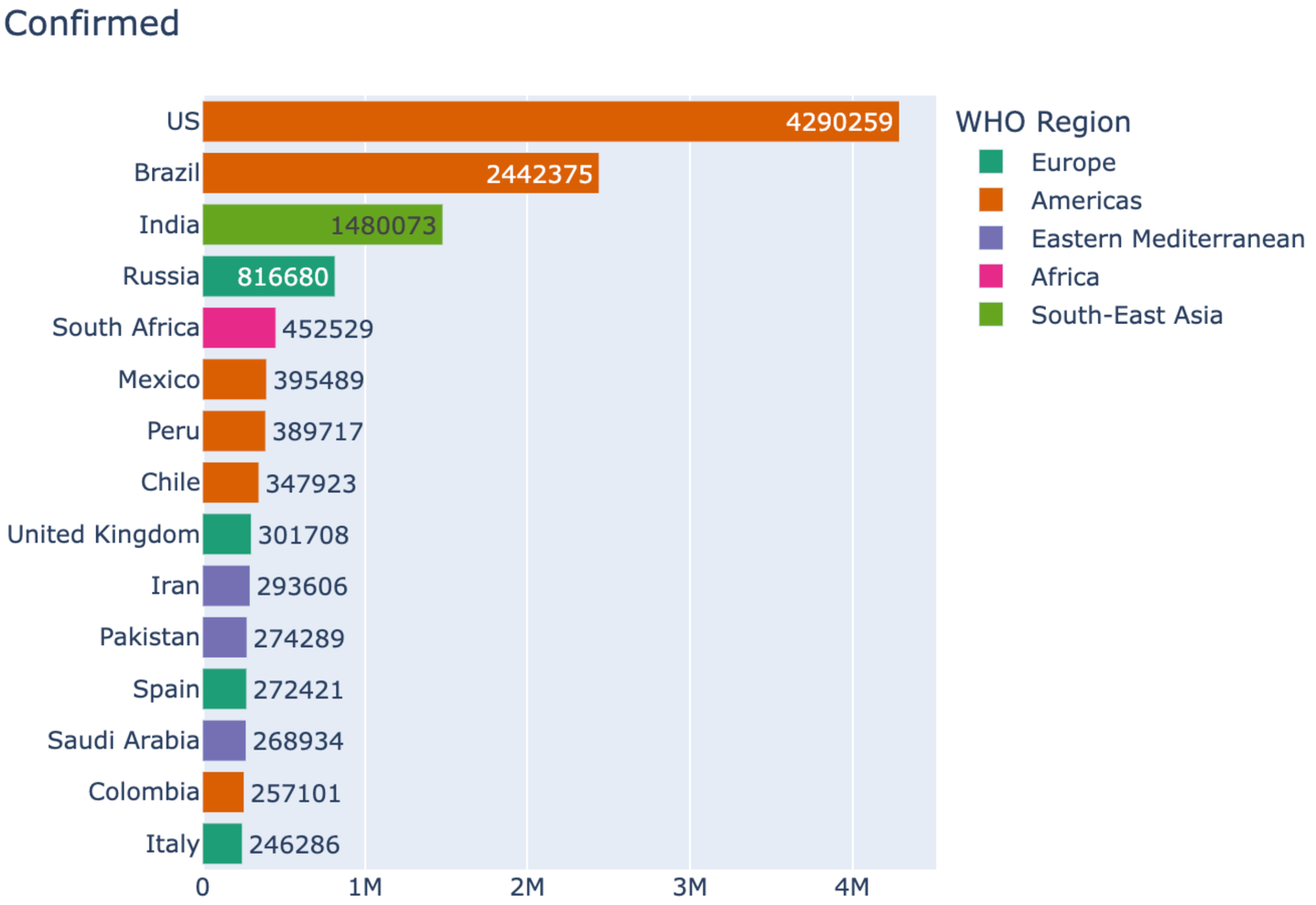


Horizontal Bar

```
def plot_hbar(df, col, n, hover_data=[]):
    fig = px.bar(df.sort_values(col).tail(n),
                 x=col, y="Country/Region", color='WHO Region',
                 text=col, orientation='h', width=700, hover_data=hover_data,
                 color_discrete_sequence = px.colors.qualitative.Dark2)
    fig.update_layout(title=col, xaxis_title="", yaxis_title="",
                      yaxis_categoryorder = 'total ascending',
                      uniformtext_minsize=8, uniformtext_mode='hide')
    fig.show()
```

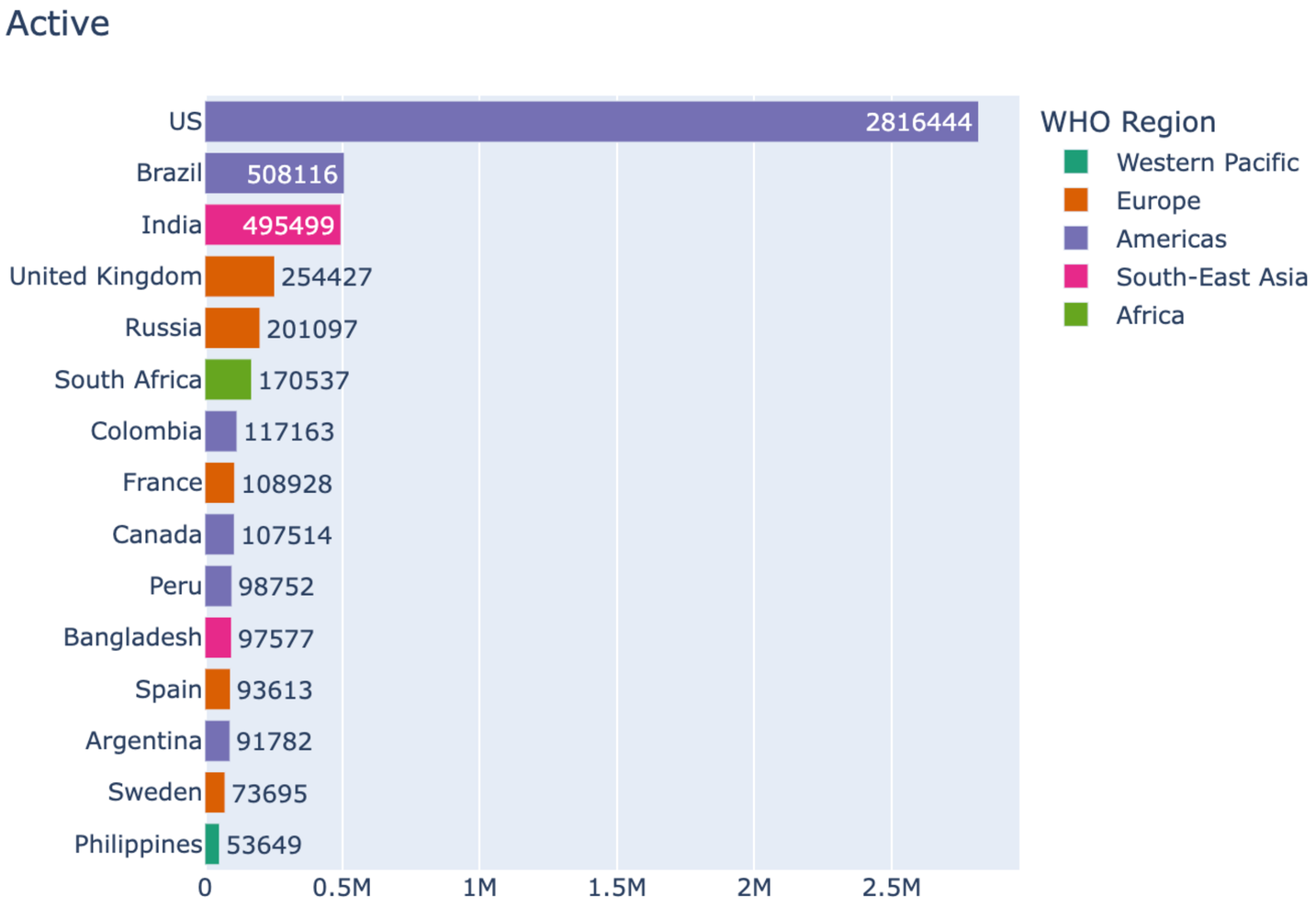
Horizontal Bar

```
1 plot_hbar(country_wise, 'Confirmed', 15)
```



Horizontal Bar

```
1 plot_hbar(country_wise, 'Active', 15)
```

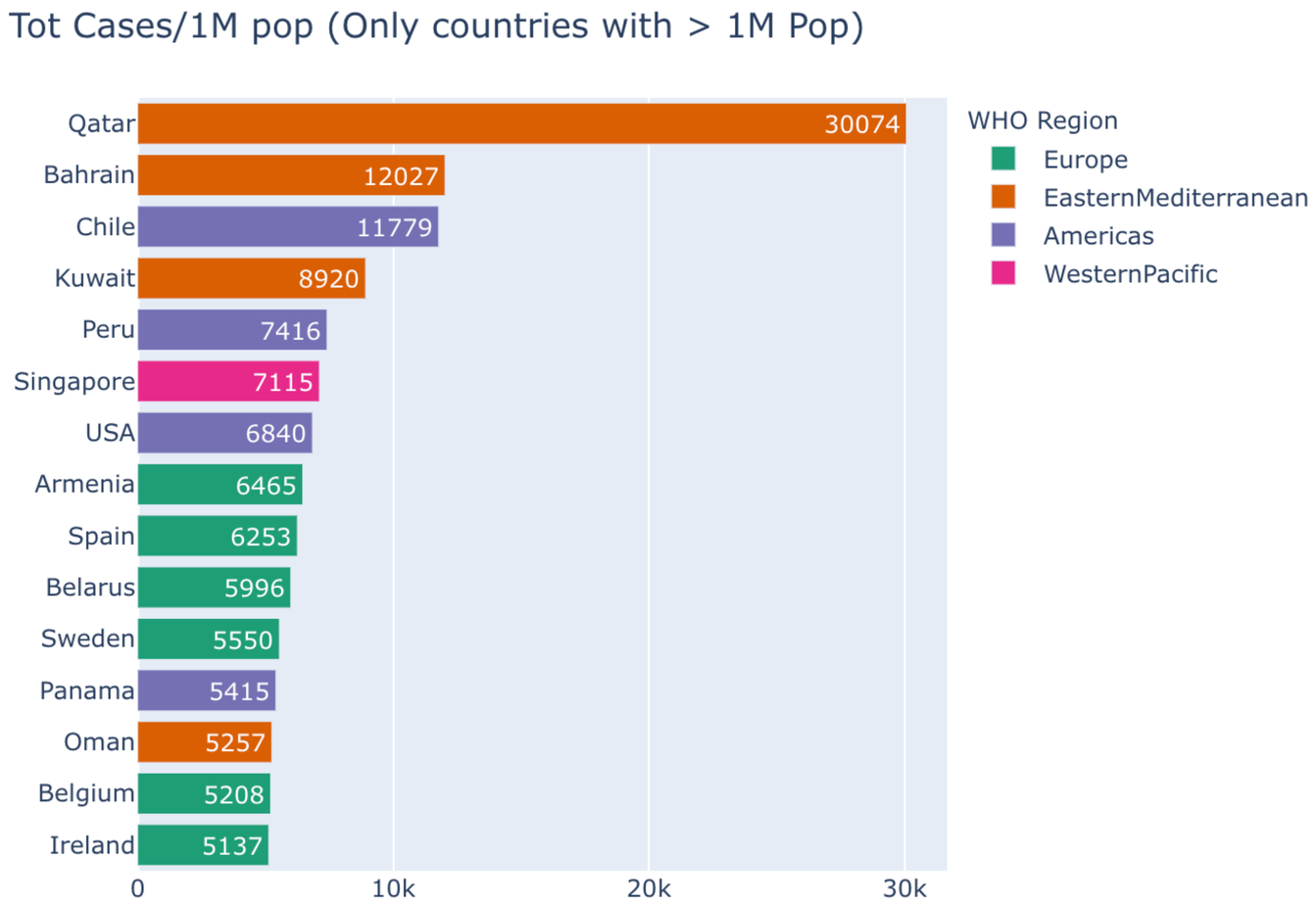


Horizontal Bar

```
def plot_hbar_wm(col, n, min_pop=1000000, sort='descending'):
    df = worldometer_data[worldometer_data['Population']>min_pop]
    df = df.sort_values(col, ascending=True).tail(n)
    fig = px.bar(df,
                 x=col, y="Country/Region", color='WHO Region',
                 text=col, orientation='h', width=700,
                 color_discrete_sequence = px.colors.qualitative.Dark2)
    fig.update_layout(title=col+' (Only countries with > 1M Pop)',
                      xaxis_title="", yaxis_title="",
                      yaxis_categoryorder = 'total ascending',
                      uniformtext_minsize=8, uniformtext_mode='hide')
    fig.show()
```

Horizontal Bar

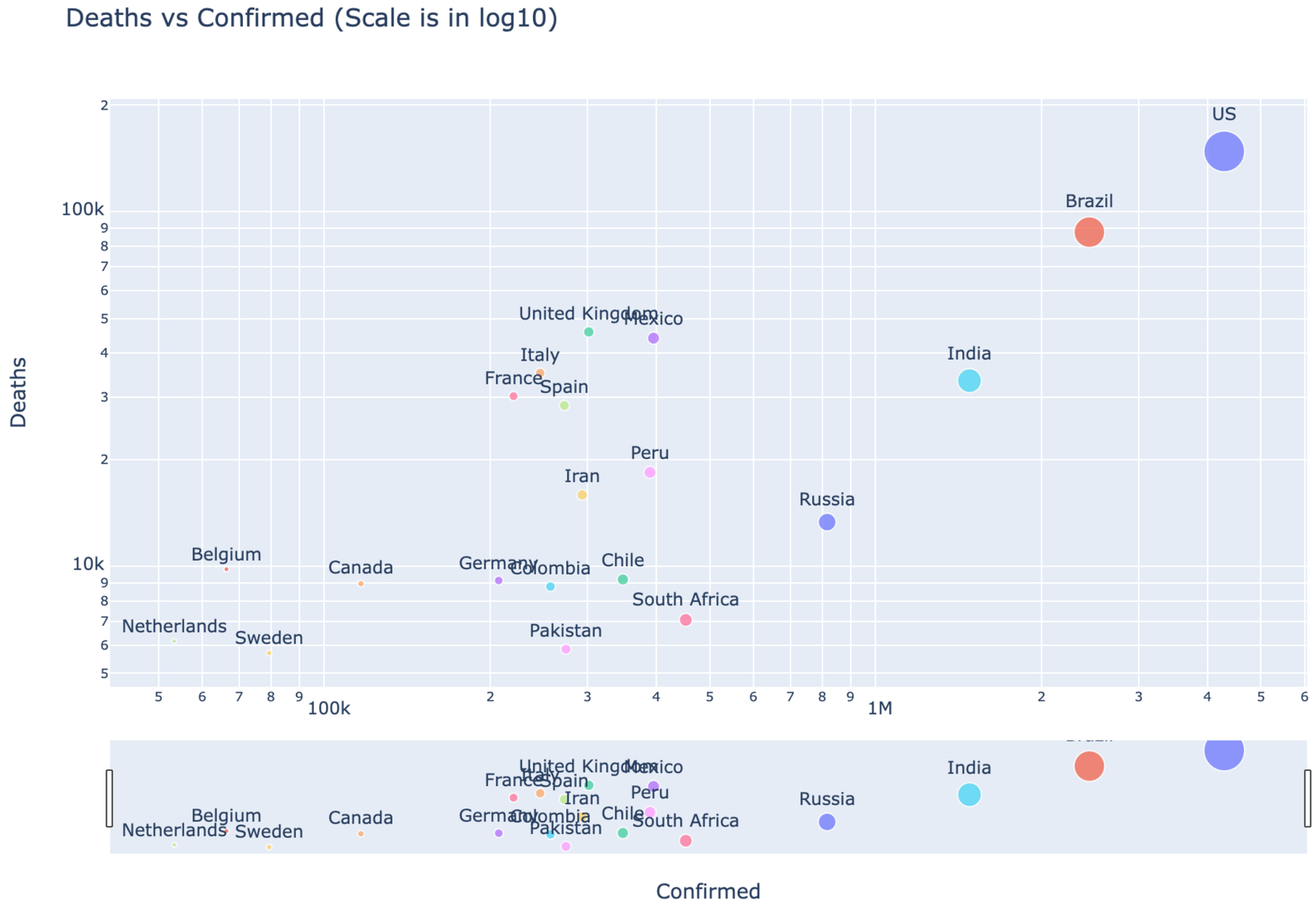
```
plot_hbar_wm('Tot Cases/1M pop', 15, 1000000)
```



Scatter plot on Confirmed vs Deaths

```
fig = px.scatter(country_wise.sort_values('Deaths', ascending=False).iloc[:20, :],
                 x='Confirmed', y='Deaths', color='Country/Region', size='Confirmed',
                 height=700, text='Country/Region', log_x=True, log_y=True,
                 title='Deaths vs Confirmed (Scale is in log10)')
fig.update_traces(textposition='top center')
fig.update_layout(showlegend=False)
fig.update_layout(xaxis_rangeslider_visible=True)
fig.show()
```

Scatter plot on Confirmed vs Deaths



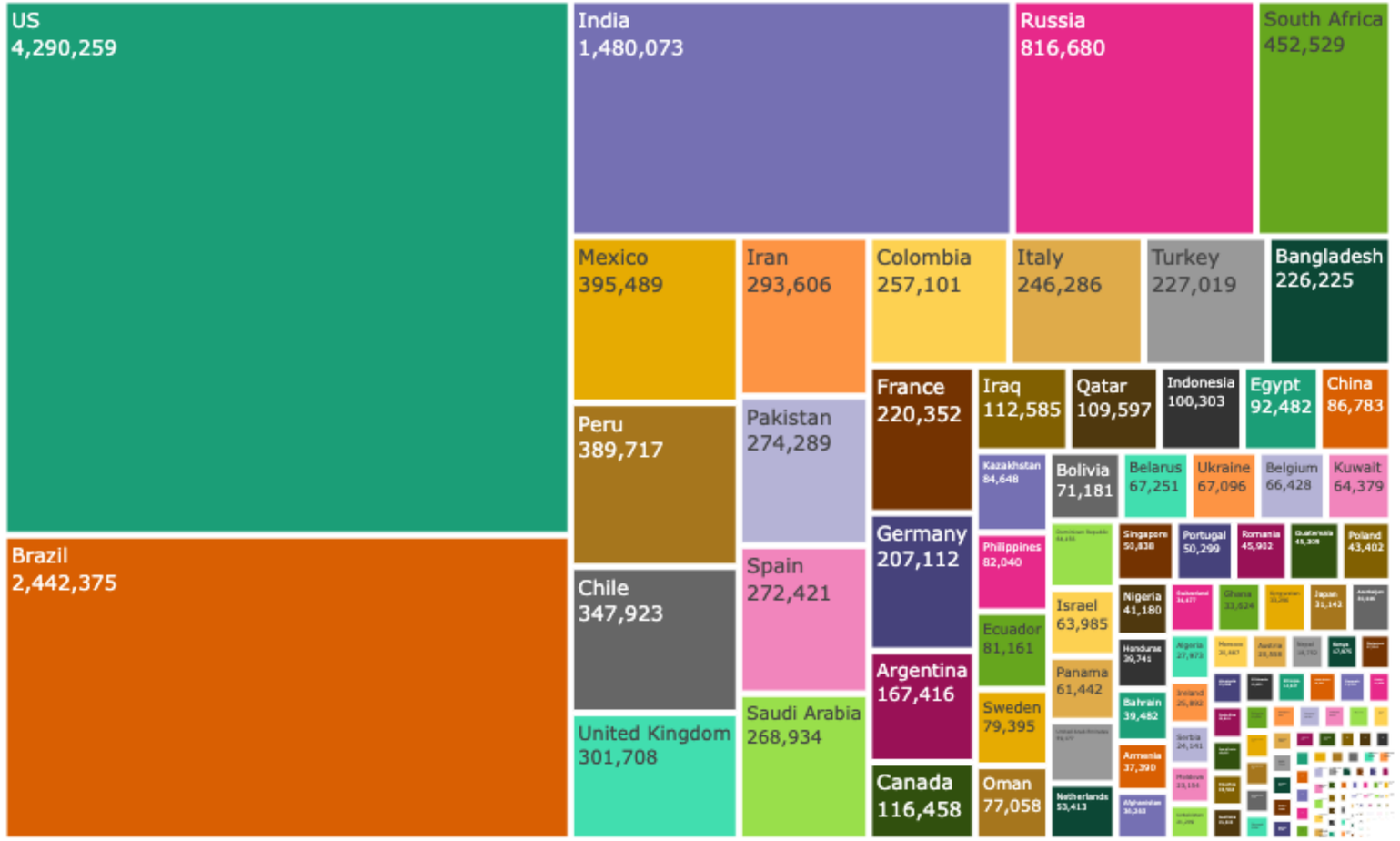
Composition with px.treemap

```
def plot_treemap(col):
    fig = px.treemap(country_wise, path=["Country/Region"], values=col, height=700,
                     title=col, color_discrete_sequence = px.colors.qualitative.Dark2)
    fig.data[0].textinfo = 'label+text+value'
    fig.show()
```

Composition with px.treemap

Confirmed

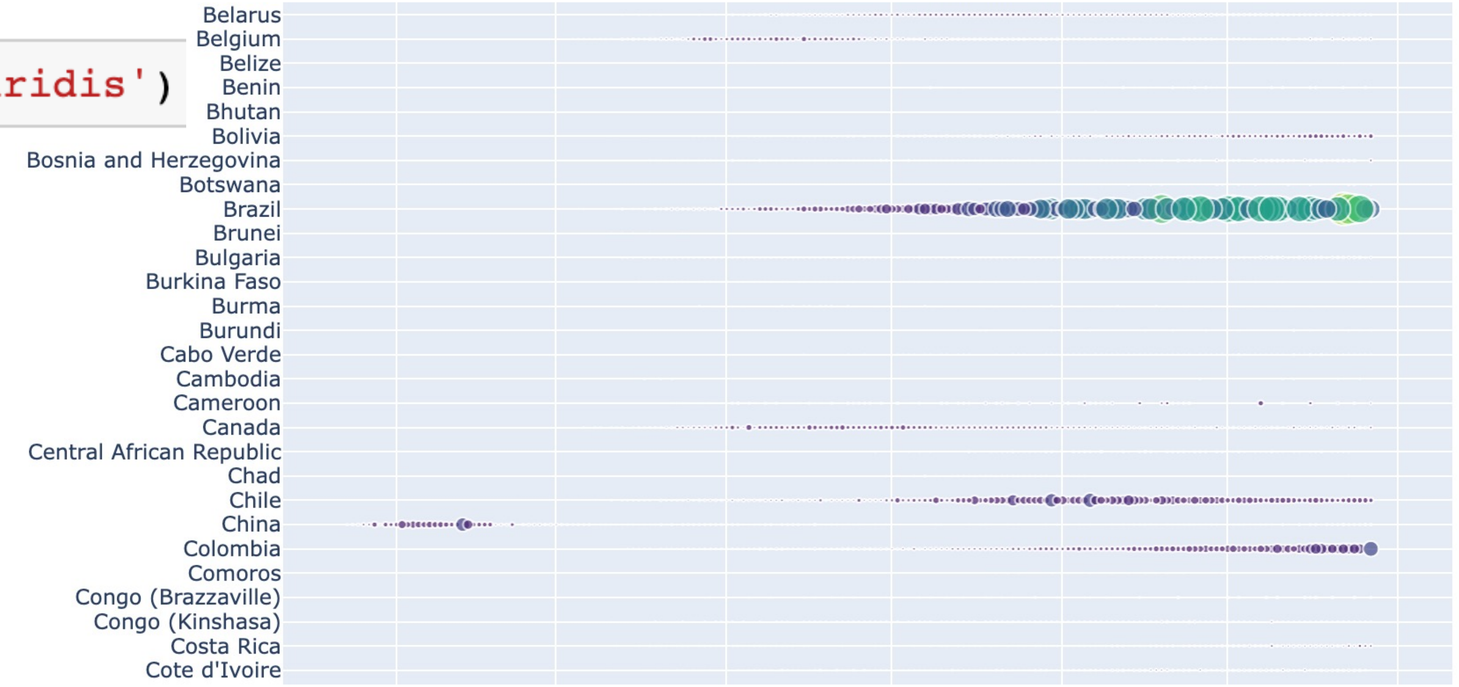
```
1 plot_treemap('Confirmed')
```



Bubble plot a long list

```
def plot_bubble(col, pal):
    temp = full_grouped[full_grouped[col]>0].sort_values('Country/Region', ascending=False)
    fig = px.scatter(temp, x='Date', y='Country/Region', size=col, color=col, height=3000,
                     color_continuous_scale=pal)
    fig.update_layout(yaxis = dict(dtick = 1))
    fig.update(layout_coloraxis_showscale=False)
    fig.show()
```

```
plot_bubble('New cases', 'Viridis')
```



Confirmed vs Deaths

```
temp = worldometer_data[worldometer_data['WHO Region']!=0]
```

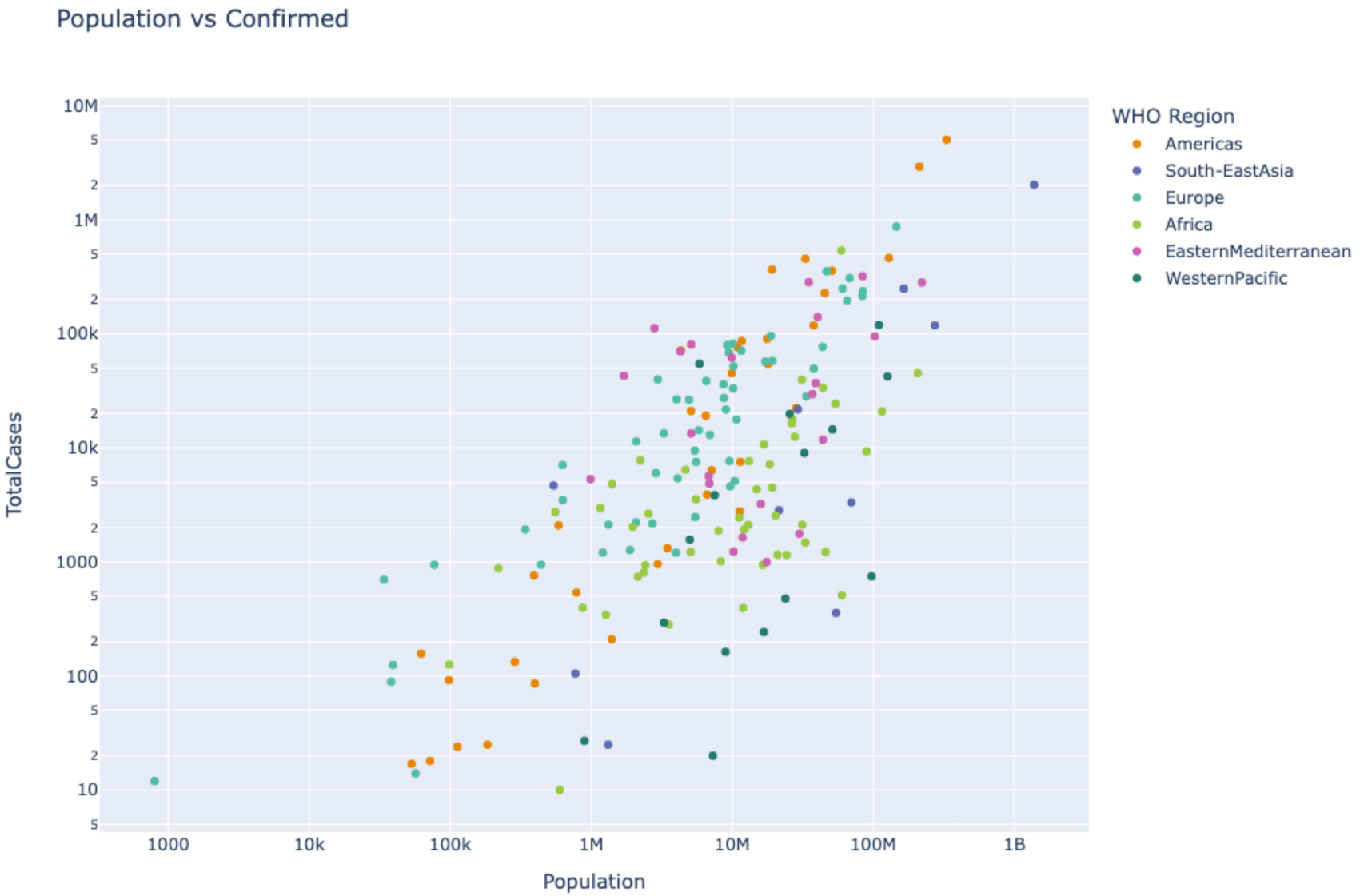
```
fig = px.scatter(temp, x='TotalCases', y='TotalDeaths', color='WHO Region',
                 height=700, hover_name='Country/Region', log_x=True, log_y=True,
                 title='Confirmed vs Deaths',
                 color_discrete_sequence=px.colors.qualitative.Vivid)
fig.update_traces(textposition='top center')
# fig.update_layout(showlegend=False)
# fig.update_layout(xaxis_rangeslider_visible=True)
fig.show()
```

Confirmed vs Deaths

Populations vs Confirmed

```
fig = px.scatter(temp, x='Population', y='TotalCases', color='WHO Region',
                 height=700, hover_name='Country/Region', log_x=True, log_y=True,
                 title='Population vs Confirmed',
                 color_discrete_sequence=px.colors.qualitative.Vivid)
fig.update_traces(textposition='top center')
# fig.update_layout(showlegend=False)
# fig.update_layout(xaxis_rangeslider_visible=True)
fig.show()
```

Populations vs Confirmed



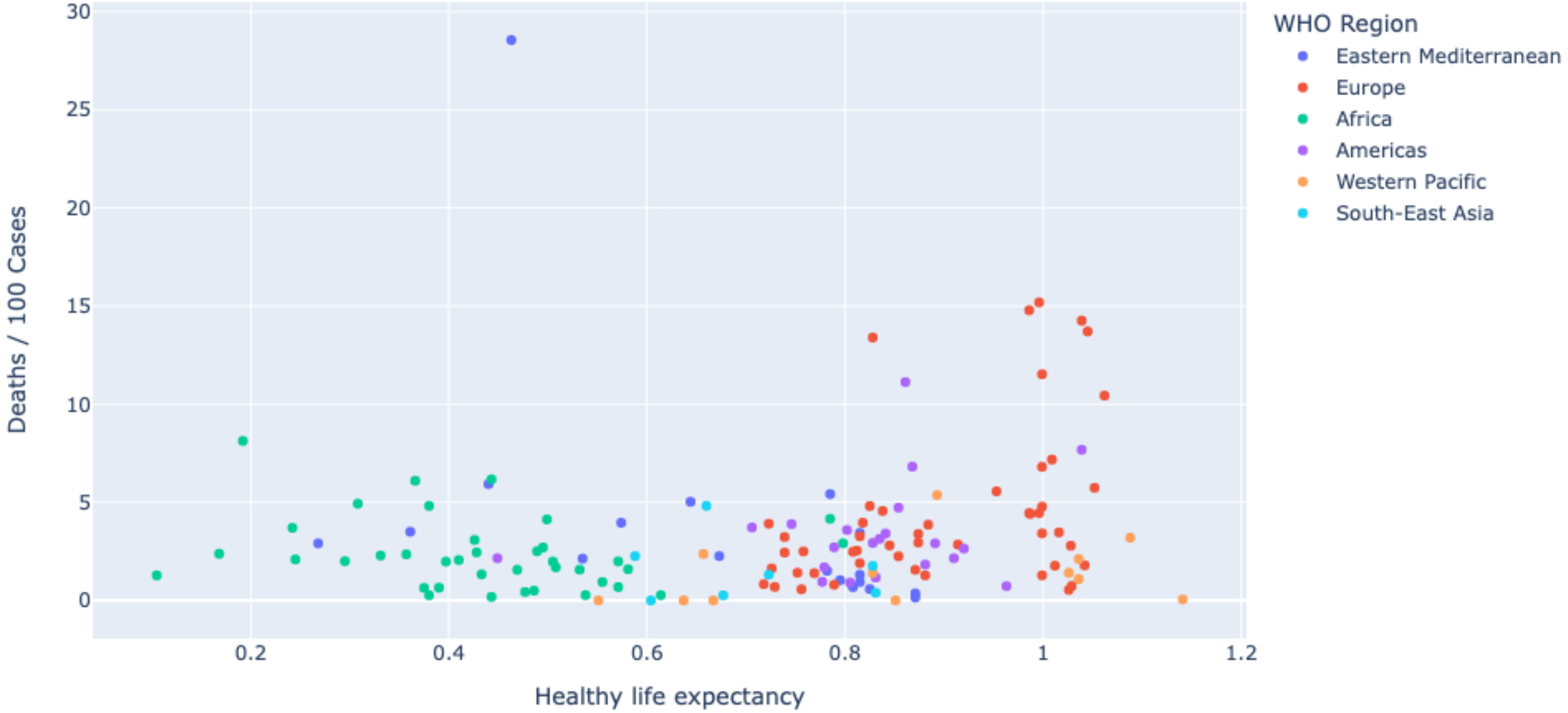
Deaths vs Health Life Expectancy

Pd.merge() to combine 2 DFs on key.

```
happiness_report = pd.read_csv('../Covid-19/Happiness2019.csv')
happiness_report = happiness_report[['Country or region', 'Healthy life expectancy']]

temp = country_wise.merge(happiness_report, left_on='Country/Region', right_on='Country or region')
px.scatter(temp, y='Deaths / 100 Cases', x='Healthy life expectancy', color='WHO Region',
                    hover_data=['Country/Region'])
```

Deaths vs Health Life Expectancy



Chapter Wrap Up

PX could be that easy and helpful. In this chapter, we demonstrate practical usage of data visualization of Covid-19 data.

Plotting function may be needed for plotting a few same kind of graph.

Once you clear the coding, everything is at your finger tip.

Reference & Resources

Official Website:

https://plotly.com/python/

Plotly Graph Objects:

https://plotly.com/python/graph-objects/

WorldBank API:

- https://blogs.worldbank.org/opendata/introducing-wbgapi-new-python-package-accessing-world-bank-data
- https://nbviewer.org/github/tgherzog/wbgapi/blob/master/examples/wbgapi-cookbook.ipynb
- https://pypi.org/project/wbgapi/

GitHub Open Source Code:

https://github.com/plotly/plotly.py