Visualizations

Data:

```
In [1]: import plotly.graph_objects as go
        import pandas as pd
        df_all = pd.read_csv('data/all_data.csv', parse_dates=['date'])
        df_all.tail()
```

Out[1]:

	group	date	area	Daily Deaths	Daily Cases	Deaths	Cases
81978	usa	2020-12-05	Virginia	16	894	4189	217527
81979	usa	2020-12-05	Washington	13	762	2768	142733
81980	usa	2020-12-05	West Virginia	6	319	662	36850
81981	usa	2020-12-05	Wisconsin	66	5996	4335	444479
81982	usa	2020-12-05	Wyoming	5	443	226	28675

Use California as an Example:

```
In [2]: df_california = df_all.query('group == "usa" and area == "California"')
        df_california = df_california.set_index('date')
        df_california.tail()
```

Out[2]:

	group	area	Daily Deaths	Daily Cases	Deaths	Cases
date						
2020-12-01	usa	California	68	3354	19716	1060993
2020-12-02	usa	California	68	3328	19784	1064321
2020-12-03	usa	California	67	3301	19851	1067622
2020-12-04	usa	California	67	3274	19918	1070896
2020-12-05	usa	California	67	3247	19985	1074143

```
In [3]: df_summary = pd.read_csv('data/summary.csv', parse_dates=['date'])
        df_summary.head()
```

Out[3]:

	group	area	Daily Deaths	Daily Cases	Deaths	Cases	code	population	Deaths per Million	Cases per Million	date
0	world	Afghanistan	4	86	1548	41814	AFG	38.928341	40.0	1070.0	2020-11-05
1	world	Albania	7	421	543	22721	ALB	2.877800	189.0	7900.0	2020-11-05
2	world	Algeria	12	642	2011	60169	DZA	43.851043	46.0	1370.0	2020-11-05
3	world	Andorra	0	90	75	5135	AND	0.077265	971.0	66460.0	2020-11-05
4	world	Angola	3	289	299	12102	AGO	32.866268	9.0	370.0	2020-11-05

```
In [4]: last_date = df_summary['date'].iloc[0]
        first_pred_date = last_date + pd.Timedelta('1D')
        last_date, first_pred_date
```

Out[4]: (Timestamp('2020-11-05 00:00:00'), Timestamp('2020-11-06 00:00:00'))

```
In [5]: x = df_california.index
        y = df_california['Deaths']
        fig = go.Figure()
        fig.add_scatter(x=x, y=y, mode="lines+markers")
```
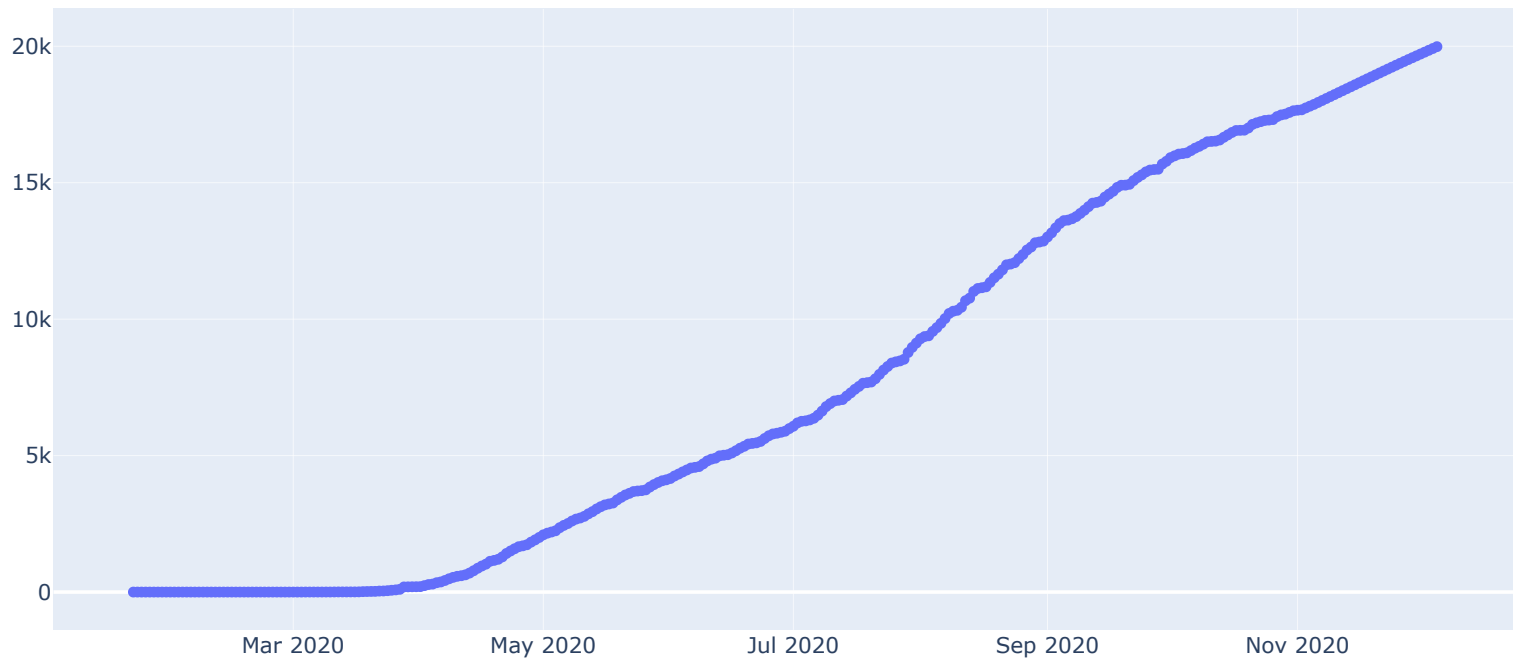
Adding Traces to Graphs

```
In [6]: import plotly.graph_objects as go
        df_california_actual = df_california[:last_date]
        df_california_pred = df_california[first_pred_date:]
        fig = go.Figure()
        fig.add_scatter(x=df_california_actual.index,
                        y=df_california_actual['Deaths'],
                        mode="lines+markers",
                        name='actual')
        fig.add_scatter(x=df_california_pred.index,
                        y=df_california_pred['Deaths'],
                        mode="lines+markers",
                        name='prediction')
        fig.update_layout(height=400, width=800)
```
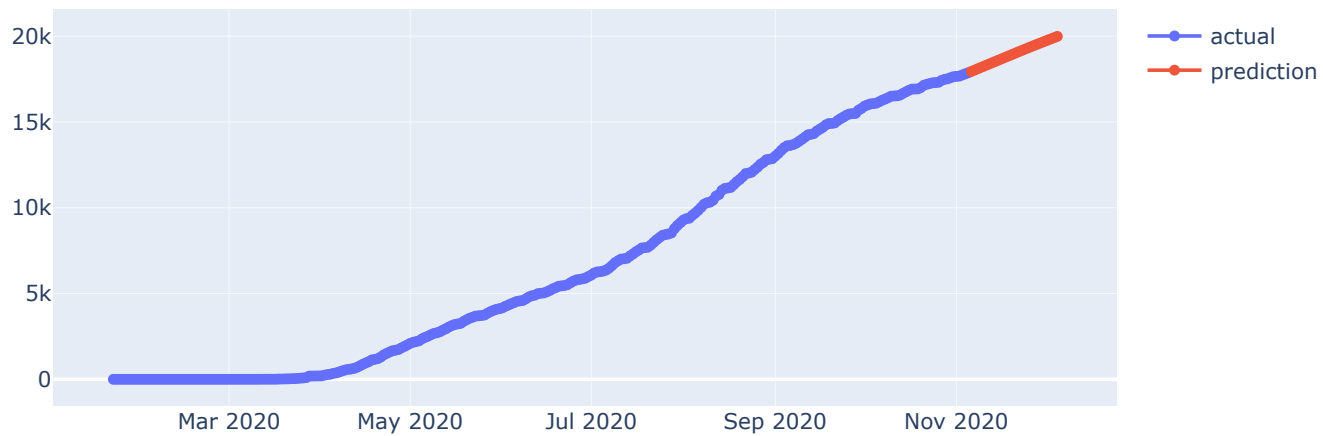
```
In [7]: import plotly.graph_objects as go

fig = go.Figure()
fig.add_scatter(x=x, y=y, mode="lines+markers")
fig.update_layout(height=400,
                  width=800,
                  title="COVID-19 Deaths in California")
```

COVID-19 Deaths in California
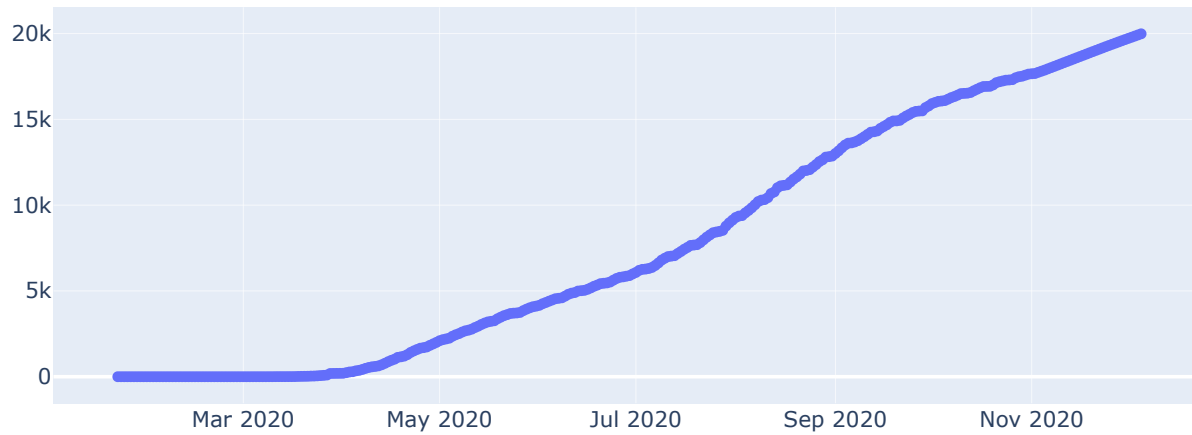
```python
In [8]: def area_bar_plot(df, group, area, kind, last_date, first_pred_date):
            """
            Generates a bar plot of actual vs. predicted values for an area.

            Parameters
            ----------
            df : DataFrame
            group, area : str
            kind : str
            last_date, first_pred_date : str

            Returns
            -------
            plotly.graph_objects.Figure
            """
            df = df.query("group == @group and area == @area").set_index("date")
            df_actual = df[:last_date]
            df_pred = df[first_pred_date:]
            fig = go.Figure()
            fig.add_bar(x=df_actual.index, y=df_actual[kind], name="actual")
            fig.add_bar(x=df_pred.index, y=df_pred[kind], name="prediction")
            return fig
```

```
from functions import area_bar_plot
area_bar_plot(df_all, 'usa', 'California', 'Cases', last_date, first_pred_date)
```
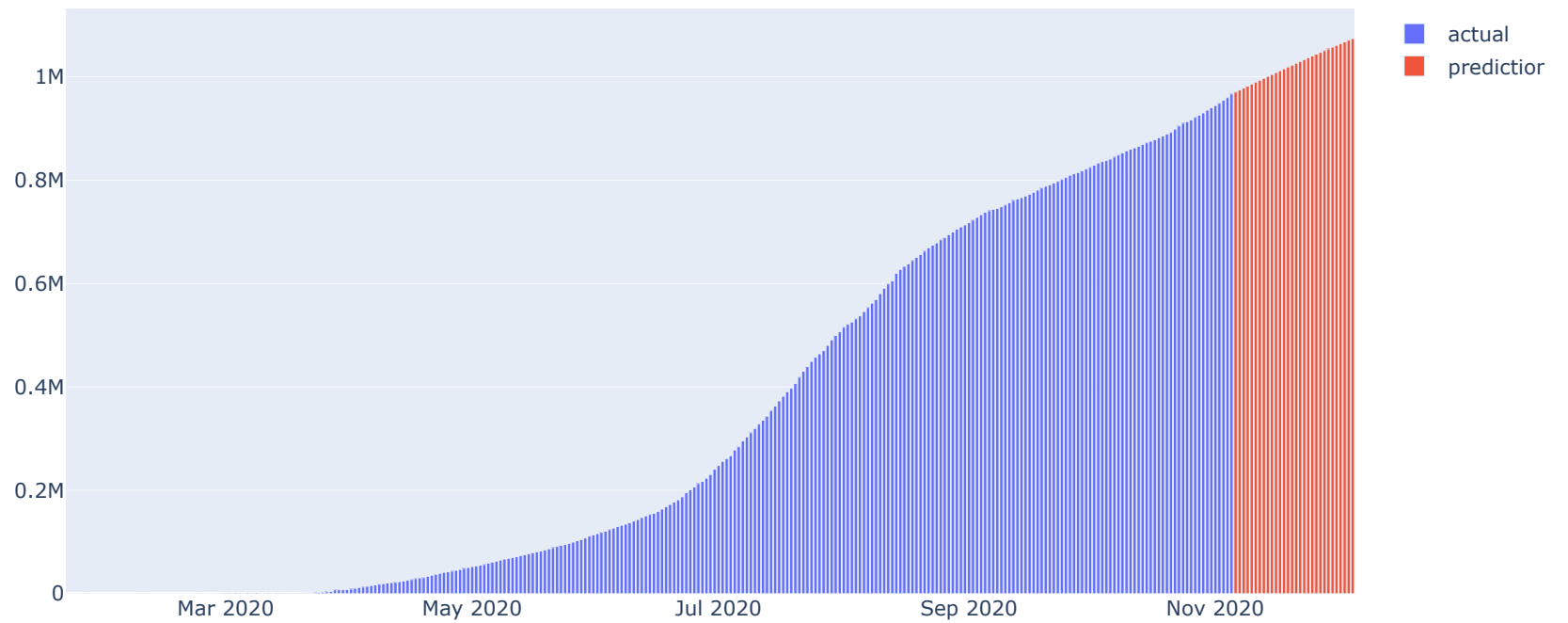
Creating subplots

In [10]:
```python
from plotly.subplots import make_subplots
fig = make_subplots(rows=2, cols=1)

# top subplot
fig.add_scatter(x=df_california_actual.index,
                y=df_california_actual['Deaths'],
                mode="lines+markers",
                name='actual',
                row=1,
                col=1)
fig.add_scatter(x=df_california_pred.index,
                y=df_california_pred['Deaths'],
                mode="lines+markers",
                name='prediction',
                row=1,
                col=1)

# bottom subplot
fig.add_scatter(x=df_california_actual.index,
                y=df_california_actual['Cases'],
                mode="lines+markers",
                name='actual',
                row=2,
                col=1)
fig.add_scatter(x=df_california_pred.index,
                y=df_california_pred['Cases'],
                mode="lines+markers",
                name='prediction',
                row=2,
                col=1)
```
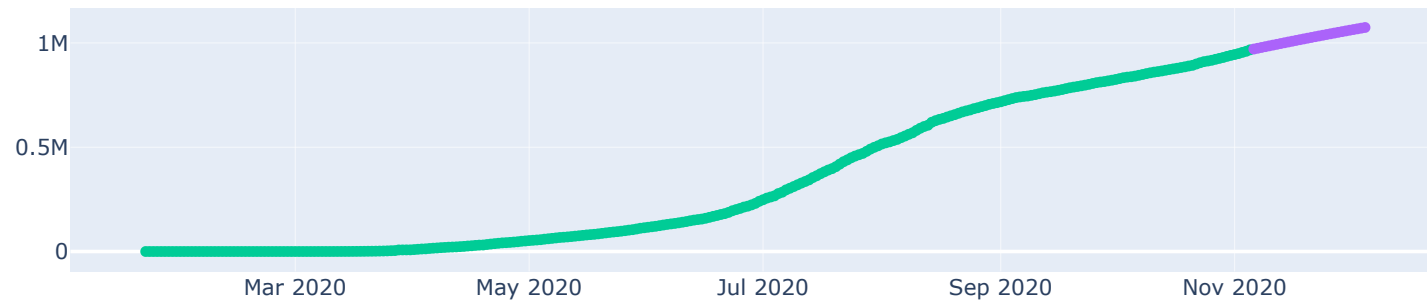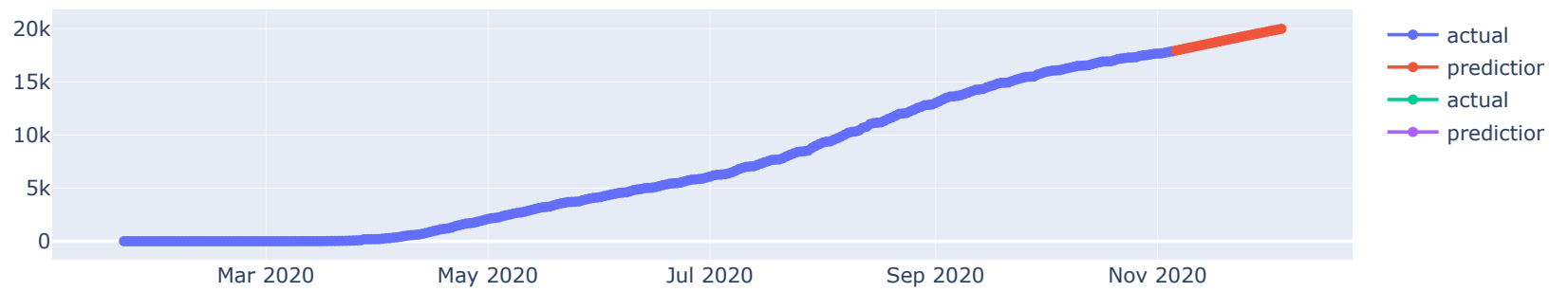
Cleaning up the subplots

In [11]:
```python
from plotly.colors import qualitative
COLORS = qualitative.T10[:2]
KINDS = 'Deaths', 'Cases'
dfs = {'actual': df_california_actual, 'prediction': df_california_pred}

fig = make_subplots(rows=2, cols=1, vertical_spacing=.1)
for row, kind in enumerate(KINDS, start=1):
    for i, (name, df) in enumerate(dfs.items()):
        fig.add_scatter(x=df.index,
                        y=df[kind],
                        mode="lines+markers",
                        name=name,
                        line={"color": COLORS[i]},
                        row=row,
                        col=1)

fig.update_traces(showlegend=False, row=2, col=1)
fig.update_layout(title={"text": "California", "x": 0.5, "y": 0.97, "font": {"size": 20}})
fig
```

California
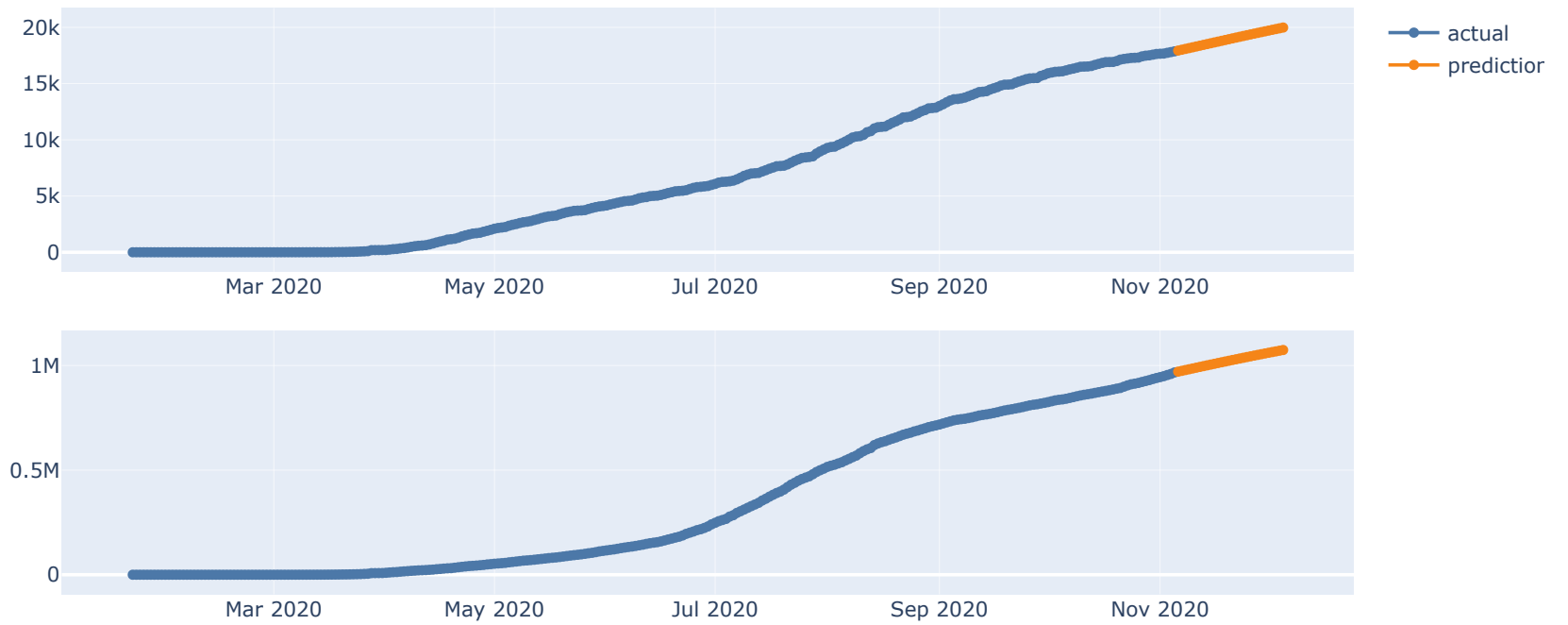
Adding annotations
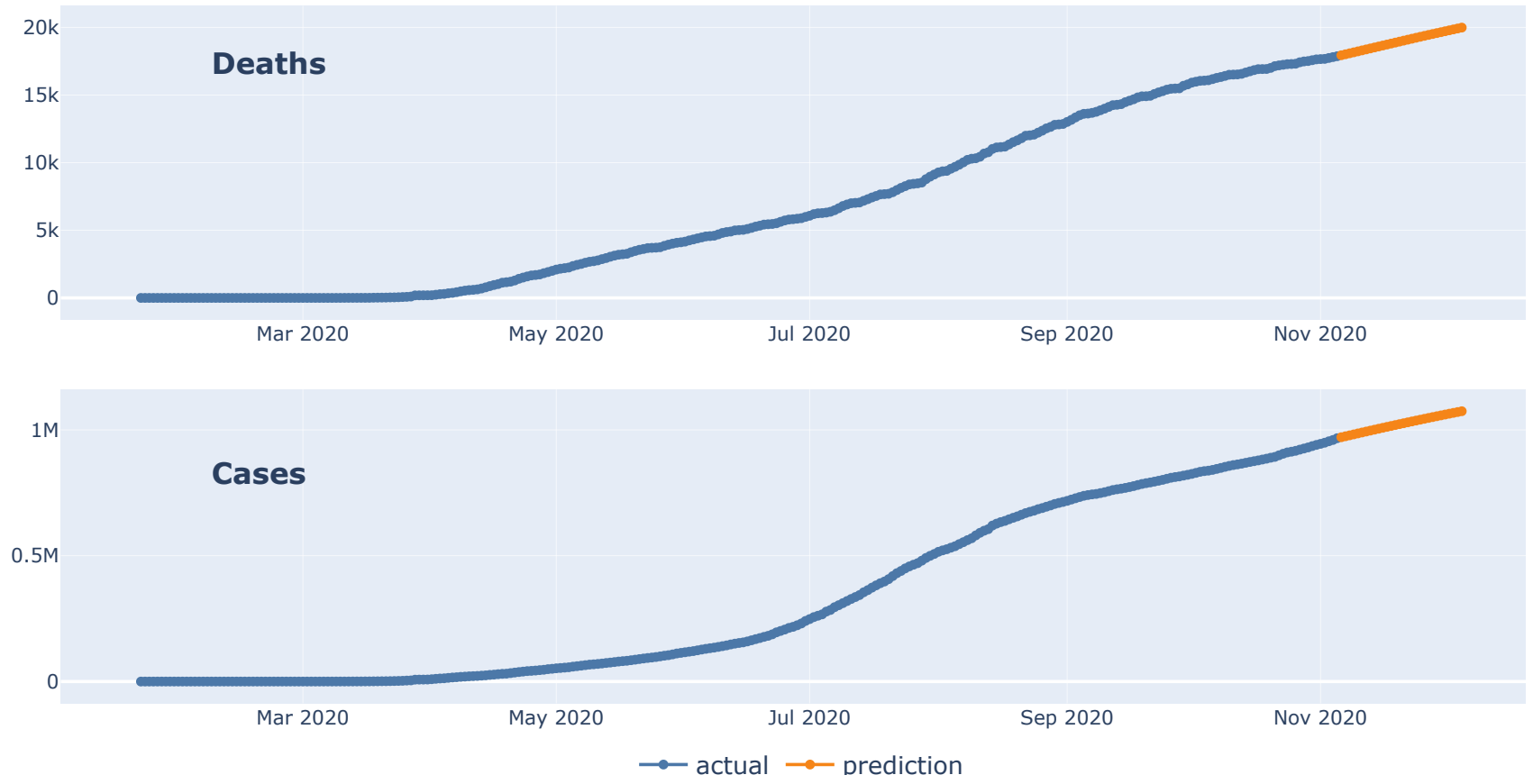
```
In [12]:  fig.update_layout(
              annotations=[
                  {"y": 0.95, "text": "<b>Deaths</b>"},
                  {"y": 0.3, "text": "<b>Cases</b>"},
              ],
              margin={"t": 40, "l": 50, "r": 10, "b": 0},
              legend={
                  "x": 0.5,
                  "y": -0.05,
                  "xanchor": "center",
                  "orientation": "h",
                  "font": {"size": 15}},
          )
          annot_props = {
                  "x": 0.1,
                  "xref": "paper",
                  "yref": "paper",
                  "xanchor": "left",
                  "showarrow": False,
                  "font": {"size": 18},
          }
          fig.update_annotations(annot_props)
          fig
```

California

Deaths

Cases

actual   prediction

Choropleth maps

## Applying Color scale to Counties:

```
In [13]:  df_world = df_summary.query("group == 'world' and population > 1")
          df_world.head(3)
```

Out[13]:

	group	area	Daily Deaths	Daily Cases	Deaths	Cases	code	population	Deaths per Million	Cases per Million	date
0	world	Afghanistan	4	86	1548	41814	AFG	38.928341	40.0	1070.0	2020-11-05
1	world	Albania	7	421	543	22721	ALB	2.877800	189.0	7900.0	2020-11-05
2	world	Algeria	12	642	2011	60169	DZA	43.851043	46.0	1370.0	2020-11-05

```
In [14]:  locations = df_world['code']
          z = df_world['Deaths']
```

```
In [15]: fig = go.Figure()
         fig.add_choropleth(locations=locations, z=z, zmin=0, colorscale="orrd")
         fig.update_layout(margin={"t": 0, "l": 10, "r": 10, "b": 0})
```
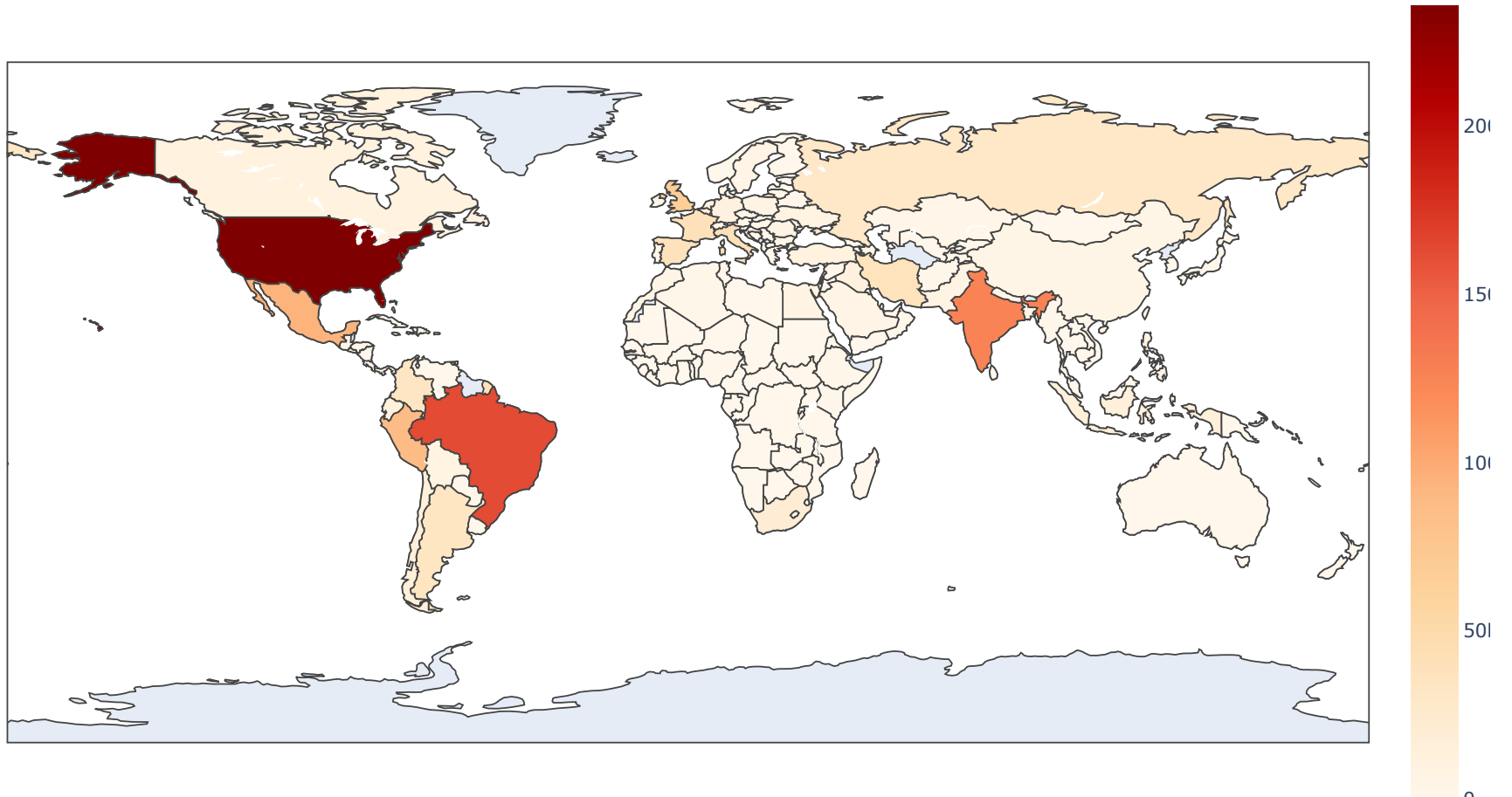
```
In [16]: fig = go.Figure()
         fig.add_choropleth(locations=locations, z=z, zmin=0, colorscale="orrd",  marker_line_width=0.5)
         fig.update_layout(
             geo={
                 "showframe": False,
                 "lataxis": {"range": [-37, 68]},
                 "lonaxis": {"range": [-130, 150]},
                 "projection": {"type": "robinson"}
             },
             margin={"t": 0, "l": 10, "r": 10, "b": 0})
```
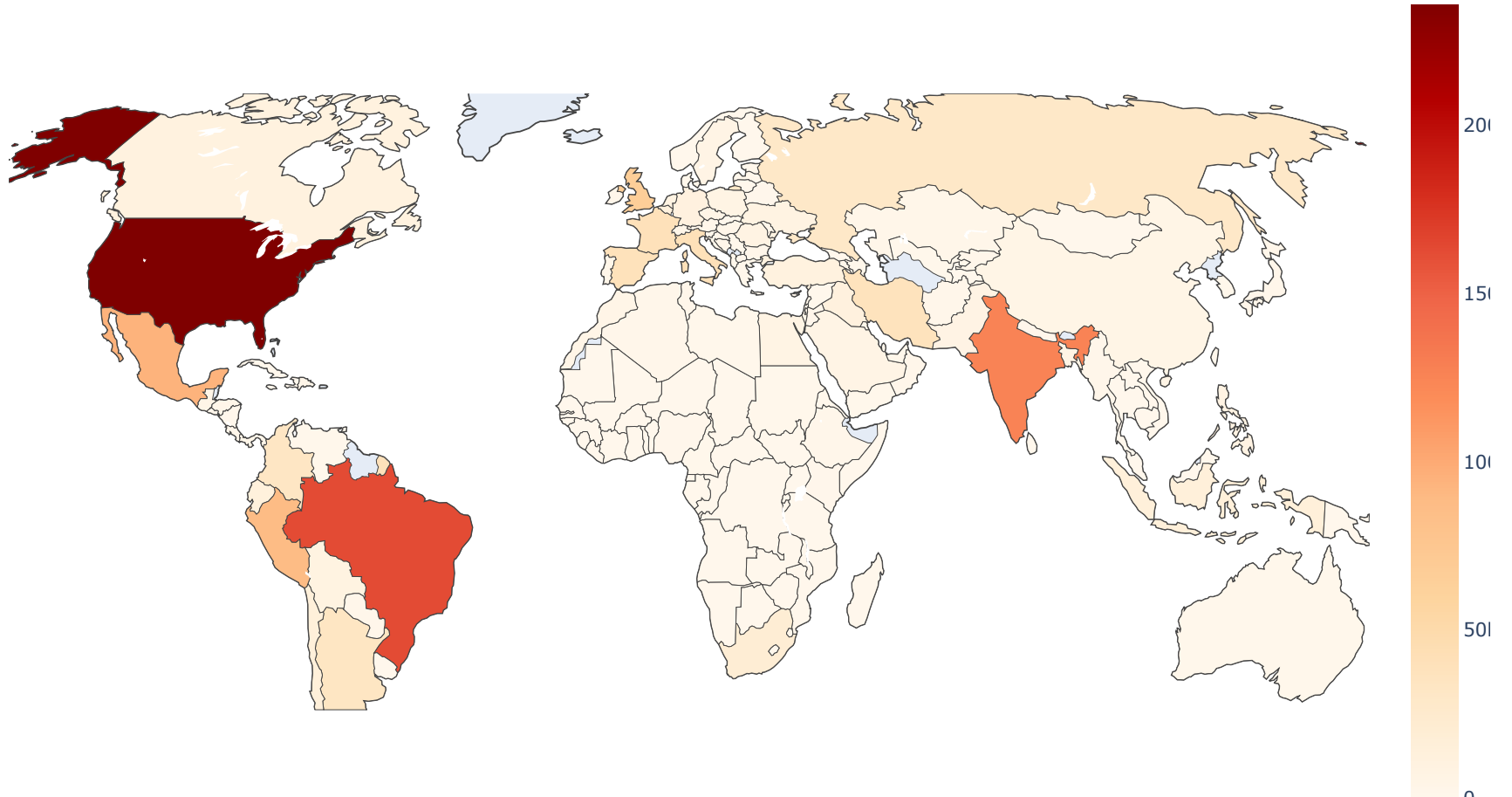
```python
def hover_text(x):
    name = x["area"]
    deaths = x["Deaths"]
    cases = x["Cases"]
    deathsm = x["Deaths per Million"]
    casesm = x["Cases per Million"]
    pop = x["population"]
    return (
        f"<b>{name}</b><br>"
        f"Deaths – {deaths:,.0f}<br>"
        f"Cases – {cases:,.0f}<br>"
        f"Deaths per Million – {deathsm:,.0f}<br>"
        f"Cases per Million – {casesm:,.0f}<br>"
        f"Population – {pop:,.0f}M"
    )


text = df_world.apply(hover_text, axis=1)
text.head()
```

Out[17]:
```
0    <b>Afghanistan</b><br>Deaths – 1,548<br>Cases ...
1    <b>Albania</b><br>Deaths – 543<br>Cases – 22,7...
2    <b>Algeria</b><br>Deaths – 2,011<br>Cases – 60...
4    <b>Angola</b><br>Deaths – 299<br>Cases – 12,10...
7    <b>Argentina</b><br>Deaths – 32,766<br>Cases –...
dtype: object
```

```
In [18]: fig = go.Figure()
         fig.add_choropleth(locations=locations, z=z, zmin=0, colorscale="orrd",
                            marker_line_width=0.5, text=text, hoverinfo="text")
         fig.update_layout(
             geo={
                 "showframe": False,
                 "lataxis": {"range": [-37, 68]},
                 "lonaxis": {"range": [-130, 150]},
                 "projection": {"type": "robinson"}
             },
             margin={"t": 0, "l": 10, "r": 10, "b": 0})
```
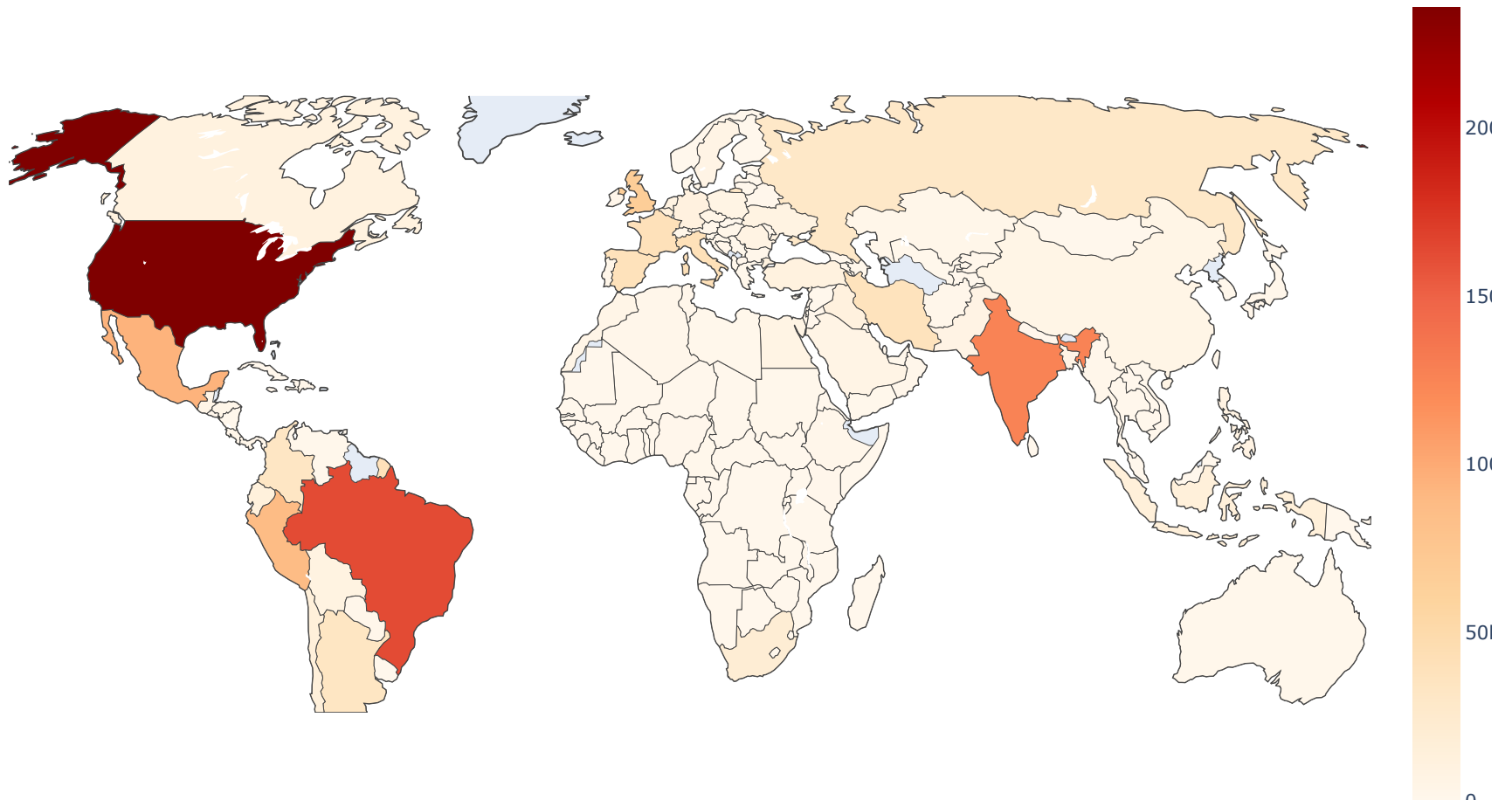
USA Choropleth

```
In [19]: df_states = df_summary.query("group == 'usa'")
         locations = df_states['code']
         z = df_states['Cases per Million']
         text = df_states.apply(hover_text, axis=1)

         fig = go.Figure()
         fig.add_choropleth(locations=locations, locationmode='USA-states', z=z, zmin=0,
                            colorscale="orrd", marker_line_width=0.5, text=text, hoverinfo="text")
         fig.update_layout(
             geo={
                 "showframe": False,
                 "projection": {"type": "albers usa"}
             },
             margin={"t": 0, "l": 10, "r": 10, "b": 0})
```