DSCI 2411 Project 2: COVID Dashboard

Farida Simaika-900201753

Katia Gabriel-900202272

Malak Gaballa-900201683

Masa Tantawy-900201312

Nawal El Boghdady- Instructor

DSCI2411

December 2021

Table of Contents

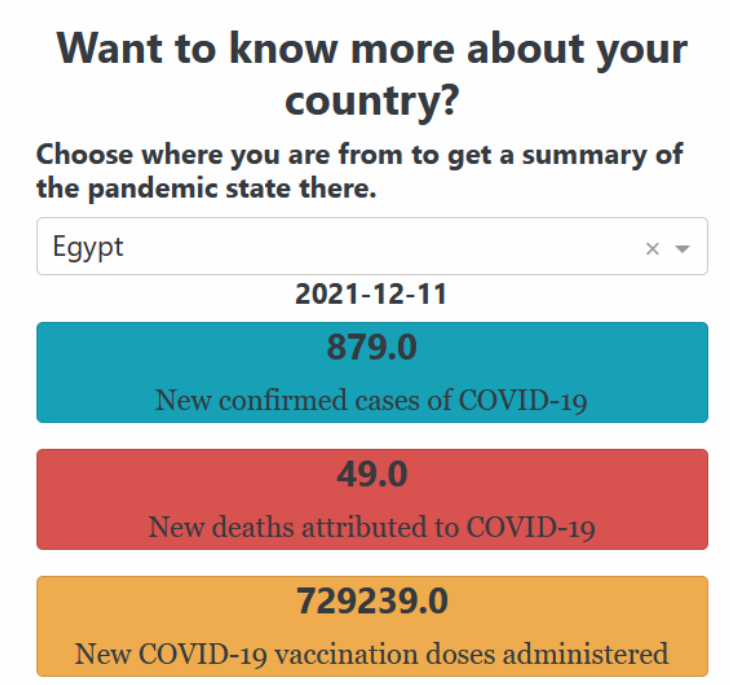
Introduction

Since the start of the COVID- 19 pandemic it arose the need for data collection and data analysis to further understand the pandemic status and trends and monitor the situation regarding lockdowns and vaccination. Enormous amounts of data was collected and stored in datasets. But what better way to bring all of this data to life than through graphs and dashboards.

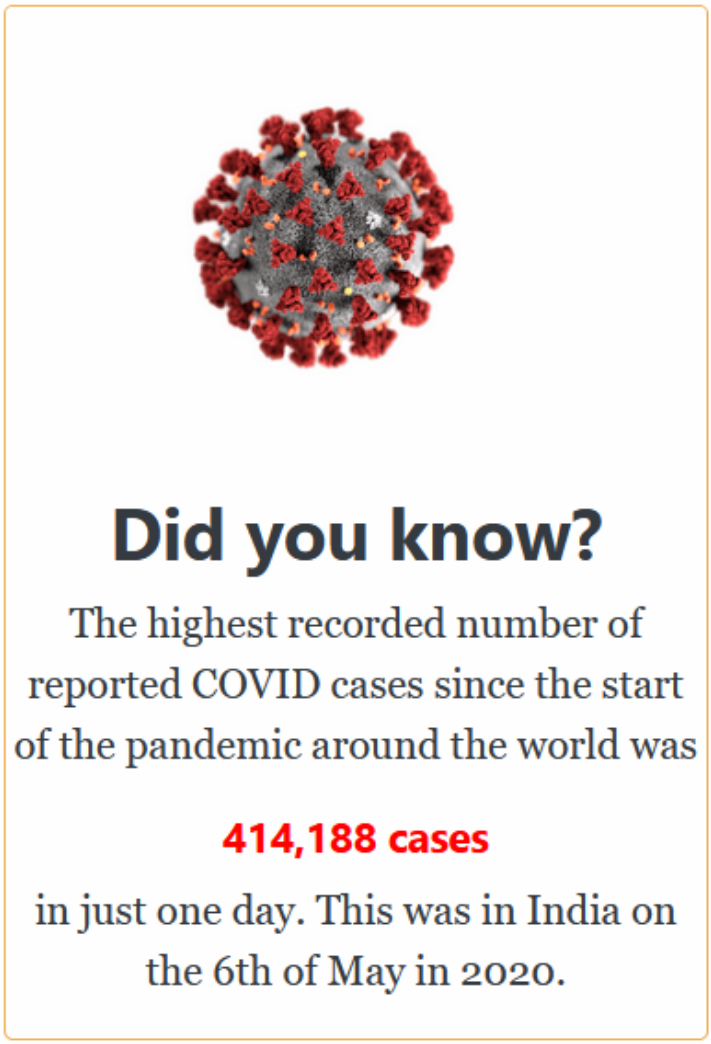
Accordingly, for the second DSCI project, our team had to create a fully-functional COVID dashboard for a country of our choice: The United Kingdom. The data set that was used: “owid-covid-data.csv” contained information regarding COVID-19 mortality, cases, vaccinations, covid-19 ICU admissions, deaths and many more variables. The dashboard contains five figures and numerous stats cards. Each component of the dashboard gives a different perspective of the pandemic. The dashboard’s main aim is to provide the user with an overview of the current pandemic situation in the United Kingdom.

Statistics Cards

The stat cards are the first component that the user sees once the dashboard is loaded. As shown above, our dashboard contains four stats cards that provide information about new COVID-19 cases, deaths, vaccinations and the number of COVID-19 ICU patients in the UK. In just one quick look, the user gets an overview of the pandemic statuts in the UK.



The dashboard contains also three more stats cards providing information about COVID-19 cases, deaths and vaccinations. These three stats cards are connected to a drop down menu that allows the user to choose a country and immediately get the following statistics for the country of their choice: COVID-19 cases, deaths and vaccinations.



Finally, this stats card is an attention grabber that provides the user with a key fact of the pandemic: the highest recorded number of cases in a single day.

Code of the Statistics Cards

```
cards = dbc.Row(
    [
        dbc.Col(dbc.Card([
            html.H5((df_UK['new_cases'][daily - 1:]).to_string(index=False),
                    className="card-title", style={'textAlign': 'center'}),
            html.P(
                "New confirmed cases of COVID-19",
                className="card-text",
                style={'textAlign': 'center'},
            ),
            ], color="info")),

        dbc.Col(dbc.Card([
            html.H5((df_UK['new_deaths'][daily - 1:]).to_string(index=False),
                    className="card-title", style={'textAlign': 'center'}),
            html.P(
                "New deaths attributed to COVID-19",
                className="card-text",
                style={'textAlign': 'center'},
            ),
            ], color="danger")),
```

As shown in the code snippet above, the four stats cards located at the top of the dashboard were created using dash bootstrap components and more precisely dbc.Card,dbc.Row() (they are all on the same row) and dbc.Col(). For the first stats card that shows the number of new COVID-19 cases in the UK, the function [daily -1] was used in order to get the latest number of cases in the UK. Furthermore, the title and color of the stat card were specified and the text was centered and aligned. This process was repeated four times for each stats card and the different variables that they represent.

```
dbc.Row([
    dbc.Col([
        html.H4('Want to know more about your country?'),
        html.H6('Choose where you are from to get a summary of the pandemic state there.'),

        dcc.Dropdown(id="Summary Dropdown",
                     options=[{"label":x,"value":x} for x in pd.unique(df_world.location)],
                    value="Egypt",multi=False
        ),
        html.H6(id='Stat Date',style={'textAlign': 'center'}),
        dbc.Card([
        html.H5(id='Summary1 card',className="card-title", style={'textAlign': 'center'}),
        html.P(
            "New confirmed cases of COVID-19",
            className="card-text",
            style={'textAlign': 'center'},
        ),
        ], color="info", outline=True),
```

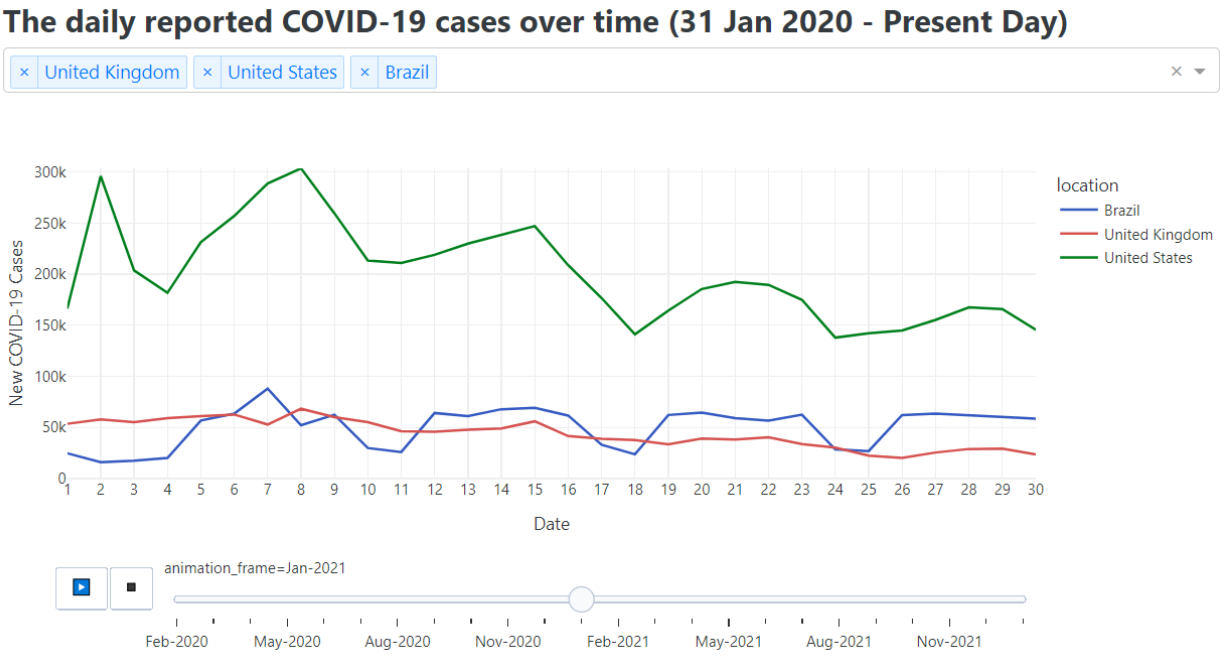
The second batch of stats cards that are connected to a drop down menu were created using the code above. First and foremost the drop down menu was created using dcc.Dropdown(), the id was chosen and a for loop that iterates over every unique location (country) in the column “location” was created. Egypt was also specified as the default value of the drop down menu. Then in order to create the three stats cards, the same process that was described above was repeated three times : the function [daily -1] was used in order to get the latest number of cases /deaths/ vaccines for every unique country in the column “location. Furthermore, the title and color of the stats cards were specified and the text was centered and aligned.

```
    dbc.Col(dbc.Card([
        dbc.CardImg(src=virus_pic, top=True),
        html.H3("Did you know?", style={'textAlign': 'center'}, className="card-title"),
        html.P("The highest recorded number of reported COVID cases since the start of the pandemic around the world was"),
        html.H5("414,188 cases", style={'textAlign': 'center', 'color': 'red', 'fontSize': 18,}),
        html.P("in just one day. This was in India on the 6th of May in 2020.")
    ], color="Light"),
        width={"size":3}, align='center')
]),
```

The picture above shows the code that was used in order to create the “Did you know” stats card. In order to insert the virus picture in the stats card, dbc.CardImg() (a dash bootstrap component) was used. The text was written in an html.P() and finally the color and width were specified.

Figure 1

Figure 1 is a multiple line chart that shows the history of the new reported COVID-19 cases starting from January 2020 until December 2021. The graph compares the history of the United Kingdom, Brazil as well as the United States as the default.

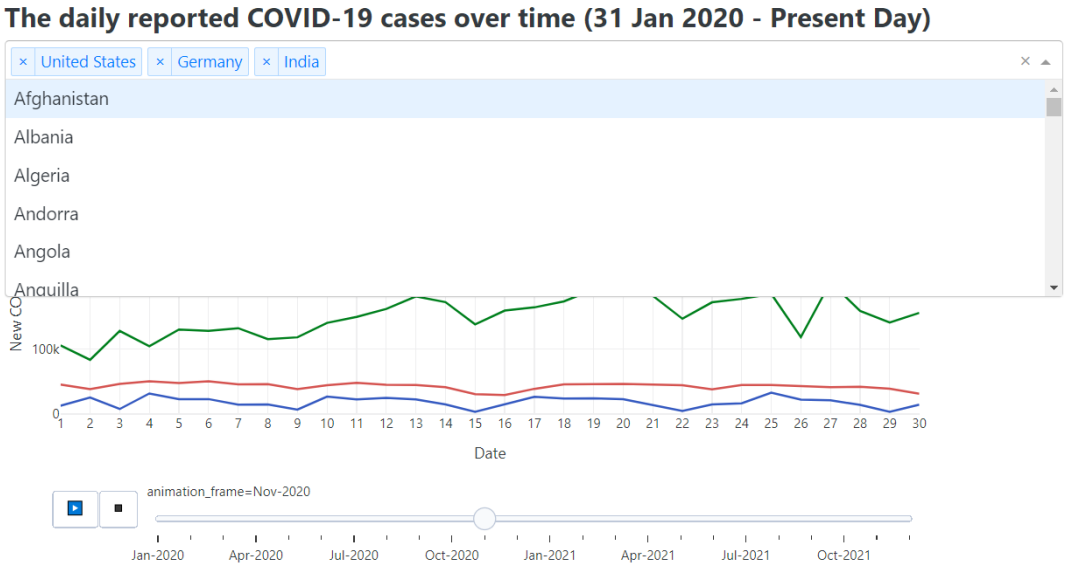


It is noticeable that at the beginning of the year 2020 the newly reported COVID-19 cases were very low, where the new cases were under 50K. With the emergence of time the number of newly reported COVID-19 cases escalated. By comparing the three countries displayed in the graph, it can be seen that the United States is the highest country with newly reported COVID-19 cases. It reached its first peak in January 2021, where the number of newly reported COVID-19 cases exceeded 300K per day, marking the highest COVID-19 spread among all three countries. In September 2021, the United States reached its second peak.

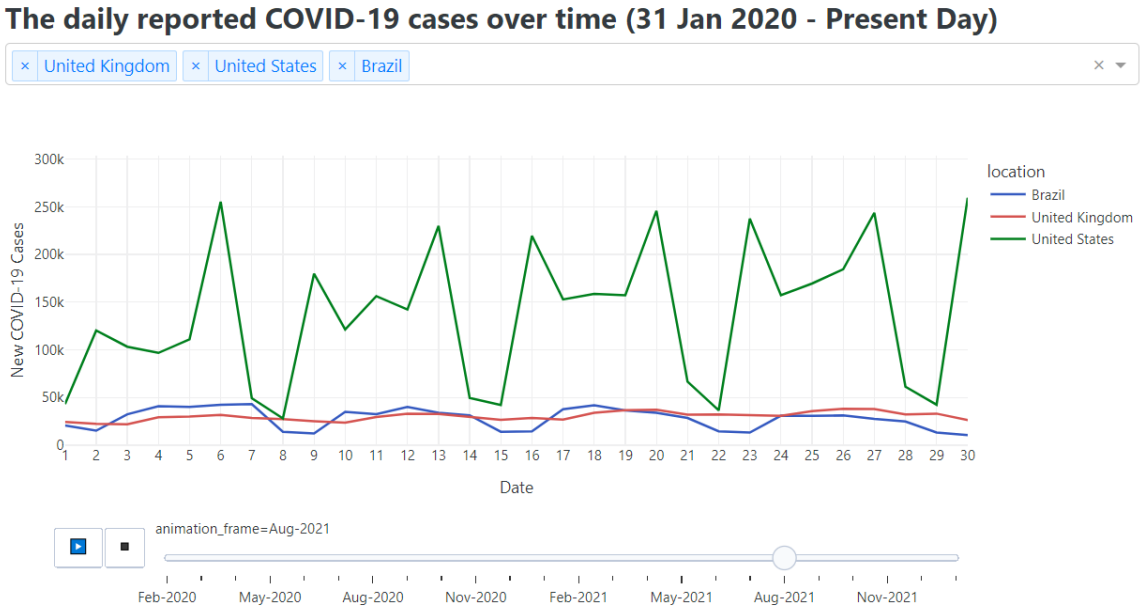
Functionality

This multiple line chart is equipped with a lot of functionalities. The dropdown menu option allows the user to choose the country of his/her choice to display its graph. The user can choose up to three countries to display at once in the graph, which gives the user the chance to compare the newly reported COVID-19 cases of multiple countries of his/her preference. The

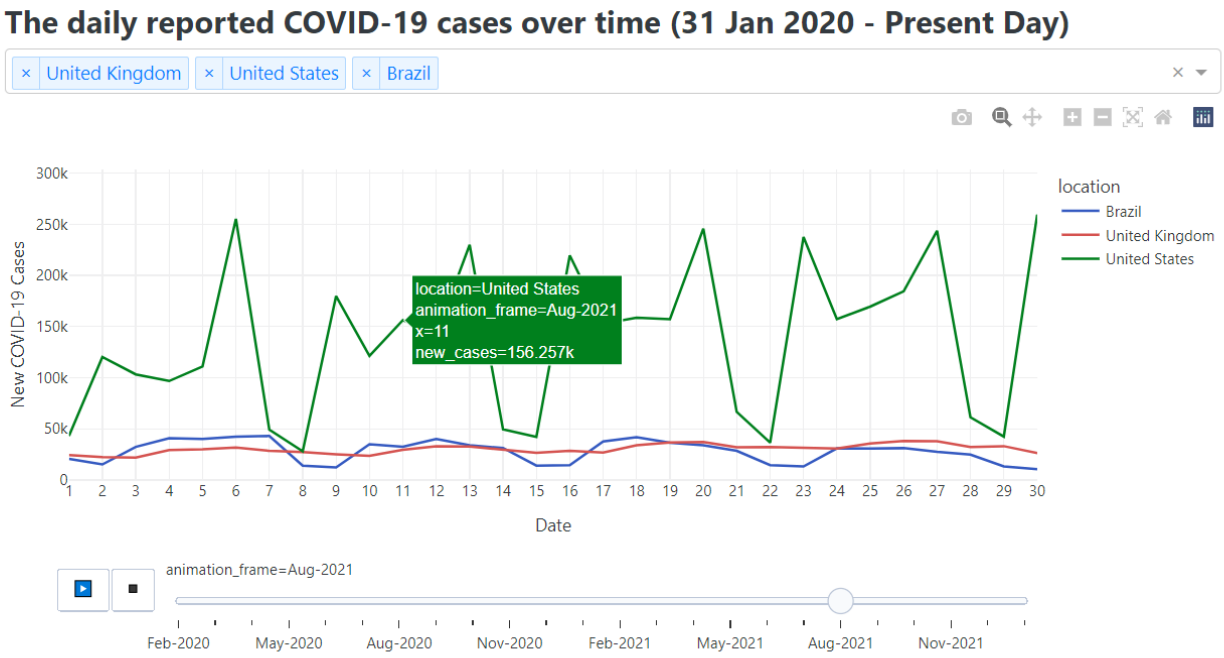
graph's default state compares the history of the progress of newly reported COVID-19 cases of the United States, the United Kingdom and Brazil.



The animation feature gives the user the chance to view the different stages of the evolution of the COVID-19 spread one at a time. The movement of the multiple line graph brings the static data displayed to life, as it displays the change in frequency of the reported COVID-19 cases over time. In addition to the animation feature, the play button allows the user to play or stop the animation at any given point of time to view data of a specific month only.



Moreover, due to the hover feature, the user can hover over the line at any given point in time and can see more detailed information, such as location (or country name), date, the exact number of new cases as listed in the dataset.



Code

```
dbc.Row([html.Div([cards])]),
dbc.Row([
    dbc.Col([html.Div(children=[
      html.Div(children="History of the evolution of the new corona cases around the world"),

      dcc.Dropdown(id="mydropdown",
     options=[{"label": x, "value": x}
     for x in pd.unique(df_world.location)],
      value=["United Kingdom","United States","Brazil"],
      multi=True
 ),
      dcc.Graph(id='Fig1', figure={})
      ]),

    ],
        width={"size":9}),
```

In the above code snippet the figure was given a location in the dashboard grid and its title was set. The dropdown menu was created in the app's layout using dash core components. The options for the dropdown menu were set using a for loop that iterates over the unique

countries in the dataset. The default values of the dropdown menu were set in the list and are the United States, the United Kingdom and Brazil. Accordingly the dropdown menu was set to accept up to three selections of different countries.

```
######### Figure 1 callback
@app.callback(
    Output("Fig1", "figure"),
    [Input("mydropdown", "value")])

def update_fig(value):
    df_subset = df_world.loc[df_world["location"].isin(value)]

    fig1 = px.line(df_subset, x=df_subset["date"].dt.day, y="new_cases", color='location',
                   range_y=[0, df_subset['new_cases'].max()], range_x=[df_subset.date.iloc[0], df_subset.date.iloc[-1]],
                   animation_frame=df_subset['date'].dt.strftime("%b-%Y"), animation_group='location')
    #fig.update_xaxes(title='Date', dtick="M1", tickformat="%b\n%Y")
    fig1.update_xaxes(title='Date', range=[1, 30], dtick="M1")
    fig1.update_yaxes(title='New COVID-19 Cases')
    return fig1
```

This code snippet shows the callback function that was created to display the multiple line chart. The app.callback works in the following fashion. It takes multiple values from the dropdown menu and overlays its respective figures. First, a subset of the dataset was created that contains the locations that are in the “value” argument set above. Afterwards, the multiple line graph was created using “plotly.express”, displaying the months against the number of new cases. The color argument was set to be the location, so that the line graph of each country could have a different color.

To animate the line graph, four additional arguments were specified. The range of y was set to start from zero and end at the overall maximum number of new cases, while the x ranges from the starting date of data collection to the present day. The animation frame, sets the dates for the play bar and the animation group, animates the line graph of the countries. Lastly, the y-axis and the x-axis.

Figure 2

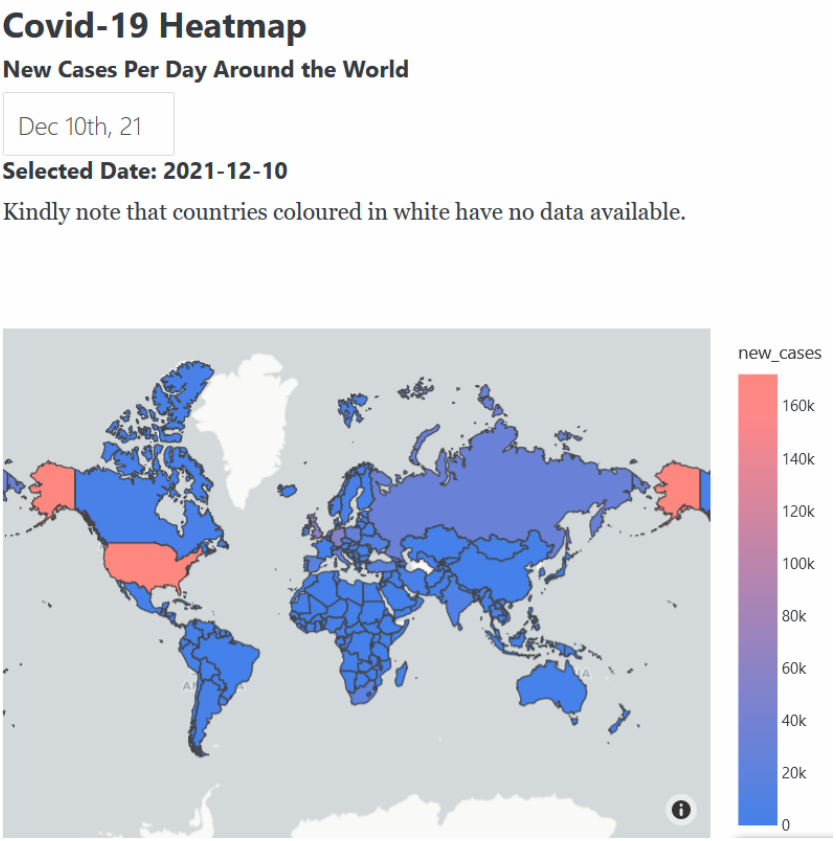
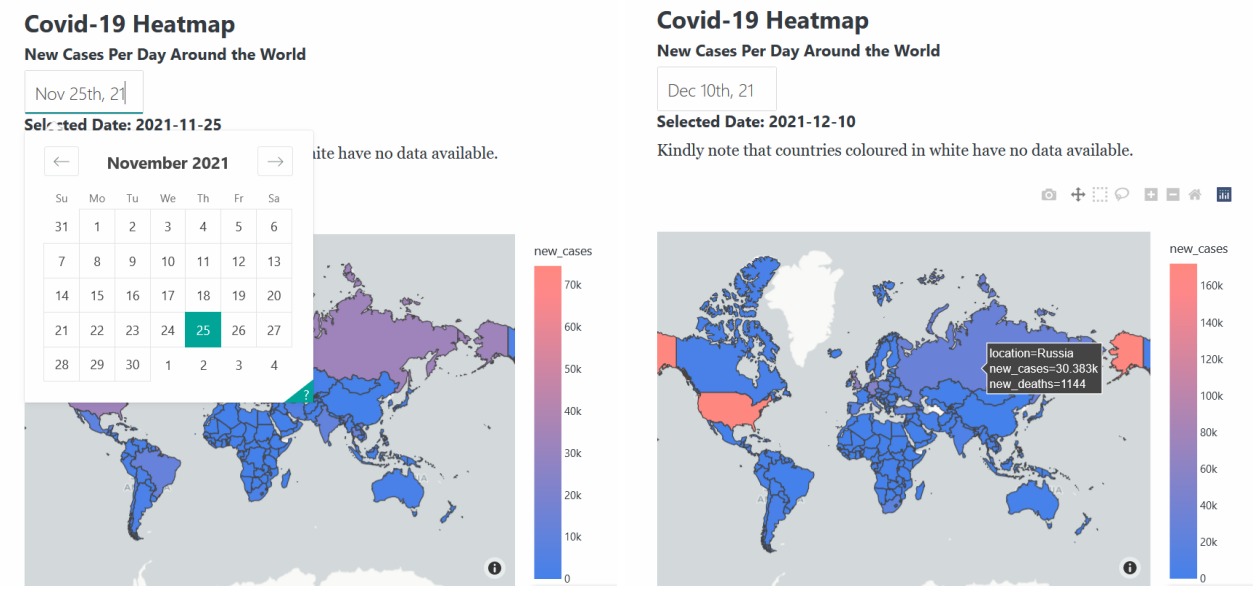


Figure 2 is a world map that shows the number of reported new COVID-19 cases for each country around the world on a specific date. It is conspicuous that the number of cases differ in certain areas compared to others. For instance, all the countries in Africa are coloured blue on most of the days, indicating that the pandemic state there is not very severe. On the other hand, the United States has the highest number of new cases on all the days hence it has an orange colour. Other countries such as Russia and the UK have a moderate number of new cases per day.

A continuous colour scale is used for the map to indicate the number of new cases in each country around the world. This colour scale starts at zero and ends at the maximum number of cases (the number of new cases in the highest country in the world). Since data is not available for the countries, certain countries are coloured in white. A note is written at the top of the graph to inform the viewer that countries where the data is not present are white.

Functionalities



This world map allows the user to select the date which they would like to explore the number of new cases around the world on. The date selected is displayed above the graph when the app updates. The last date allowed is the last date for which data is available in the database, which is the date in the last row. Likewise, the first date allowed is the one for which data is available- first row's date. The map's default shows the latest data available; it displays the number of cases on the last day in the database.

Another functionality that is present in this figure is the hover information. The viewer can hover over any country in the world where the data is known to know more about the pandemic situation there. The hover data that is displayed are: the name of the country, the number of new cases, and the number of new deaths on the chosen date.

Code

```
######### Figure 2 callback
@app.callback(
    [Output('Fig2', 'figure')],
    [Output('Printed Date', 'children')],
    [Input('date-picker-single', 'date')])

def map_covid(date_value):
    # Selecting data only for the specified date
    filtered_df = df_world[df_world.date == date_value]
    fig2 = px.choropleth_mapbox(filtered_df, color='new_cases', geojson=geojson, locations='iso_code', featureidkey='properties.iso_a3',
                                center={'lat': 40.52, 'lon': 34.34}, zoom=0, mapbox_style='carto-positron',
                                range_color=(0, filtered_df['new_cases'].max()), color_continuous_midpoint=40000,
                                title='New Cases Per Day Around the World',
                                hover_data={'location':True, 'new_cases':True, 'iso_code': False, 'new_deaths':True})

    return (fig2,'Selected Date: {}'.format(date_value))
```

The code snippet above depicts the callback function used to create the world map. First, a subset of the data for the selected date is extracted. Next, a choropleth mapbox is drawn, where the colour is the number of new cases, and the countries are matched with the geojson file that contains the exact coordinates (longitude and latitude) of each country via the iso code column in the data frame which is the same as the iso_a3 property in the geojson file. Lastly, the other options are set, such as the range of the colour scale, the title, and the hover data. The callback function returns the figure and the chosen date to be displayed.

```
dbc.Col([html.Div(children=[
    html.H4('Figure 2'),
    html.Div(children="Covid-19 Heatmap"),

    dcc.DatePickerSingle(id="date-picker-single",
                         min_date_allowed=date(2020, 2, 24),
                         max_date_allowed=df_world.date.iloc[-1],
                         date=df_world.date.iloc[-1],
                         display_format='MMM Do, YY', ),
    html.H6(id='Printed Date'),
    html.P("Kindly note that countries coloured in white have no data available."),
    dcc.Graph(id='Fig2'),

])], width={"size": 6})
```

In the app's layout, the datepicker is created allowing the viewer to select only 1 date to view the map. As explained earlier, the maximum date allowed and the initial value are set

according to the last row in the dataframe, and the minimum date allowed is the oldest date in the dataframe. A note for the user is displayed after the calendar then the graph is shown.

Figure 3

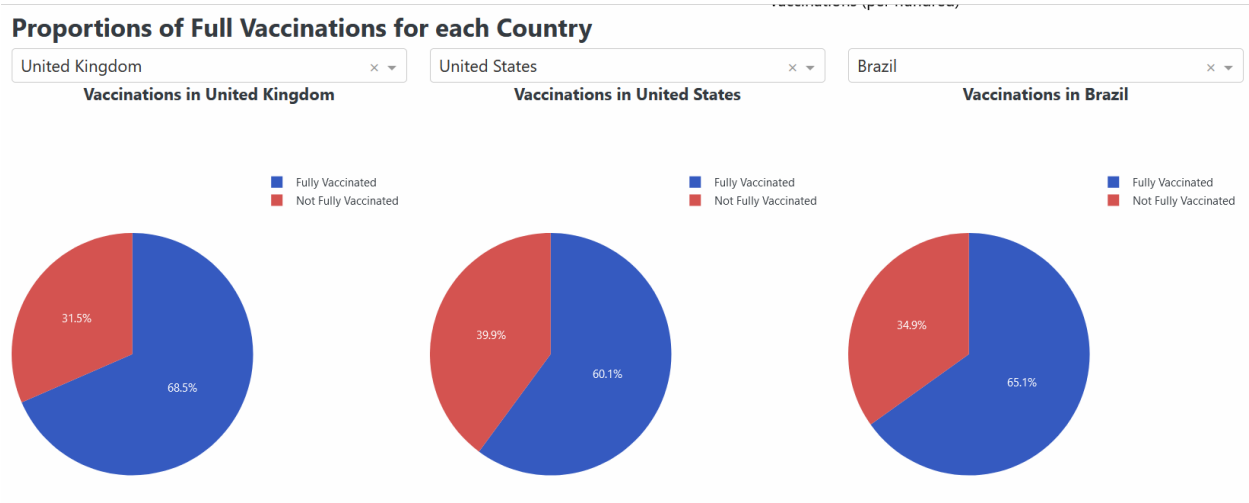


Figure 3 is a group of pie charts that represents the proportion of fully vaccinated people to those not fully vaccinated by percentage in a given country. There are three pie charts for comparison; United Kingdom, United States and Brazil. This figure was done using pie charts because they easily visualize and compare proportions.

Functionalities

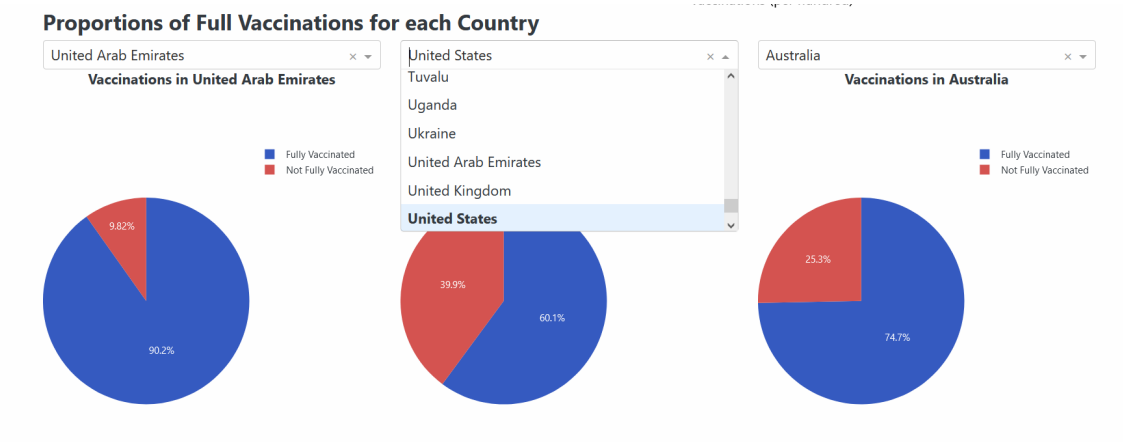


Figure 3 has several pie charts for comparison between the percentage of fully vaccinated people in a given country. Therefore, a dropdown menu has been created to make the user compare the proportion between countries of his/her choice. To be more specific , a dropdown

menu is added to each pie chart so that the user can easily change between the countries and compare the differences between the percentages presented.

Code

```
# UK
df_UK['people_fully_vaccinated'] = df_UK['people_fully_vaccinated'].fillna(method='ffill')
UK_vaccine_percentage = df_UK['people_fully_vaccinated'][df_UK.index[-1]] / df_UK['population'][df_UK.index[-1]]
UK_no_vaccine = 1 - UK_vaccine_percentage

UK_total = [['Fully Vaccinated', UK_vaccine_percentage], ['Not Fully Vaccinated', UK_no_vaccine]]

UK_smalldf = pd.DataFrame(UK_total, columns=['Type', 'Proportion'])

fig3a = px.pie(UK_smalldf, values='Proportion',
               title='United Kingdom Full Vaccination Proportion ', names='Type')

# USA
df_US['people_fully_vaccinated'] = df_US['people_fully_vaccinated'].fillna(method='ffill')
US_vaccine_percentage = df_US['people_fully_vaccinated'][df_US.index[-1]] / df_US['population'][df_US.index[-1]]
US_no_vaccine = 1 - US_vaccine_percentage

US_total = [['Fully Vaccinated', US_vaccine_percentage], ['Not Fully Vaccinated', US_no_vaccine]]

US_smalldf = pd.DataFrame(US_total, columns=['Type', 'Proportion'])

fig3b = px.pie(US_smalldf, values='Proportion',
               title='United Kingdom Full Vaccination Proportion ', names='Type')
# Brazil
df_Brazil['people_fully_vaccinated'] = df_Brazil['people_fully_vaccinated'].fillna(method='ffill')
B_vaccine_percentage = df_Brazil['people_fully_vaccinated'][df_Brazil.index[-1]] / df_Brazil['population'][
    df_Brazil.index[-1]]
B_no_vaccine = 1 - B_vaccine_percentage

B_total = [['Fully Vaccinated', B_vaccine_percentage], ['Not Fully Vaccinated', B_no_vaccine]]

B_smalldf = pd.DataFrame(B_total, columns=['Type', 'Proportion'])

fig3c = px.pie(B_smalldf, values='Proportion',
               title='United Kingdom Full Vaccination Proportion ', names='Type')
```

As shown in the code above, three pie charts for different countries were constructed using the same line of code for each chart. First, the ‘people_fully_vaccinated’ column in each country-specific data frame was modified in a way that any missing value is replaced by the last observed non-null value. This is done since the ‘people_fully_vaccinated’ column represents the cumulative number of fully vaccinated people in a country. Afterwards, the proportion of full vaccinations has been calculated by dividing the latest number of full vaccinations by the population and stored in a variable (*countryname*_vaccine_percentage).Consequently, the proportion of those not fully vaccinated has been calculated and stored in another variable (*countryname*_no_vaccine). To be able to plot a pie chart, a data frame was created with both variables. Finally , a pie chart was constructed. Please note that this explained code has been implemented for all three pie charts in the same way regardless of the country.

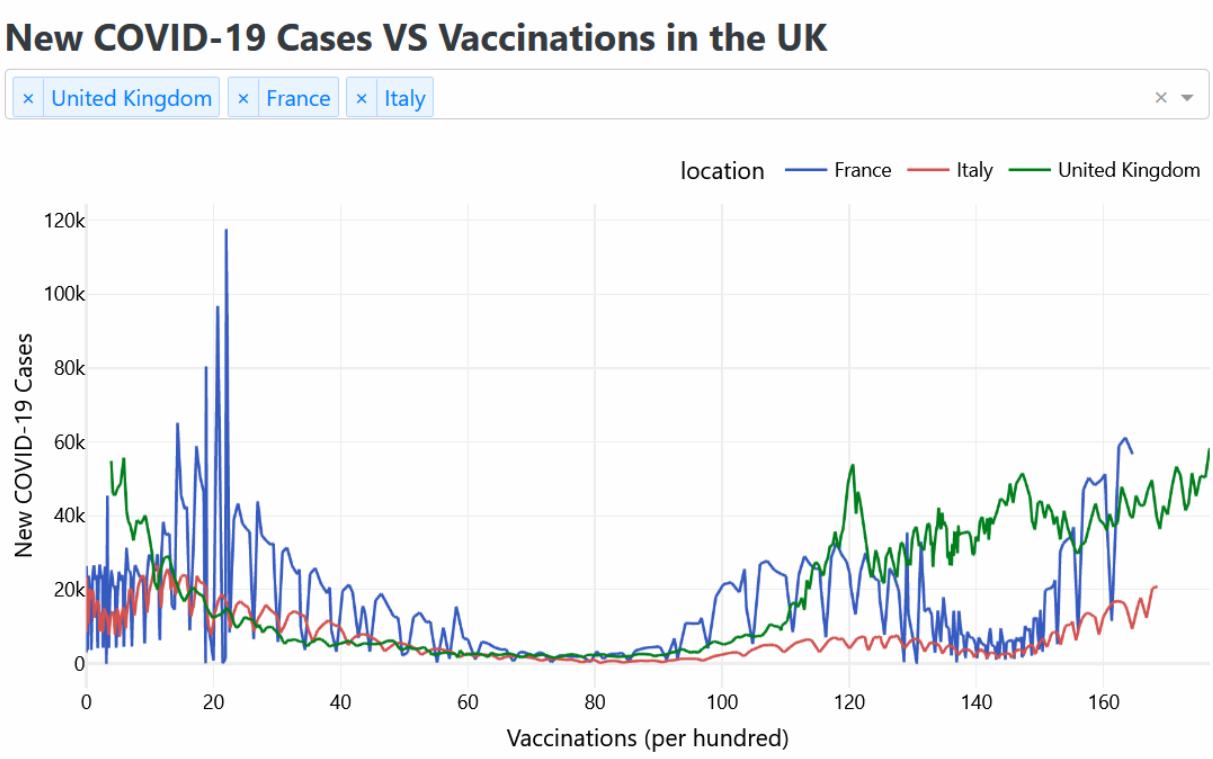
```
    dbc.Row([
        dbc.Col([html.Div(children=[
         html.H4('Figure 3'),
            dbc.Row([
    dbc.Col([
        dcc.Dropdown(id="f3dropdown1",
                     options=[{"label": x, "value": x}
                              for x in pd.unique(df_world.location)],
                     value="United Kingdom",multi=False),
        html.H6(id='3a country', style={'textAlign': 'center'}),
        dcc.Graph(id='Fig3a', figure={})
    ],
    width=4),
    dbc.Col([
       dcc.Dropdown(id="f3dropdown2",
                     options=[{"label": x, "value": x}
                              for x in pd.unique(df_world.location)],
                     value="United States", multi=False),
        html.H6(id='3b country', style={'textAlign': 'center'}),
        dcc.Graph(id='Fig3b', figure={})
    ],
    width=4),
    dbc.Col([
        dbc.Row([dcc.Dropdown(id="f3dropdown3",
                     options=[{"label": x, "value": x}
                              for x in pd.unique(df_world.location)],
                     value="Brazil", multi=False)]),
        dbc.Row([
            html.H6(id='3c country', style={'textAlign': 'center'}),
            dcc.Graph(id='Fig3c', figure={})])
    ],
    width=4),
])
        ],)
    ]),
```

Moving onto the app layout, dash bootstrap components were used to create a row containing three columns in the dashboard and each column will contain a pie chart and a dropdown menu. The ID of the drop down menus were specified and their functionalities were created using a for loop that will iterate over every unique “location” (country) in the data set. The default values were set to the United Kingdom, United States and Brazil respectively. For these dropdowns, the user is only allowed to choose one value or country per dropdown since the graph is a pie chart, hence adding the argument *multi=False*.

```python
## 3a
@app.callback(
    [Output("Fig3a", "figure")],
    [Output('3a country', 'children')],
    [Input("f3dropdown1", "value")])
def proportion_pie(value):
    df_subvaccine = df_world[df_world["location"] == value]
    df_subvaccine['people_fully_vaccinated'] = df_subvaccine['people_fully_vaccinated'].fillna(method='ffill')
    vaccine_percentage = df_subvaccine['people_fully_vaccinated'][df_subvaccine.index[-1]] / df_subvaccine['population'][df_subvaccine.index[-1]]
    no_vaccine = 1 - vaccine_percentage

    total = [['Fully Vaccinated', vaccine_percentage], ['Not Fully Vaccinated', no_vaccine]]
    smalldf = pd.DataFrame(total, columns=['Type', 'Proportion'])
    fig3a = px.pie(smalldf, values='Proportion', names="Type")
    return (fig3a, 'Vaccinations in {}'.format(value))
##3b
@app.callback(
    [Output("Fig3b", "figure")],
    [Output('3b country', 'children')],
    [Input("f3dropdown2", "value")])
def proportion_pie(value):
    df_subvaccine = df_world[df_world["location"] == value]
    df_subvaccine['people_fully_vaccinated'] = df_subvaccine['people_fully_vaccinated'].fillna(method='ffill')
    vaccine_percentage = df_subvaccine['people_fully_vaccinated'][df_subvaccine.index[-1]] / df_subvaccine['population'][df_subvaccine.index[-1]]
    no_vaccine = 1 - vaccine_percentage


    total = [['Fully Vaccinated', vaccine_percentage], ['Not Fully Vaccinated', no_vaccine]]
    smalldf = pd.DataFrame(total, columns=['Type', 'Proportion'])
    fig3b = px.pie(smalldf, values='Proportion', names='Type')
    return (fig3b, 'Vaccinations in {}'.format(value))
##3c
@app.callback(
    Output("Fig3c", "figure"),
    [Output('3c country', 'children')],
    [Input("f3dropdown3", "value")])
def proportion_pie(value):
    df_subvaccine = df_world[df_world["location"] == value]
    df_subvaccine['people_fully_vaccinated'] = df_subvaccine['people_fully_vaccinated'].fillna(method='ffill')
    vaccine_percentage = df_subvaccine['people_fully_vaccinated'][df_subvaccine.index[-1]] / df_subvaccine['population'][df_subvaccine.index[-1]]
    no_vaccine = 1 - vaccine_percentage


    total = [['Fully Vaccinated', vaccine_percentage], ['Not Fully Vaccinated', no_vaccine]]
    smalldf = pd.DataFrame(total, columns=['Type', 'Proportion'])
    fig3c = px.pie(smalldf, values='Proportion', names='Type')
    return (fig3c, 'Vaccinations in {}'.format(value))
```

Finally, three callback functions ,one for each pie chart or figure, were added to allow us to specify the functionalities of the dropdown menus that were created. In the function *proportion_pie(value)* for each figure, the same lines of code that were used when creating the default pie charts were used in this function as well with the only difference that the callback function takes a subset of the data where ‘location’ is equal to a certain value or in this case a certain country. Then the typical lines of code were used to generate the proportions and consequently the pie charts.

Figure 4



The figure above is a multiple line chart that shows the relationship between new covid-19 cases and vaccinations. As shown above, there seems to be no correlation between the two variables. Covid-19 cases seem to fluctuate independently. The increase in vaccine doses administered has no subsequent effect on new covid-19 cases. However, this does not mean that the covid-19 vaccines are ineffective. Vaccines do not prevent a person from getting infected with the virus, they are highly effective against severe illness and reduce hospitalization rates.

Functionalities

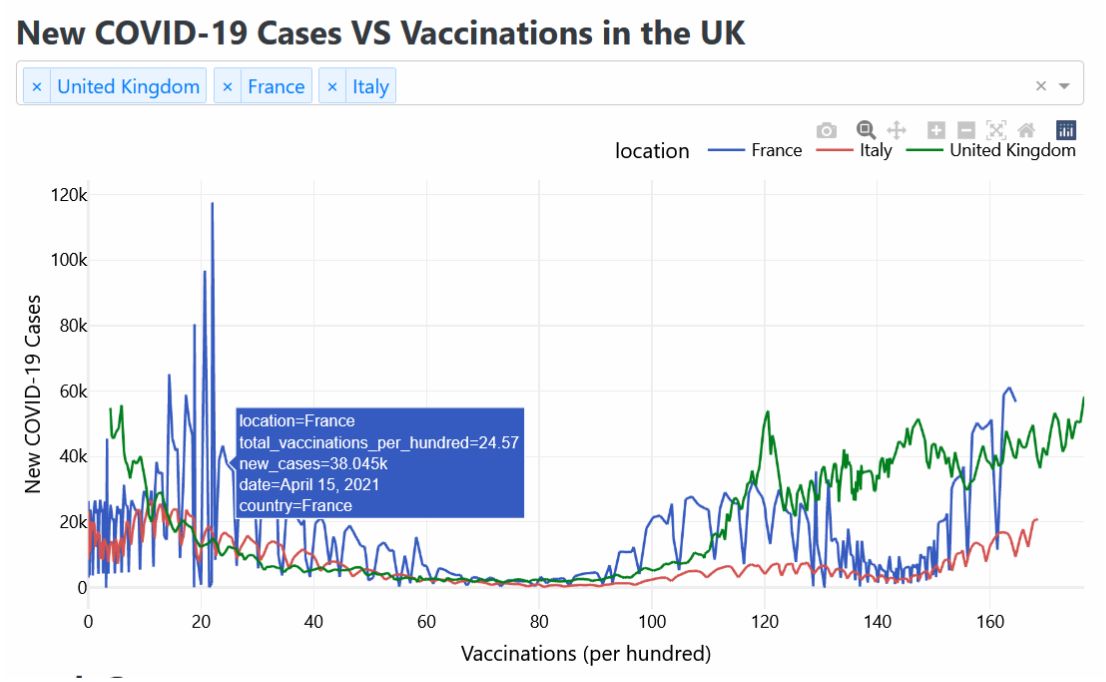
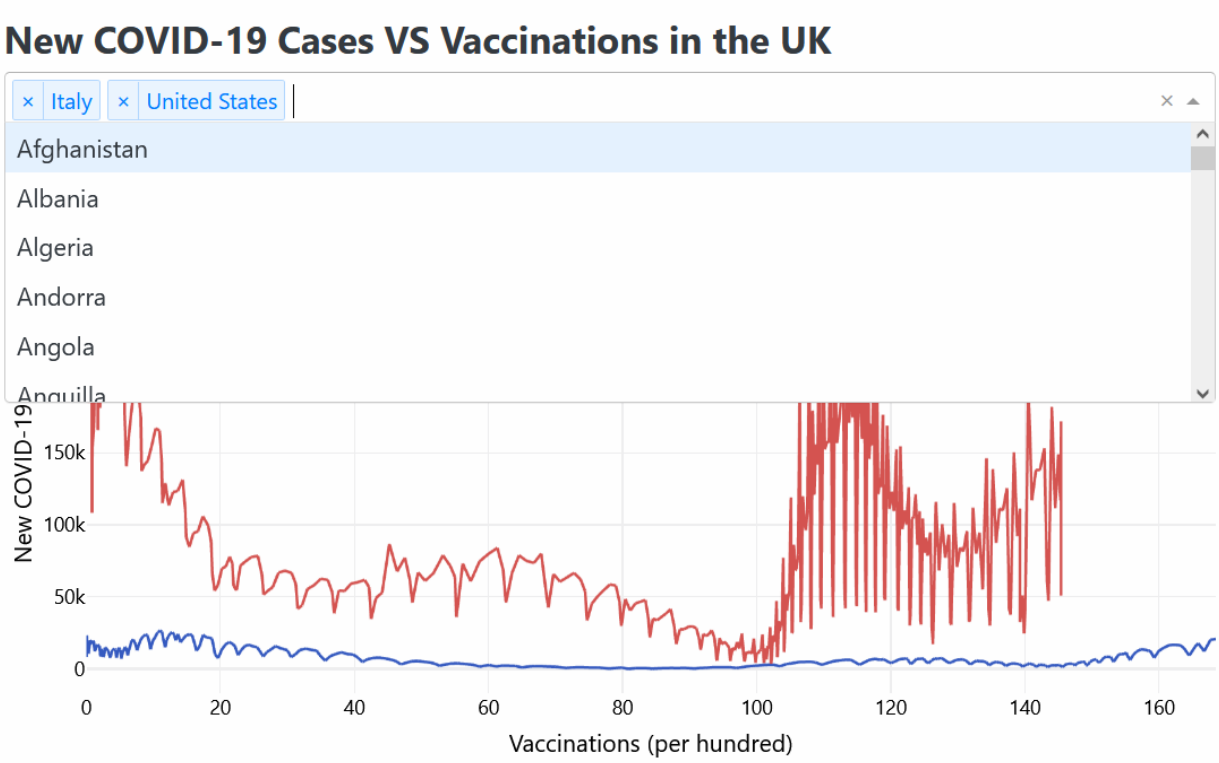


Figure 4 is equipped with many functionalities for a given user. The latter can hover over any line in the graph and will get the following detailed information: the location (and country name), number of new cases, vaccinations and the date from which both variables were recorded.

In addition, as shown below, there is a drop down menu that allows the user to select the country/countries of his choice. There are also 3 default countries: the United Kingdom, France and Italy. Those default countries are removable.



[Code]

```
df_UK = df_UK[df_UK["new_cases"] >= 0]

fig4 = px.line(df_UK, x="total_vaccinations_per_hundred", y="new_cases",
               title="New COVID-19 Cases VS Vaccinations in the UK",
               hover_data={"date": "|%B %d, %Y",
                           "country": df_UK['location']})

fig4.update_traces(marker=dict(size=13,
                               line=dict(width=1,
                                         color="#E05194")),
                   selector=dict(mode='markers'))

# modifying legend position and scatter plot dimensions
fig4.layout = go.Layout(
    plot_bgcolor="#FFF",  # Sets background color to white
    xaxis=dict(  # Sets color of X-axis line
        showgrid=False  # Removes X-axis grid lines
```

As shown in the code snippet above, figure 4 was created using plotly express and more specifically px.line() . Plotly Graph Object was also used to set up the layout, title fonts, colors and background.

```
dbc.Row([
    dbc.Col([html.Div(children=[
        html.H4("Figure 4"),
        html.Div(children="COVID-19 and vaccines"),

        dcc.Dropdown(id="fig4dropdown",
                     options=[{"label":x,"value":x}
                     for x in pd.unique(df_world.location)],
                     value=["United Kingdom","France","Italy"],
                     multi=True
    ),
        dcc.Graph(id="Fig4",figure={}),
    ])
     ],width={"size":6})
```

Then, dash bootstrap components were used to create a row in the dashboard that will contain figure 4 and a drop down menu. The ID of the drop down menu was specified and its functionalities were created using a for loop that will iterate over every unique “location” (country) in the data set. The default values were also set using the for loop.

```
@app.callback(
    Output("Fig4","figure"),
    [Input("fig4dropdown", "value")])
def fig4_update(value):
    df_subset=df_world.loc[df_world["location"].isin(value)]
    df_subset = df_subset[df_subset["new_cases"] >= 0]

    fig4 = px.line(df_subset, x="total_vaccinations_per_hundred", y="new_cases",color="location",
                   title="New COVID-19 Cases VS Vaccinations in the UK",
                   hover_data={"date": "|%B %d, %Y",
                               "country": df_subset['location']})
    return fig4
```

Finally, a callback for figure 4 was added to allow us to specify the functionality of the dropdown menu that was just created.

Figure 5

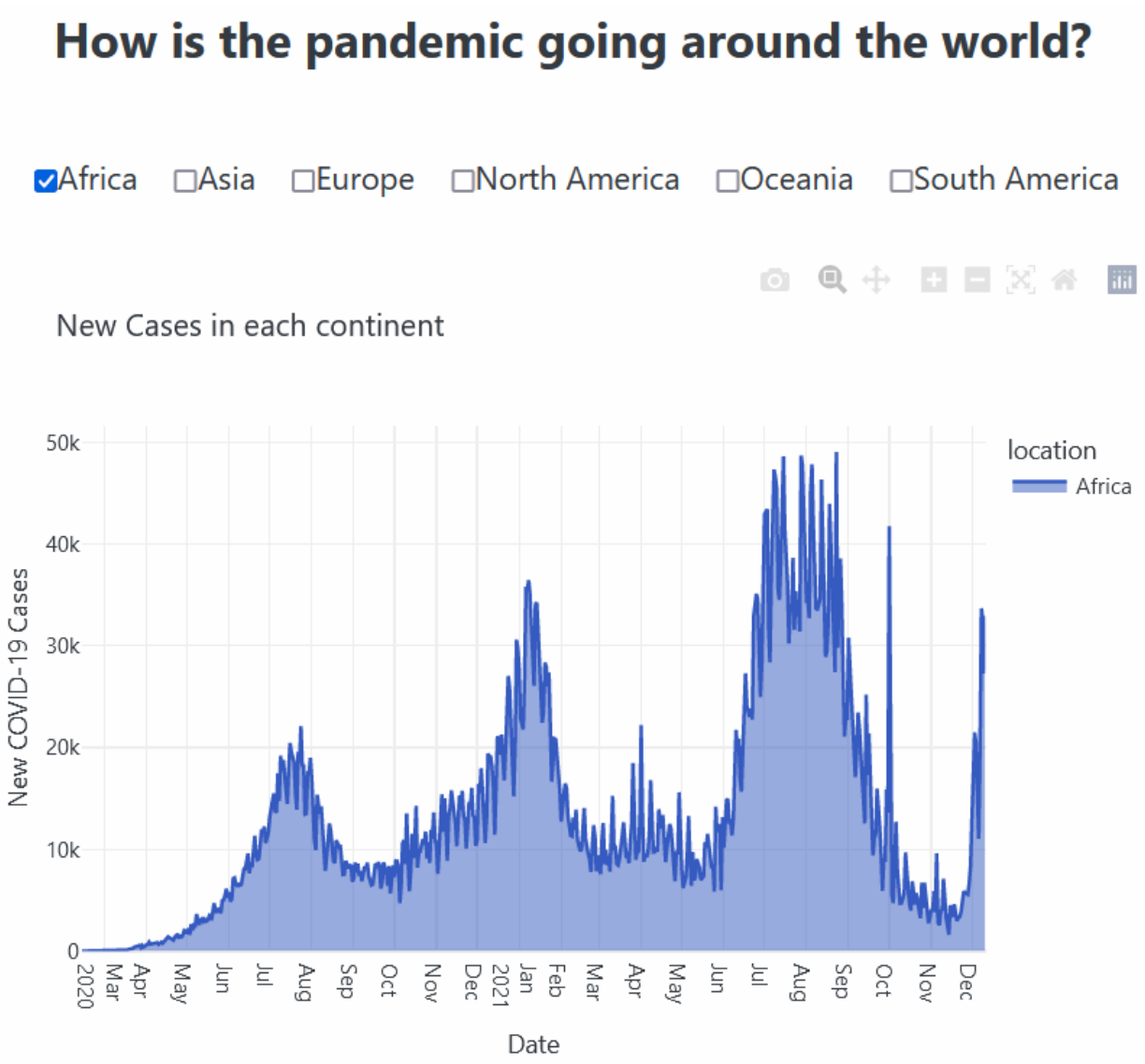
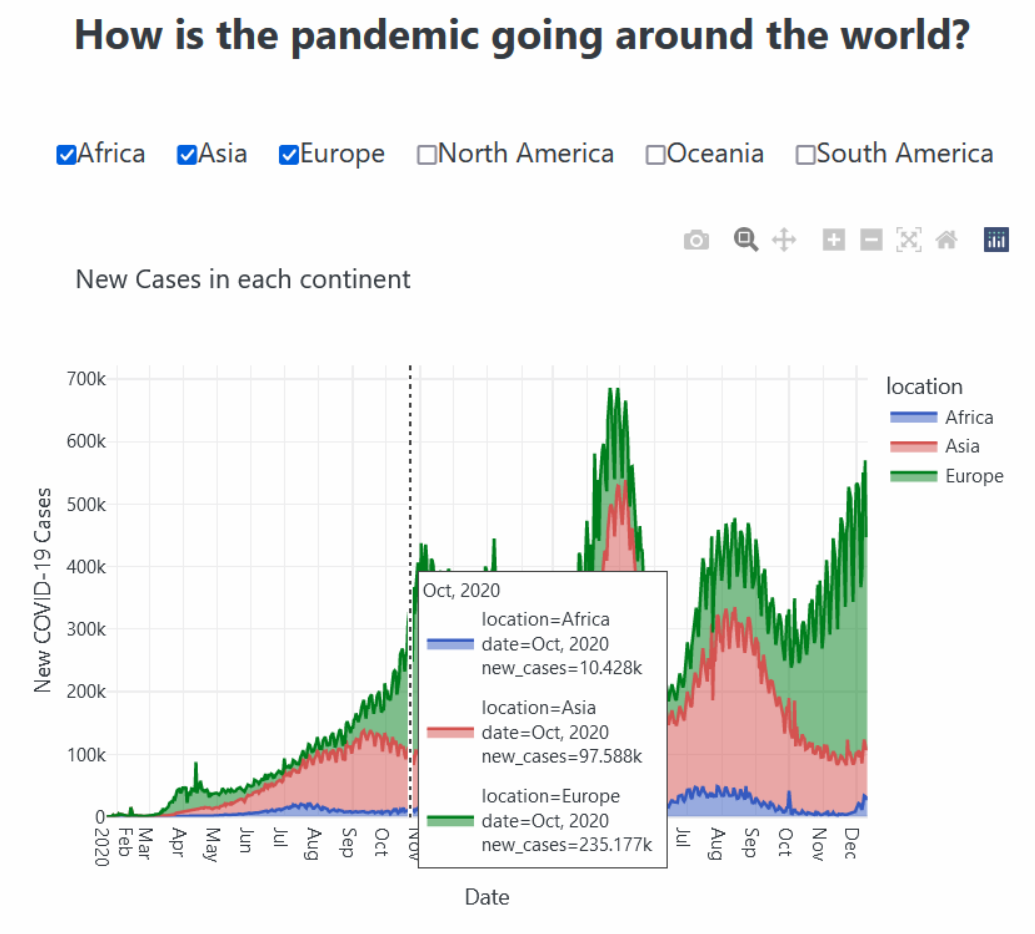


Figure 5 is a density plot that shows the changes in the number of daily reported new COVID-19 cases over all the months for each continent. In all the months, South America has the highest numbers of new cases in a single day whereas Africa has the lowest numbers. This is also visible in figure 2 since, as explained earlier, America had the brightest color while all the countries in Africa were coloured blue showing a low number of new cases per day.

Functionalities



To allow the viewer to gain knowledge about different continents, there is a checklist at the top of the graph. A single box or multiple boxes can be ticked to display one or more continents on the graph, thus making comparisons between the pandemic state in the different world regions to be vivid. In addition, when the viewer hovers over the plot, the continent, month and number of new cases is displayed for all the selected continents in the checklist to afford easy comparison.

Code

```
########### Figure 5 callback
@app.callback(
    Output("Fig5","figure"),
    [Input("fig5 checklist", "value")])

def fig5_update(value):
    df_subset = df_continents.loc[df_continents["location"].isin(value)]
    df_subset = df_subset[df_subset["new_cases"] >= 0]

    fig5 = px.area(df_subset, x="date", y="new_cases", color="location", title="New Cases in each continent")
    fig5.update_xaxes(title='Date', dtick="M1", tickformat="%b\n%Y")
    fig5.update_yaxes(title='New COVID-19 Cases')
    fig5.update_layout(hovermode="x unified")
    return fig5
```

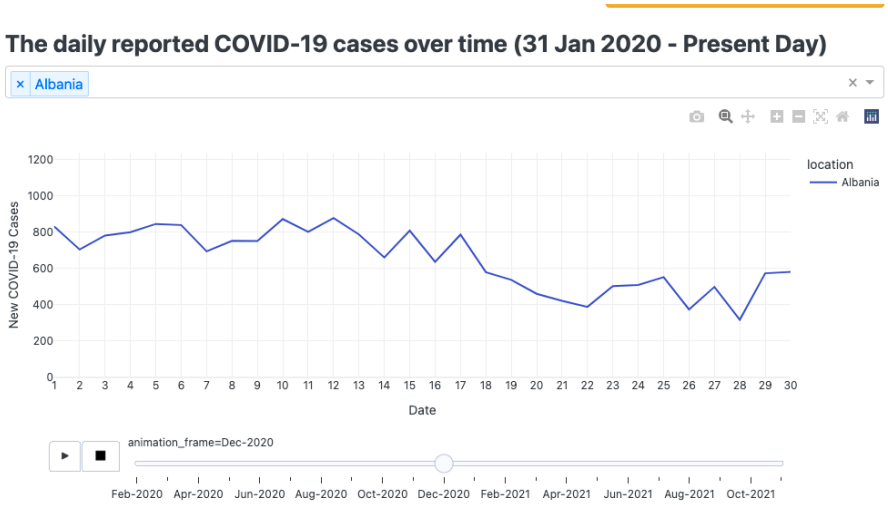
To create figure 5, the selected continents from the checklist are extracted from the dataframe then the days on which the number of cases is negative (representing unknown) are removed. Plotly express was used to create this graph where the months were plotted against the number of new cases per day and the color represents the continent. The hover mode is set to unified to show the data from all the plotted continents. The checklist was created in the app's layout using dash core components, as visible in the code snippet below. The options available are made using a for loop on the unique continents in the dataframe. The default value of this graph is Africa.

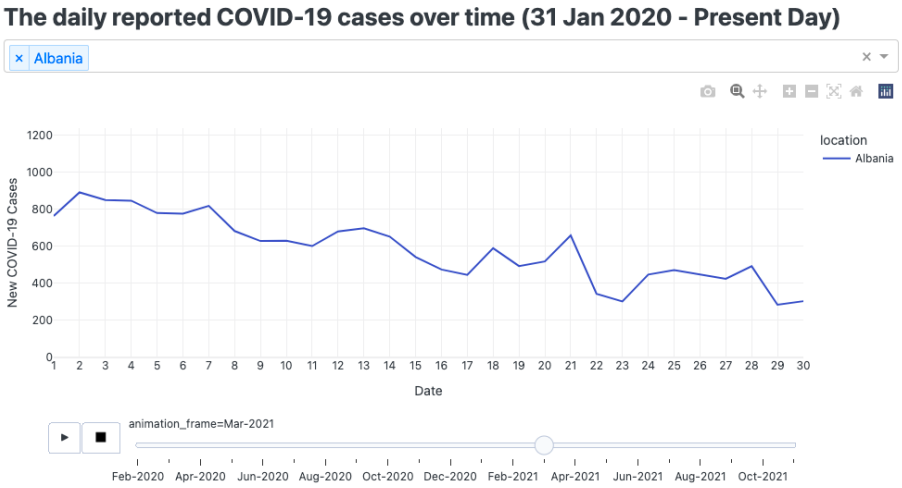
```
dbc.Row([
    html.H3("How is the pandemic going around the world?",style={'textAlign': 'center'}),
    dbc.Col([html.Div(children=[
        html.H4("Figure 5"),
        dcc.Checklist(id="fig5 checklist",
                      options=[{"label":x,"value":x} for x in pd.unique(df_continents.location)],
                      value=["Africa"],
                      labelStyle = {'display': 'inline-block', 'cursor': 'pointer', 'margin-left':'20px'}
        ),

        dcc.Graph(id="Fig5",figure={}),
     ])
    ],width={"size":8}),
])
```

User Inputs

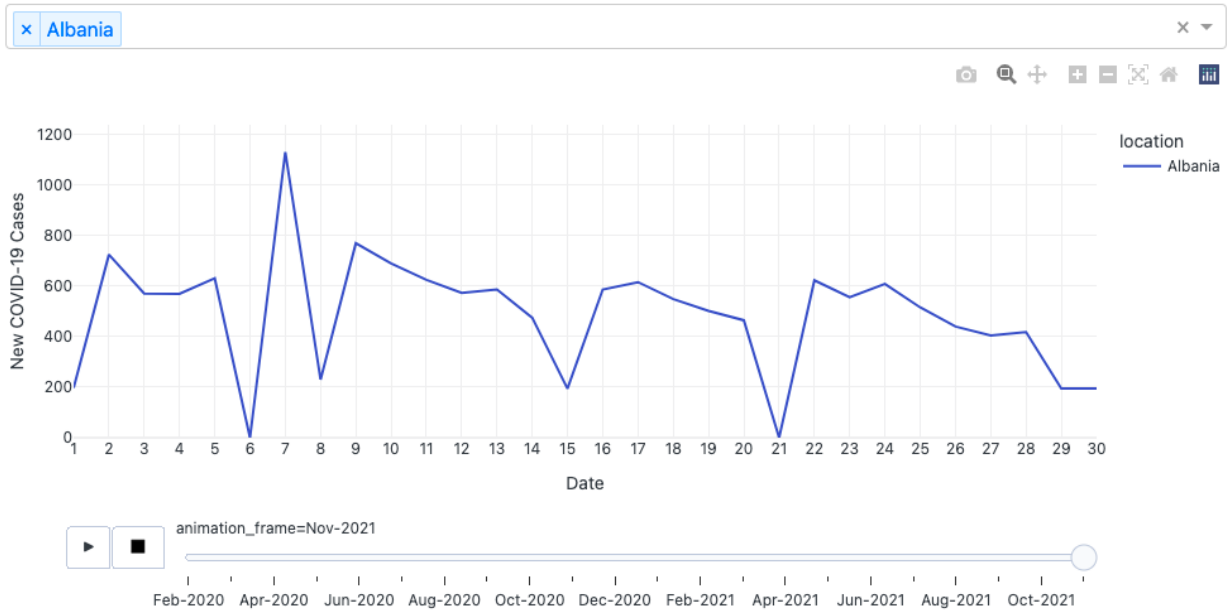
This section demonstrates some of various other possible user inputs for each figure in the dashboard.

Figure 1:





The daily reported COVID-19 cases over time (31 Jan 2020 - Present Day)



The daily reported COVID-19 cases over time (31 Jan 2020 - Present Day)

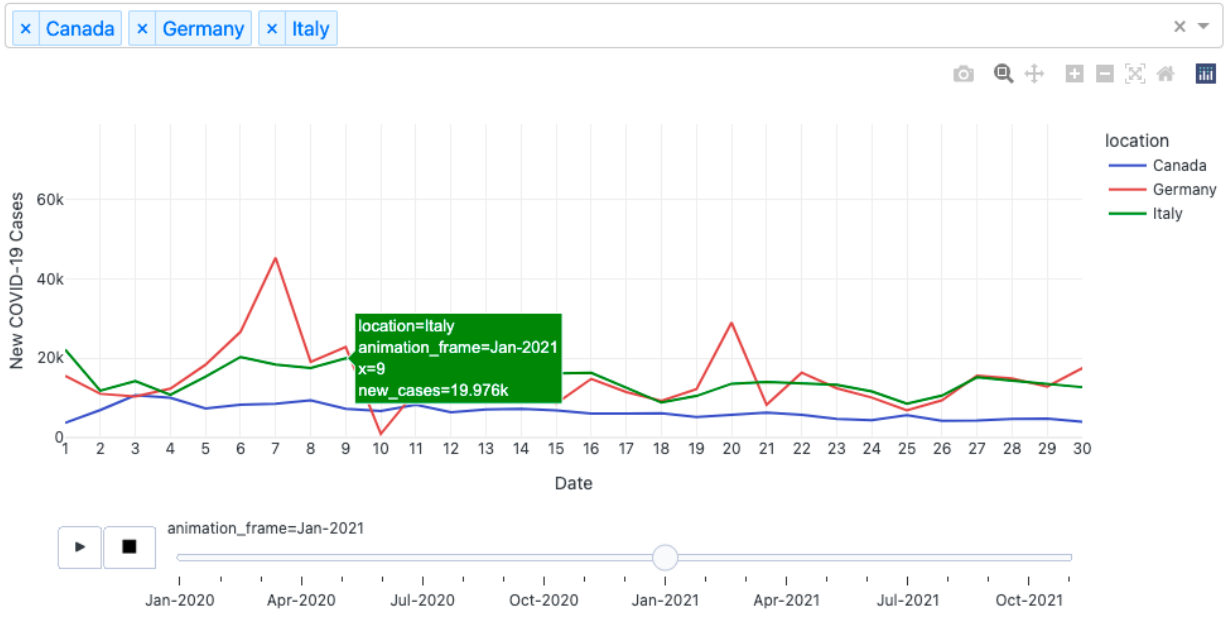
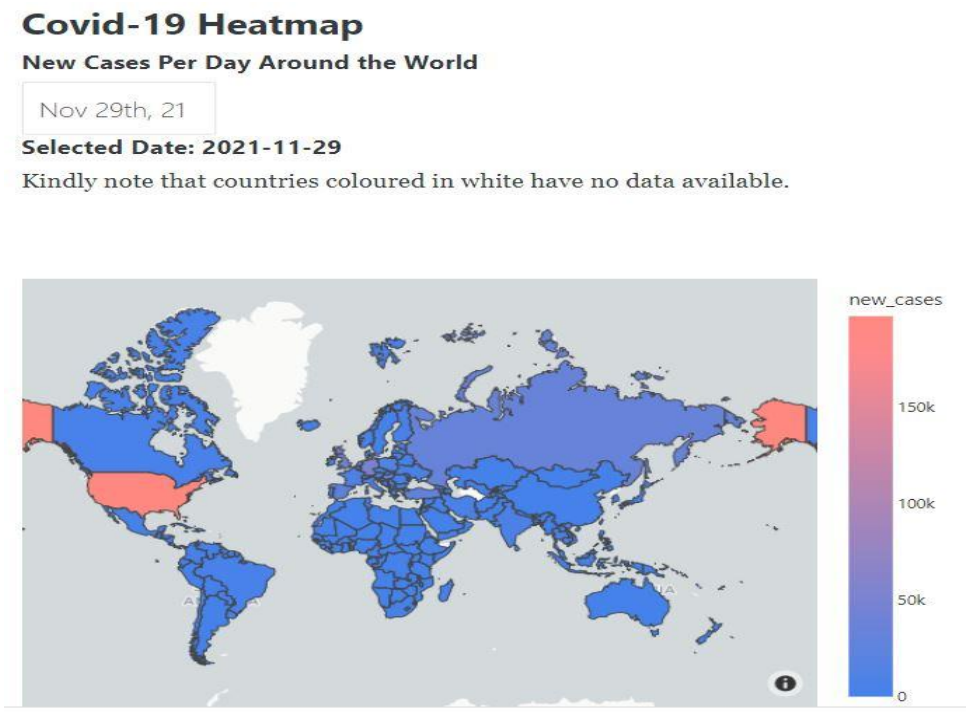


Figure 2


Covid-19 Heatmap
New Cases Per Day Around the World
Nov 29th, 21
Selected Date: 2021-11-29
Kindly note that countries coloured in white have no data available.
new_cases
150k
100k
50k
0

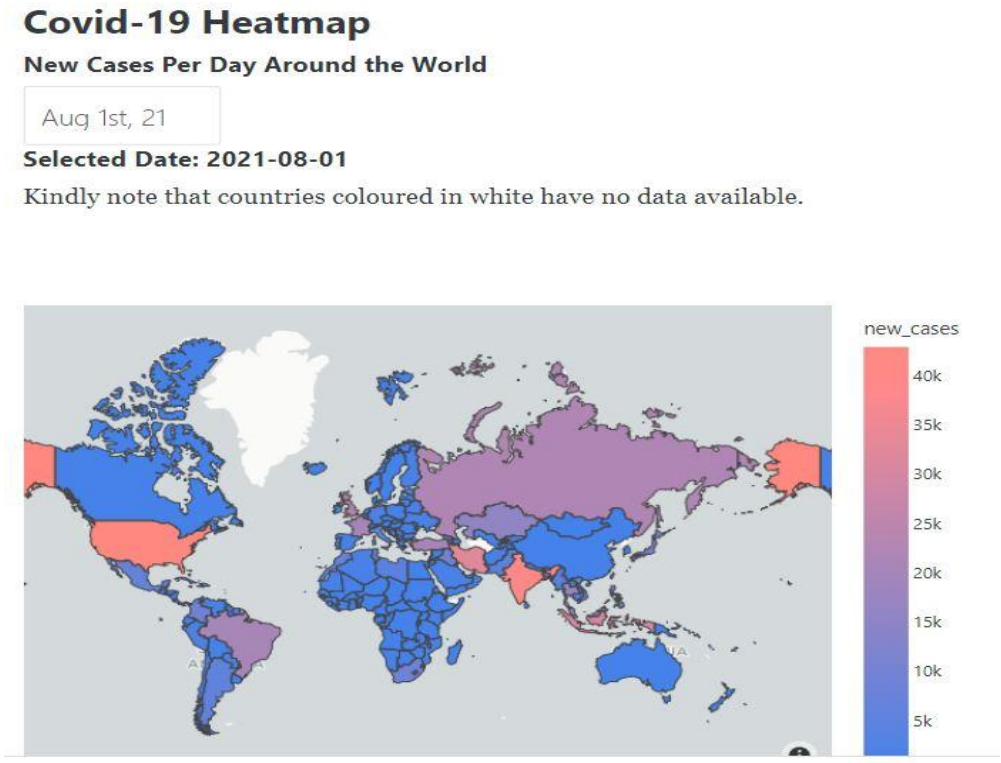
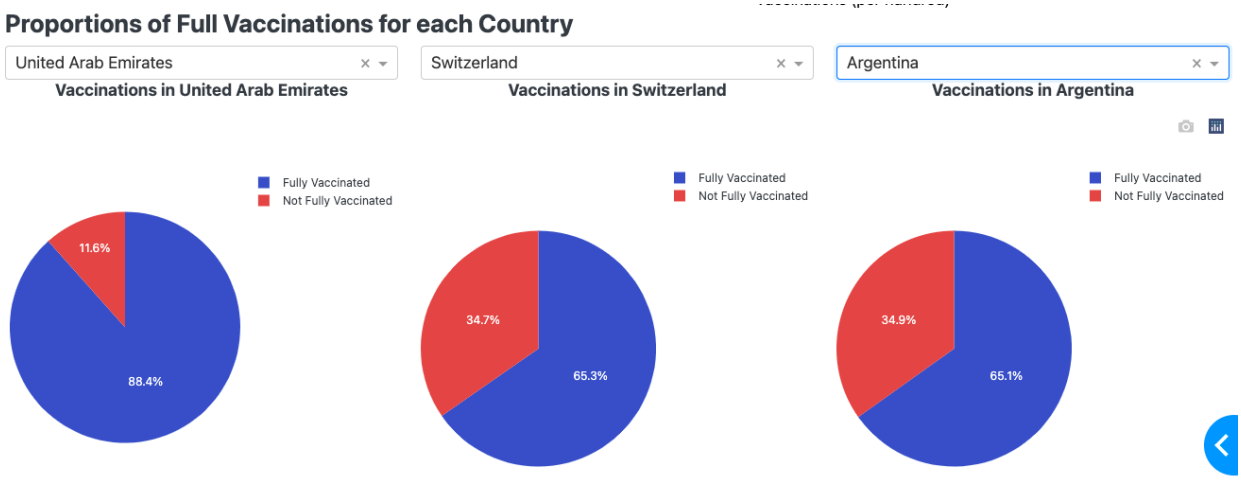

Covid-19 Heatmap
New Cases Per Day Around the World
Aug 1st, 21
Selected Date: 2021-08-01
Kindly note that countries coloured in white have no data available.
new_cases
40k
35k
30k
25k
20k
15k
10k
5k

Figure 3:


Proportions of Full Vaccinations for each Country
United Arab Emirates
Switzerland
Argentina
Vaccinations in United Arab Emirates
Vaccinations in Switzerland
Vaccinations in Argentina
Fully Vaccinated
Not Fully Vaccinated
11.6%
88.4%
34.7%
65.3%
34.9%
65.1%

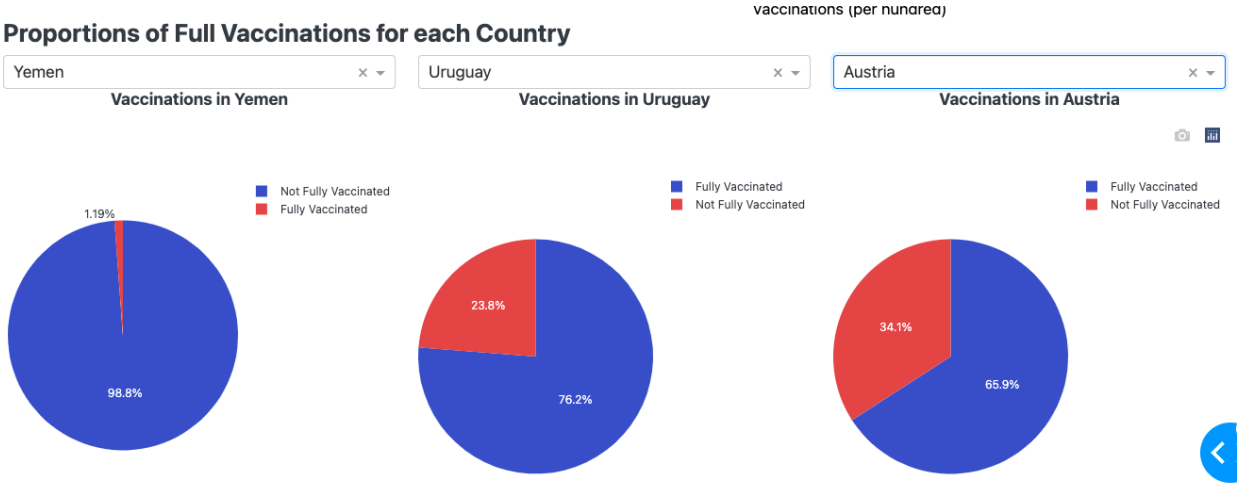

vaccinations (per hundred)
Proportions of Full Vaccinations for each Country
Yemen
Uruguay
Austria
Vaccinations in Yemen
Vaccinations in Uruguay
Vaccinations in Austria
Not Fully Vaccinated
Fully Vaccinated
1.19%
98.8%
23.8%
76.2%
34.1%
65.9%

Figure 4:

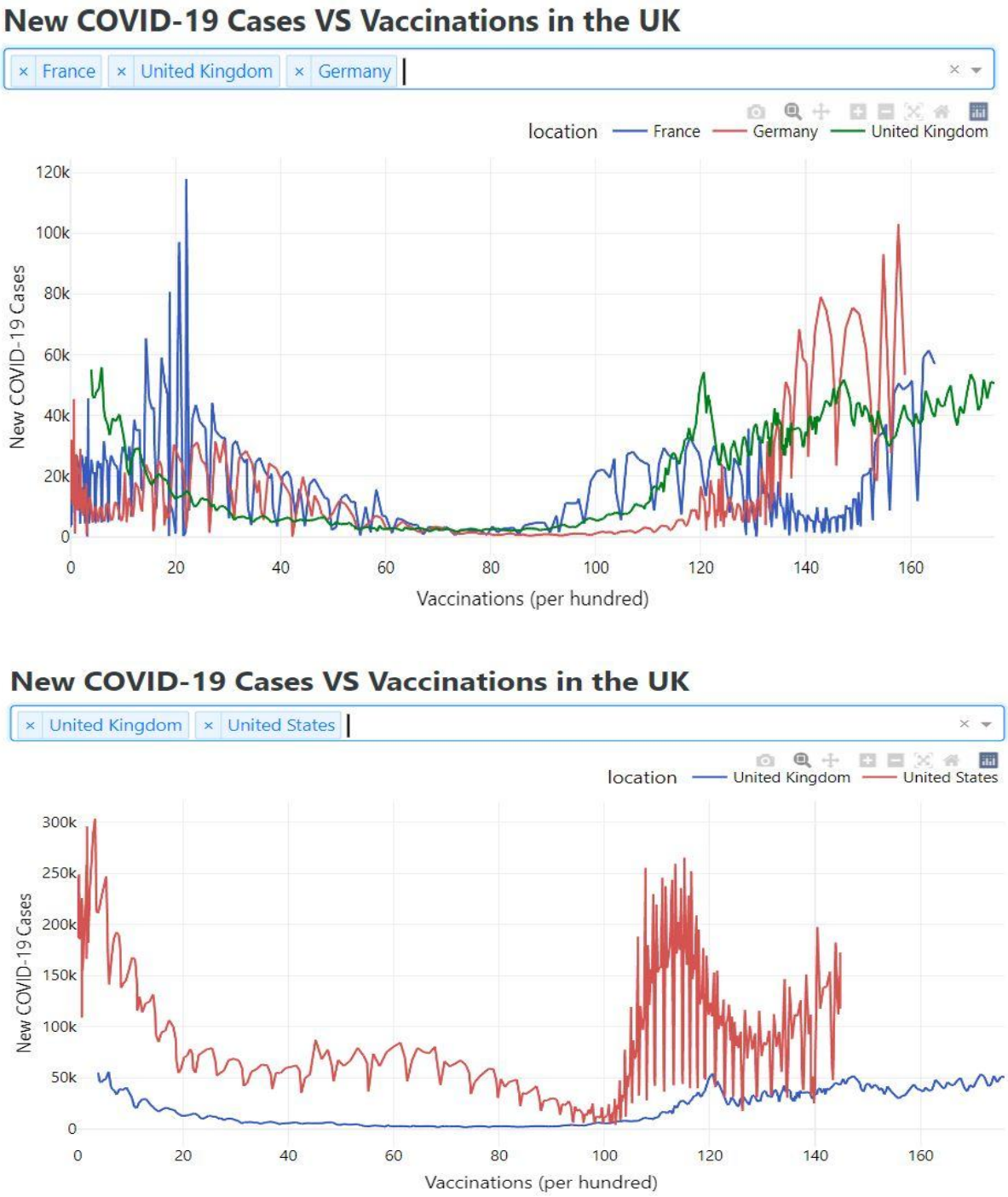
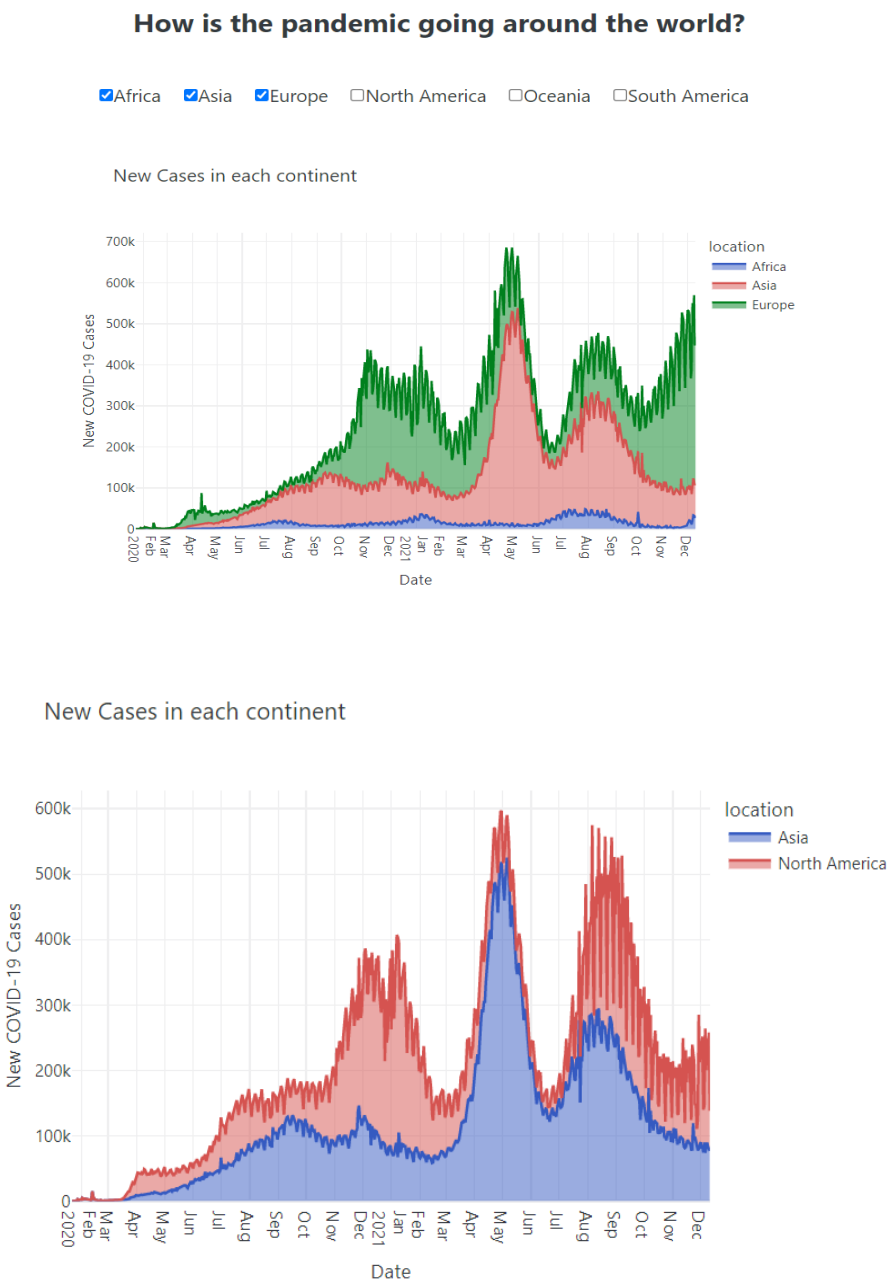
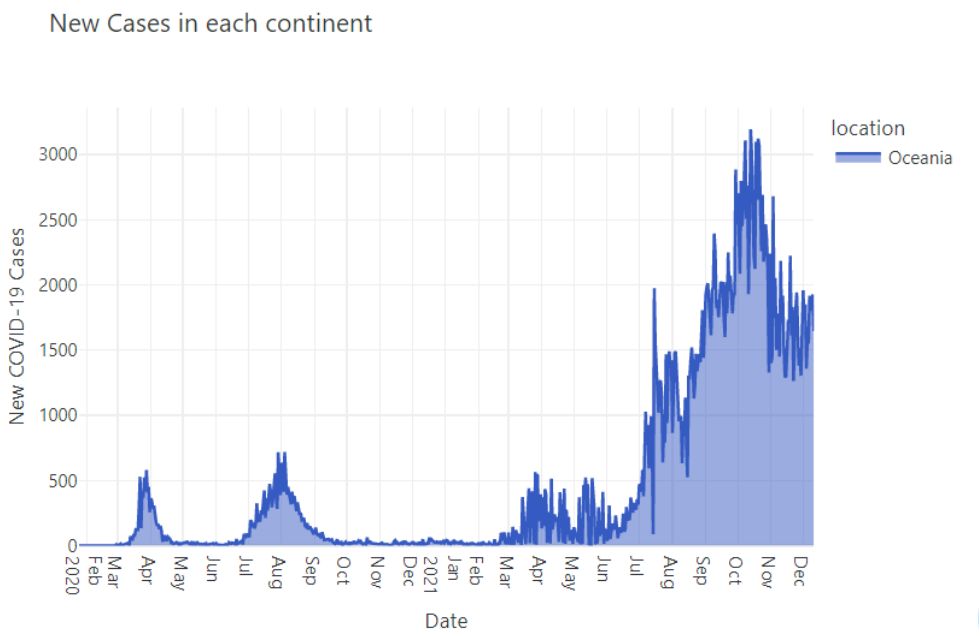

New COVID-19 Cases VS Vaccinations in the UK
France
United Kingdom
Germany
location
France
Germany
United Kingdom
120k
100k
80k
60k
40k
20k
0
New COVID-19 Cases
0
20
40
60
80
100
120
140
160
Vaccinations (per hundred)
New COVID-19 Cases VS Vaccinations in the UK
United Kingdom
United States
location
United Kingdom
United States
300k
250k
200k
150k
100k
50k
0
New COVID-19 Cases
0
20
40
60
80
100
120
140
160
Vaccinations (per hundred)

Figure 5:


How is the pandemic going around the world?
Africa
Asia
Europe
North America
Oceania
South America
New Cases in each continent
New COVID-19 Cases
700k
600k
500k
400k
300k
200k
100k
0
location
Africa
Asia
Europe
2020
2021
Date
New Cases in each continent
New COVID-19 Cases
600k
500k
400k
300k
200k
100k
0
location
Asia
North America
Date


New Cases in each continent
location
Oceania
New COVID-19 Cases
3000
2500
2000
1500
1000
500
0
2020
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec
2021
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec
Date

Statistics Cards: