COVID-19 STATISTICAL VISUALIZATION

Shreejaa Talla
Department of Computer Science
Georgia State University
Atlanta, GA
stalla1@student.gsu.edu

Harshitha Tirumani
Department of Computer Science
Georgia State University
Atlanta, GA
Htirumani1@student.gsu.edu

Abstract: The covid-19 outbreak has put the entire world to a stop. It is a newly emerging issue, and hence to get updates, we need to visit different websites and analyze. Our project has proposed a solution by gathering all the required information about covid from cases to death, recoveries, and vaccinations. Moreover, this collected data is analyzed based on the crucial points to know and displays different visualization types. The graph's interactivity introduces access to massive data, manages it, and enables an ordinary person to interpret that data in less than a min. Humans analyze images in a much faster way than text.

Get the updates that keep changing accordingly and give us live updates from different data sources. To have a clear and quick understanding of information accurately, visualizing the data can explain these events clearly and concisely for people to interpret data well. Various data visualization tools and techniques could achieve this. In this project, we identified why data visualization is necessary and its challenges and issues. We also noted that the interactivity of visualization is of utmost importance, and designing the plots applying different techniques and displaying data in various formats so that everyone can interpret it with ease.

1. DATA

1.1 USA Counties:

This data consists of five columns naming Date, County, State, Fips, Cases, Deaths. They were some empty cells in fips so we made sure to fill them while cleaning the data. So, we separately created the USA Codes table using the website. It gave us the state codes. We took only state codes as even if the fips are missing we can retrieve the data using State Codes. So that is the reason we took the state codes. We have done a cleaning step for the date columns. Data columns are basically month -month-date-date and year-year-year-year that is four times year. But using excel we have changed it to Date-Month-Name-Year. The reason behind this is while shifting the data to pandas data frame there are some bugs in the data, in order to avoid those bugs and to understand the data more clearly.

	date	county	state	fips	cases	deaths
0	21 January 2020	Snohomish	Washington	53061	1	0.0
1	22 January 2020	Snohomish	Washington	53061	1	0.0
2	23 January 2020	Snohomish	Washington	53061	1	0.0
3	24 January 2020	Cook	Illinois	17031	1	0.0
4	24 January 2020	Snohomish	Washington	53061	1	0.0
...	...	...	...	...	...	...
1047006	19 February 2021	Sweetwater	Wyoming	56037	3645	34.0
1047007	19 February 2021	Teton	Wyoming	56039	3318	9.0
1047008	19 February 2021	Uinta	Wyoming	56041	2024	12.0
1047009	19 February 2021	Washakie	Wyoming	56043	876	26.0
1047010	19 February 2021	Weston	Wyoming	56045	619	5.0

1047011 rows × 6 columns

1.2 Worldwide cases:

Worldometer is a recently calculated data for each country. It has the attributes Country/Region, continent, Population, TotalCases, NewCases, TotalDeaths, NewDeaths, TotalRecovered, Newrecovered, Activecases, serious/critical, Tot cases/ 1M population, Deaths 1M population. We have all these columns related to all the countries in the world.

]:

	Country/Region	Continent	Population	TotalCases	NewCases	TotalDeaths	NewDeaths	TotalRecovered	NewRecovered	ActiveCases	Serious,Criti
0	USA	North America	3.311981e+08	5032179	NaN	162804.0	NaN	2576668.0	NaN	2292707.0	1829
1	Brazil	South America	2.127107e+08	2917562	NaN	98644.0	NaN	2047660.0	NaN	771258.0	831
2	India	Asia	1.381345e+09	2025409	NaN	41638.0	NaN	1377384.0	NaN	606387.0	894
3	Russia	Europe	1.459409e+08	871894	NaN	14606.0	NaN	676357.0	NaN	180931.0	230
4	South Africa	Africa	5.938157e+07	538184	NaN	9604.0	NaN	387316.0	NaN	141264.0	53
...	...	...	...	...	...	...	...	...	...	...	
204	Montserrat	North America	4.992000e+03	13	NaN	1.0	NaN	10.0	NaN	2.0	N
205	Caribbean Netherlands	North America	2.624700e+04	13	NaN	NaN	NaN	7.0	NaN	6.0	N
206	Falkland Islands	South America	3.489000e+03	13	NaN	NaN	NaN	13.0	NaN	0.0	N
207	Vatican City	Europe	8.010000e+02	12	NaN	NaN	NaN	12.0	NaN	0.0	N
208	Western Sahara	Africa	5.986820e+05	10	NaN	1.0	NaN	8.0	NaN	1.0	N

200 rows × 16 columns

1.3 Country_Vaccinations:

Country vaccinations has the data of recent vaccinations that is until 22 April. It has the columns such as iso codes, date, country, total vaccinations, fully vaccinated people, daily vaccinated people, total vaccination, source name and the combination of different vaccinations.

]:

	country	iso_code	date	total_vaccinations	people_vaccinated	people_fully_vaccinated	daily_vaccinations_raw	daily_vaccinations	total_vacc
0	Albania	ALB	10 January 2021	0.0	0.0	NaN	NaN	NaN	
1	Albania	ALB	11 January 2021	NaN	NaN	NaN	NaN	64.0	
2	Albania	ALB	12 January 2021	128.0	128.0	NaN	NaN	64.0	
3	Albania	ALB	13 January 2021	188.0	188.0	NaN	60.0	63.0	
4	Albania	ALB	14 January 2021	266.0	266.0	NaN	78.0	66.0	
...	...	...	...	...	...	...	...	...	
3804	Wales	NaN	16 February 2021	820339.0	807351.0	12988.0	17161.0	23033.0	
3805	Wales	NaN	17 February 2021	841975.0	822633.0	19342.0	21636.0	22012.0	
3806	Wales	NaN	18 February 2021	864498.0	839065.0	25433.0	22523.0	20649.0	
3807	Wales	NaN	19 February 2021	885906.0	853904.0	32002.0	21408.0	18891.0	
3808	Wales	NaN	20 February 2021	897856.0	860083.0	37773.0	11950.0	17376.0	

3809 rows × 15 columns

1.4 USA Codes:

USA codes data is prepared to support the USA counties data as there are few missing fips which need to be identified while implementing maps and here this data is extracted from (United States Zip Codes (worldpostalcode.com)) and formed into an excel file. The data consists of state codes and states of the USA.

```
us_codes = pd.read_csv("us_Codes.csv",dtype={'code':str
del us_codes['Unnamed: 2']
us_codes
```

']:

	code	state
0	AL	Alabama
1	AK	Alaska
2	AS	American Samoa
3	AZ	Arizona
4	AR	Arkansas
...	...	...
61	AE	Armed Forces Africa
62	AE	Armed Forces Canada
63	AE	Armed Forces Europe
64	AE	Armed Forces Middle East
65	AP	Armed Forces Pacific

66 rows × 2 columns

2. DATA CLEANING

In the data cleaning process, all the null and NAN values are replaced by the possible values such as 0 or unknown, or they can be replaced based on the type data. For float/int values the most acceptable replacement would be 0.0 or 0. Similarly for a string it would be “unknown”. This data cleaning can be done in various methods using backward fill and forward fill or the combination backward and forward fill.

2.1 USA counties data:

Here, for us_counties we first check for null or NAN values using isnull() and sum() on the dataframe. Based on the below fig1, we change only those column values where the null values are present and here fips and deaths have null values. Hence they are replaced by using the replace function as shown below fig2.

```
In [4]: import pandas as pd
        # import plotly.figure_factory as ff
        import numpy as np
        import plotly.express as px

        us_counties = pd.read_csv("us-counties.csv",dtype={'fips':str})
        us_counties.isnull().sum()

Out[4]: date          0
        county        0
        state         0
        fips       9640
        cases         0
        deaths    22661
        dtype: int64
```

fig1 US countries data before cleaning

```
In [5]: import pandas as pd
        # import plotly.figure_factory as ff
        import numpy as np
        import plotly.express as px

        us_counties = pd.read_csv("us-counties.csv",dtype={'fips':str})
        us_counties['fips']=us_counties['fips'].replace(to_replace = np.nan,value = 0)
        us_counties['deaths']=us_counties['deaths'].replace(to_replace = np.nan,value = 0)
        us_counties.isnull().sum()

Out[5]: date      0
        county    0
        state     0
        fips      0
        cases     0
        deaths    0
        dtype: int64
```

fig2 US countries data after cleaning

2.2 Worldwide cases data:

For world_wide_cases we first check for null or NAN values using isnull() and sum() on the dataframe. Based on the below fig1, we change only those column values where the null values are present by replacing those null values with appropriate data mentioned above in the data cleaning process.

```
In [1]:
import pandas as pd
import numpy as np
import plotly.express as px
import datetime

world = pd.read_csv("worldometer_data.csv")
```

```
In [2]:
world.isnull().sum()
```

```
Out[2]:
Country/Region        0
Continent             1
Population            1
TotalCases            0
NewCases            205
TotalDeaths          21
NewDeaths           206
TotalRecovered        4
NewRecovered        206
ActiveCases           4
Serious,Critical     87
Tot Cases/1M pop      1
Deaths/1M pop        22
TotalTests           18
Tests/1M pop         18
WHO Region           25
dtype: int64
```

fig1 world_wide_cases data before cleaning

```
In [180]:
import pandas as pd
import numpy as np
import plotly.express as px
import datetime

world = pd.read_csv("worldometer_data.csv")
world['Continent']=world['Continent'].replace(to_replace = np.nan,value = "unknown")
world['WHO Region']=world['WHO Region'].replace(to_replace = np.nan,value = "unknown")
world['Population']=world['Population'].replace(to_replace = np.nan,value = 1)
world['NewCases']=world['NewCases'].replace(to_replace = np.nan,value = 0)
world['TotalDeaths']=world['TotalDeaths'].replace(to_replace = np.nan,value = 0)
world['NewDeaths']=world['NewDeaths'].replace(to_replace = np.nan,value = 0)
world['TotalRecovered']=world['TotalRecovered'].replace(to_replace = np.nan,value = 0)
world['NewRecovered']=world['NewRecovered'].replace(to_replace = np.nan,value = 0)
world['ActiveCases']=world['ActiveCases'].replace(to_replace = np.nan,value = 0)
world['Serious,Critical']=world['Serious,Critical'].replace(to_replace = np.nan,value = 0)
world['Tot Cases/1M pop']=world['Tot Cases/1M pop'].replace(to_replace = np.nan,value = 0)
world['Deaths/1M pop']=world['Deaths/1M pop'].replace(to_replace = np.nan,value = 0)
world['TotalTests']=world['TotalTests'].replace(to_replace = np.nan,value = 0)
world['Tests/1M pop']=world['Tests/1M pop'].replace(to_replace = np.nan,value = 0)
```

```
In [181]:
world.isnull().sum()
```

```
Out[181]:
Country/Region      0
Continent           0
Population          0
TotalCases          0
NewCases            0
TotalDeaths         0
NewDeaths           0
TotalRecovered      0
NewRecovered        0
ActiveCases         0
Serious,Critical    0
Tot Cases/1M pop    0
Deaths/1M pop       0
TotalTests          0
Tests/1M pop        0
WHO Region          0
dtype: int64
```

fig2 world_wide_cases data after cleaning

2.3 Country vaccinations data:

For Country vaccination data, cleaning process is performed based on nulls present and vaccinations,people_vaccinated,people_fully_vaccinated,daily_vaccinations_raw,Tota_vaccinations_per_hundered,people_vaccinated_per_hundered,people_fully_vaccinated_per_hundered,daily_vaccinations_per_million,daily_vaccinations have null values. Hence they are replaced by using the replace function as shown below fig2.

```
In [13]:
import dash
import pandas as pd
import plotly.express as px
import numpy as np


vaccines = pd.read_csv('country_vaccinations.csv')
vaccines.isnull().sum()
```

```
Out[13]: country                                 0
iso_code                                280
date                                      0
total_vaccinations                     1342
people_vaccinated                      1718
people_fully_vaccinated                2426
daily_vaccinations_raw                 1729
daily_vaccinations                      139
total_vaccinations_per_hundred         1342
people_vaccinated_per_hundred          1718
people_fully_vaccinated_per_hundred    2426
daily_vaccinations_per_million          139
vaccines                                  0
source_name                               0
source_website                            0
dtype: int64
```

fig1 Country_Vaccinations data before cleaning

```
In [16]:
import dash
import pandas as pd
import plotly.express as px
import numpy as np


vaccines = pd.read_csv('country_vaccinations.csv')
vaccines['iso_code'] = vaccines['iso_code'].replace(to_replace = np.nan,value = "unknown")
vaccines['total_vaccinations'] = vaccines['total_vaccinations'].replace(to_replace = np.nan,value = 0)
vaccines['people_vaccinated'] = vaccines['people_vaccinated'].replace(to_replace = np.nan,value = 0)
vaccines['people_fully_vaccinated'] = vaccines['people_fully_vaccinated'].replace(to_replace = np.nan,value = 0)
vaccines['daily_vaccinations_raw'] = vaccines['daily_vaccinations_raw'].replace(to_replace = np.nan,value = 0)
vaccines['total_vaccinations_per_hundred'] = vaccines['total_vaccinations_per_hundred'].replace(to_replace = np.nan,value = 0
vaccines['people_vaccinated_per_hundred'] = vaccines['people_vaccinated_per_hundred'].replace(to_replace = np.nan,value = 0)
vaccines['people_fully_vaccinated_per_hundred'] = vaccines['people_fully_vaccinated_per_hundred'].replace(to_replace = np.nan
vaccines['daily_vaccinations_per_million'] = vaccines['daily_vaccinations_per_million'].replace(to_replace = np.nan,value = 0
vaccines['daily_vaccinations'] = vaccines['daily_vaccinations'].replace(to_replace = np.nan,value = 0)
vaccines['date']=pd.to_datetime(vaccines.date)
vaccines['date']= vaccines["date"].dt.strftime("%Y %m %d")
vaccines.isnull().sum()
```

```
Out[16]: country                               0
iso_code                              0
date                                  0
total_vaccinations                    0
people_vaccinated                     0
people_fully_vaccinated               0
daily_vaccinations_raw                0
daily_vaccinations                    0
total_vaccinations_per_hundred        0
people_vaccinated_per_hundred         0
people_fully_vaccinated_per_hundred   0
daily_vaccinations_per_million        0
vaccines                              0
source_name                           0
source_website                        0
dtype: int64
```

fig2 Country_Vaccinations data after cleaning

3. QUESTIONS TO FORM THE GRAPHS

1. The USA map and the spread of covid based on cases and deaths.
2. The spread in each county of a state, based on the data hovered on a USA map.
3. The spread in state through the timeline based on hover data from the USA map.
4. The most affected countries of each continent based on total cases.
5. Serious and active cases of each WHO region for a continent.
6. The total recovered, tested, deaths for different WHO regions for a continent.
7. The fully vaccinated people by the number of vaccinations present in each country and the type of vaccines.
8. The total vaccines present per 100 and daily vaccinated people throughout the world are based on different sources.
9. The total vaccinations and display vaccinations by ministry of health.
10. Total vaccinations and fully vaccinated people for each vaccine based on the dropdown value.

4. CODE FOLDER EXPLANATION

Term_project

|------- apps

|------ __init__.py

|------ navigation.py

|------ spread.py

|------ vaccination.py

|------ wcases.py

|------- Data

|------ Country-Vaccination.csv

|------ US-codes.csv

|------ US-counties.csv

|------ Worldometer-data.csv

|------- Docs

|------ Report

|------ PPT

|------- app.py

|------- Index.py

APPS:

This folder contains all the sub python files used by index.py.

- __init__.py : This python file is used to start other .py files present in the apps folder.
- navigation.py : This python file consists of a navigation bar implemented from dash bootstrap components which has three links - spread, cases and vaccinations. The home page is set to spread link and contains the layout of all the figures.
- spread.py : This python file uses the US_counties.csv, US_codes.csv file and cleans this data using pandas and depicts three plots based on that data and contains the layout of all the figures.
- vaccination.py : This python file uses the country_vaccinations.csv file and performs a data cleaning step on it and generates scatter geo, bubble and scatter plots and contains the layout of all the figures.
- wcases : This python file uses worldometer.csv data and performs data cleaning operations on them and illustrates different charts such as sunburst, line plot, categorical axes plot and box plots.

app.py : This python file is used in the above sub python file to access the dash app and the dash app is initialised inside it.

index.py : This python file is the main page which imports all the above sub python code files and app.py and applies a conditional based layout to each sub code based on the navigation.

5. TOOLS USED

- Plotly graphs:

  Plotly graphs used throughout the project are

  - *Sunburst chart*: Sunburst plots visualize hierarchical data spanning outwards radially from root to leaves. The sunburst sector hierarchy is determined by the entries in labels (names in px.sunburst) and in parents. The root starts from the center and children are added to the outer rings[1].
  - *Choropleth map*:A Choropleth Map is a map composed of colored polygons. It is used to represent spatial variations of a quantity. This page documents how to build tile-map choropleth maps, but you can also build outline choropleth maps using our non-Mapbox trace types.
  - *Scatter-geo*: each line of the dataframe is represented as a marker point. The column set as the size argument gives the size of markers in a map[1].
  - *Time-series graph*: This graph is similar to a line chart with a standard property to add x-axis as a variable of time/date.

- *Bar graph*: The values of x and y axis are defined with a bar.
- *Categorical-axes chart*: Similar to a bar chart but show two different types of data on a single x-axis by stacking or separately based on the attributes entered.
- *Scatter plot*: Each data point is represented as a marker point, whose location is given by the x and y columns[1].
- *Bubble plot*: Similar to the scatter plot but bubble chart has an extra property to define size of markers.
- *Line chart*: Each data point is represented as a vertex (which location is given by the x and y columns) of a polyline mark in 2D space[1].
- *Box plot*: Box plots represent the quartile ranges of the given data.

- Plotly Dash:

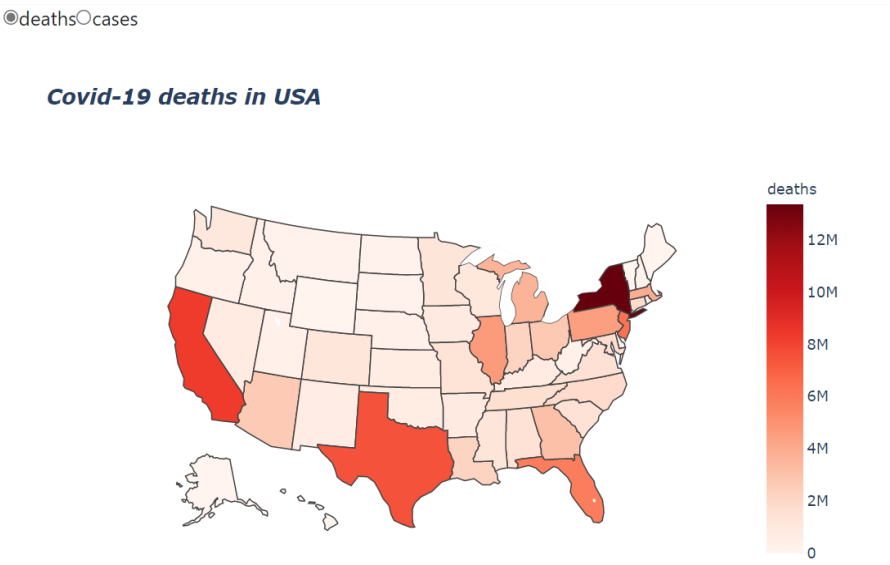
  The following components of plotly dash are used in the code:

  - *Dash core components*: Dash core components such as dropdown, markdown, radio items, input and graph are used to interact with figures.
  - *Dash HTML components*: Dash HTML components such as html headers, html center, html div, and html labels are used for different html elements.
  - *Dash bootstrap components*: Row, columns, bootstrap style sheets, header classes and simple navigation bar are used for navigation and arrangement of figures.

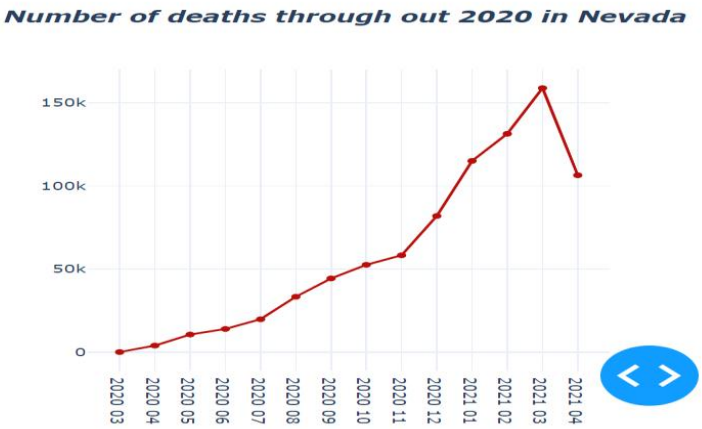
6. CODE COMPILATION

In order to compile the code and display the results, go to the term project folder and import all the required libraries such as dash, plotly, dash bootstrap and other basic python libraries. Finally, open the command prompt (on windows) with the project folder where index.py is visible and run the index.py python file(python index.py). It runs the code and generates the flask/dash based localhost link, that is, localhost:8050. Detailed explanation is provided in the demo video and each interaction is performed in it.

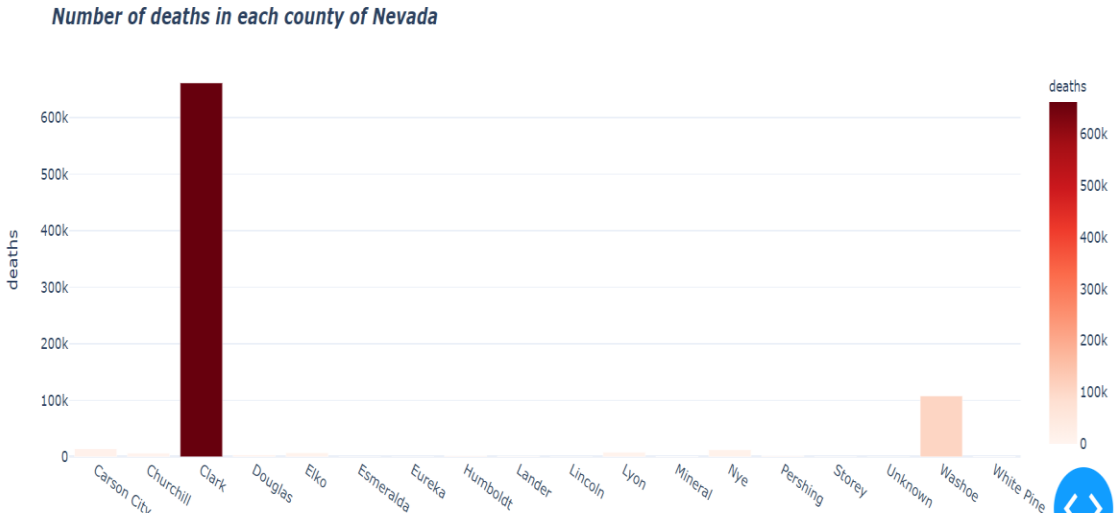
7. DATA VISUALISATION RESULTS



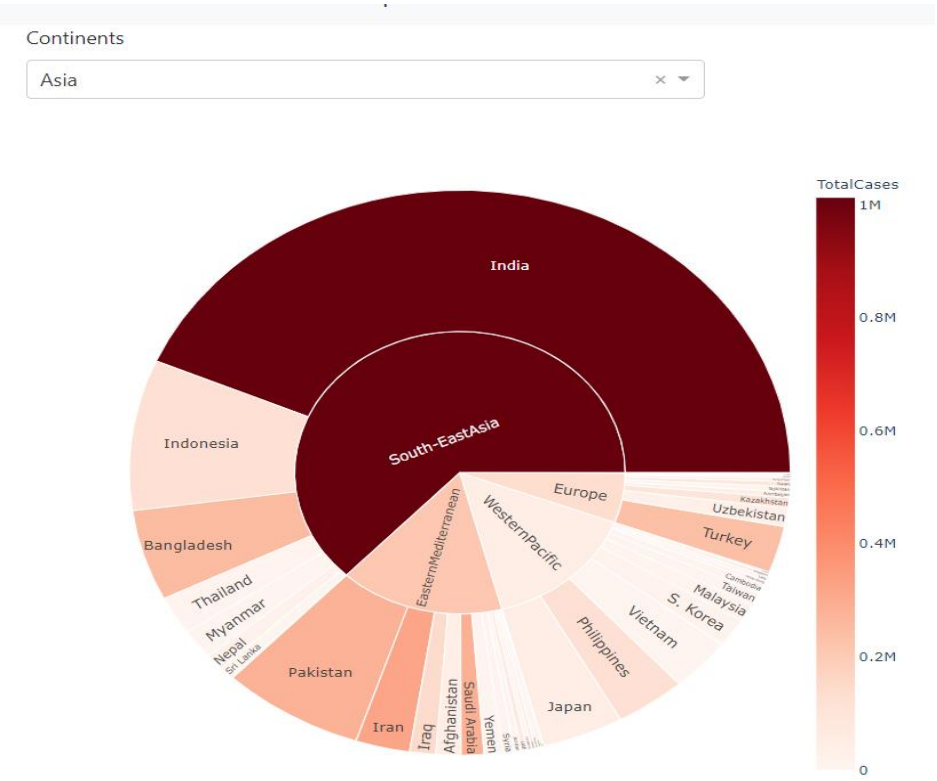
This graph illustrates the spread of covid-19 deaths or cases based on the radio button in the USA map and when we hover on this map based on every hover the data is collected which is in the form of json and this json data is used by other two plots to change the state based on each hover. This data might take a couple of minutes to load but at the end results are produced accurately.



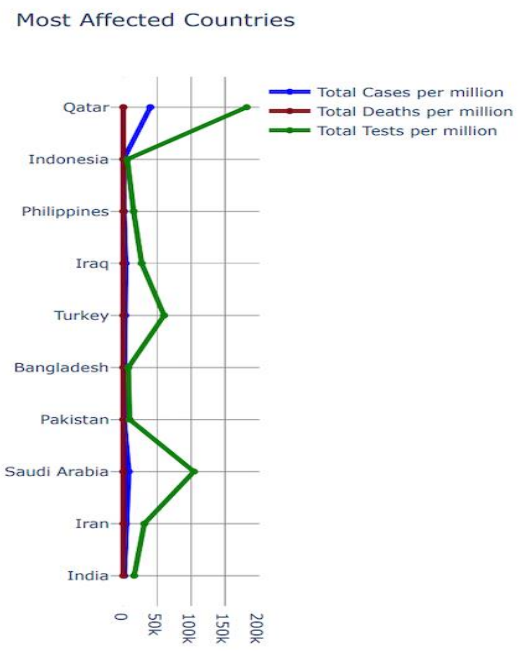
This graph illustrates the number of deaths or cases based on the radio button throughout 2020 in each county of a state which is obtained from the map data.



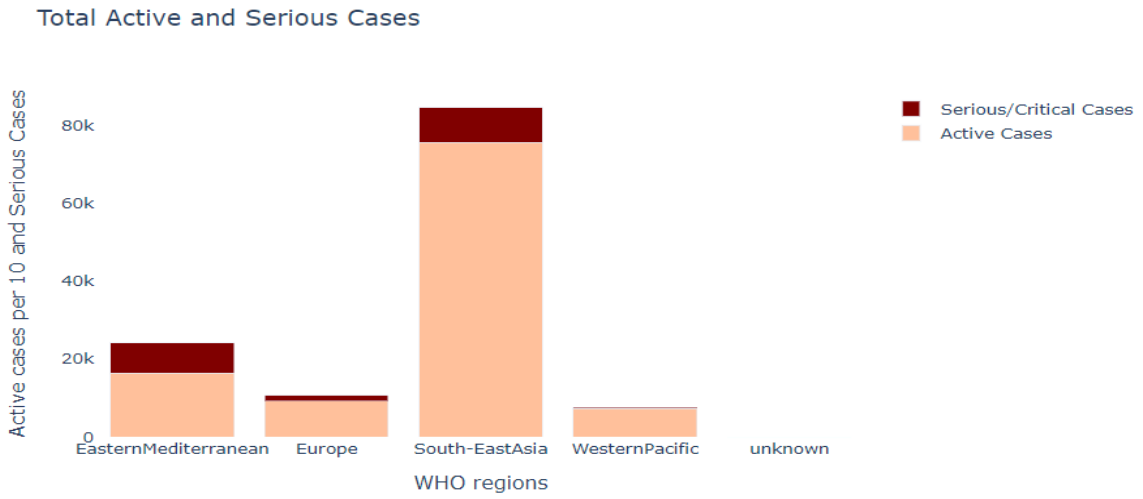
This graph illustrates the number of deaths or cases based on the radio button in each county of a state which is obtained from the map data.



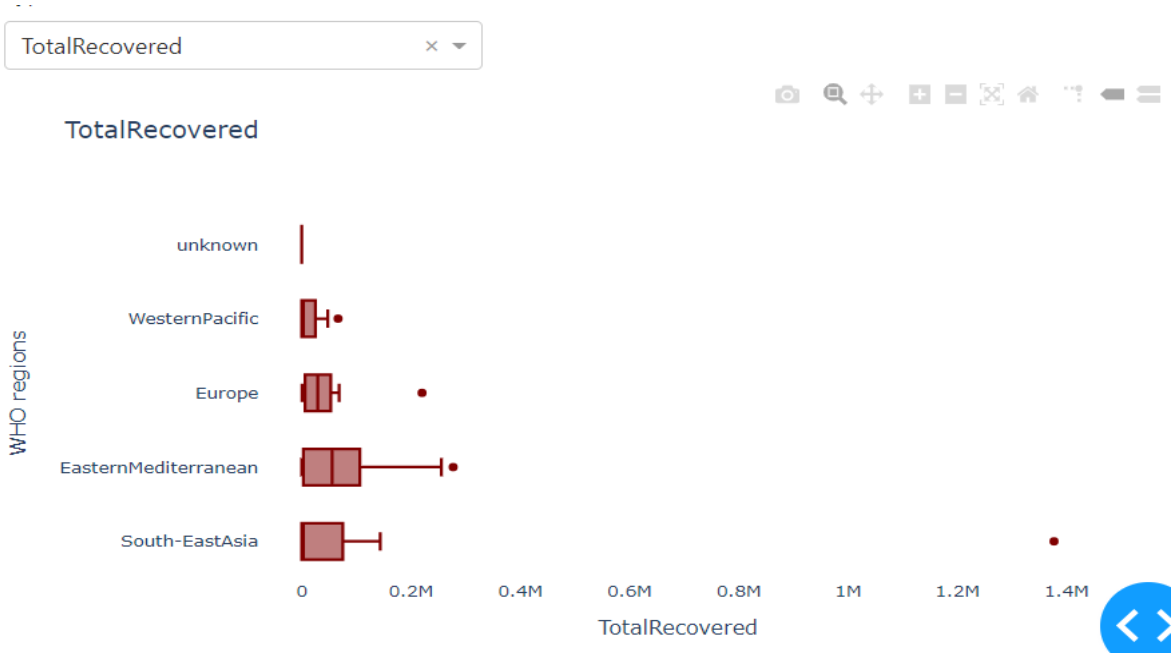
This graph illustrates the total cases in which the intensity of the color and population are displayed as the weights or partitions in the graph. Here, the inner circle or parent nodes depicts the WHO regions of each continent based on the values obtained from dropdown and children or outer circle illustrates the countries. This is interacted based on its parent nodes and shows the data related to one parent node when it is clicked.



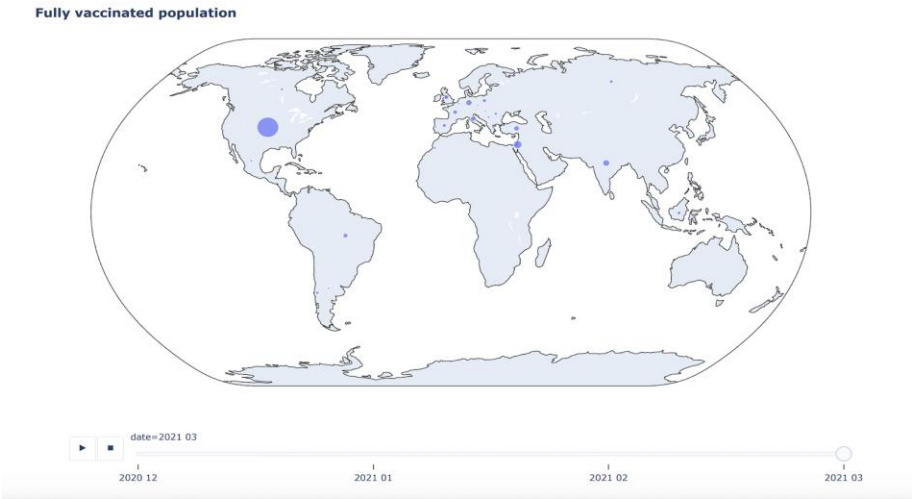
This graph illustrates the most affected countries of each continent via dropdown values and depicts total cases, deaths and tests performed in those countries per million.



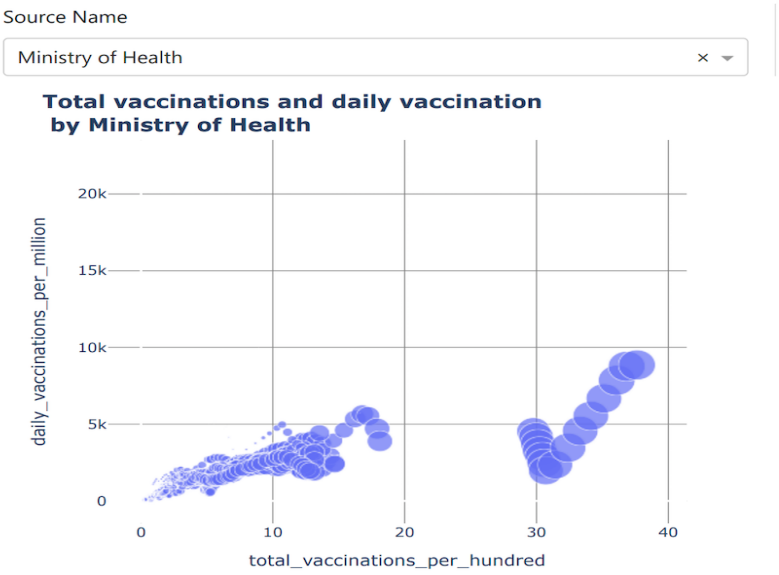
This graph illustrates the serious and active cases of each WHO region for a continent and the continent value is obtained from the dropdown .



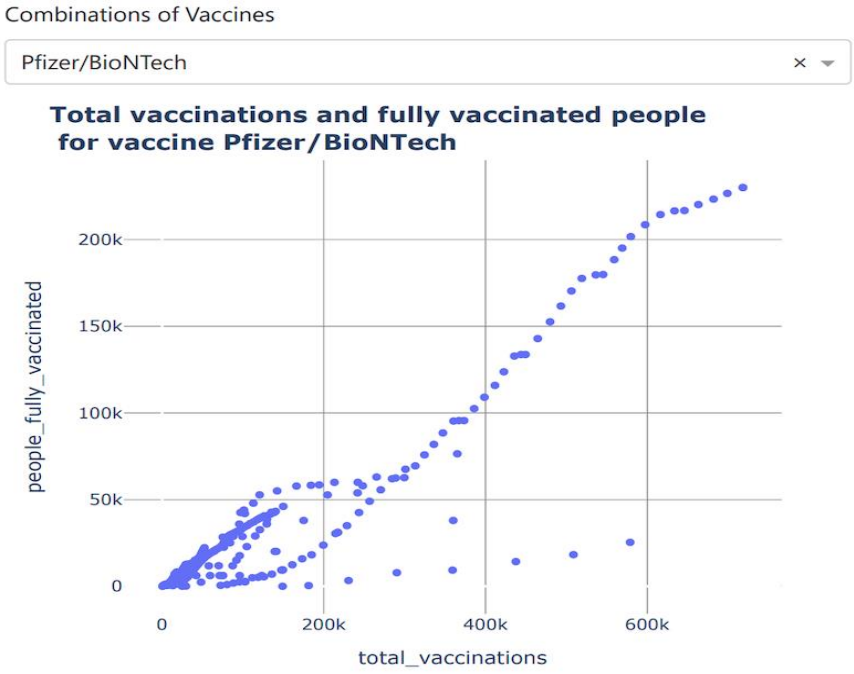
This graph illustrates the total recovered with respect to the WHO regions for each continent based on the values of dropdown. The dropdown of type is used to change the x-axis data to total recovered or total test or total deaths.



This graph illustrates the world map describing the fully vaccinated people through the world based on timelines. with an animation based on year and months and we can also move the slider to a particular time, the data related to that time is obtained on the map. We can see that the country with the most population fully vaccinated is the United states. when we hover on this map we get the data related to that particular country.



This graph illustrates the total vaccinations per hundred with the daily vaccinations per million with the different country colours. Each colour representing a country with the drop down we can select the source name accordingly. In the above chart we have selected the source name ministry of health.



This graph illustrates the total vaccinations with the people fully vaccinated for the vaccine Pfizer/BioNTech.while we have each country with different colours. We have a drop down which has different combinations of vaccines. the graph automatically changes the respective graph according to the vaccine.

6. REFERENCES

1. https://plotly.com/python/
2. Deploy your Dash App | Dash for Python Documentation | Plotly
3. United States Zip Codes (worldpostalcode.com)
4. https://plotly.com/python/bubble-charts/
5. https://plotly.com/python/sunburst-charts/
6. https://plotly.com/python/categorical-axes/

CONTRIBUTIONS:

Shreejaa Talla:

- Data collection and understanding the data.
- Analyze different plots for the data.
- Data cleaning and processing for all the data files.
- Implementing all dashboard elements.
- Developing the code for navigation.py, index.py, app.py, spread.py and wcases.py.
- Developing one dropdown and its interaction in vaccinations.py.
- Documentation part about 40%.

Harshitha Tirumani:

- Presentation.
- Documentation.
- Implementing plots on vaccination.py.

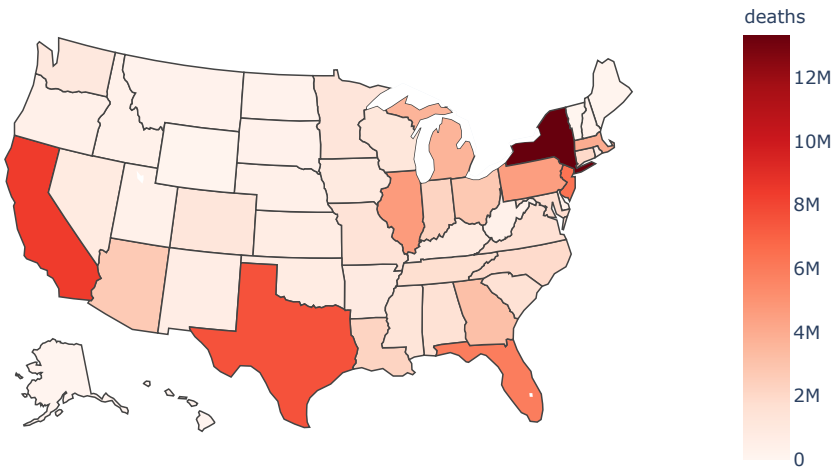
USA Covid-19 Statistics

These graphs represents the covid-19 statistics for different counties of USA. **Graph1** represents *the USA map and the spread of covid based on cases and deaths.* **Graph2** represents *the spread in each county of a state, based on the data hovered on USA map.* **Graph3** represents *the spread in state through the timeline.*

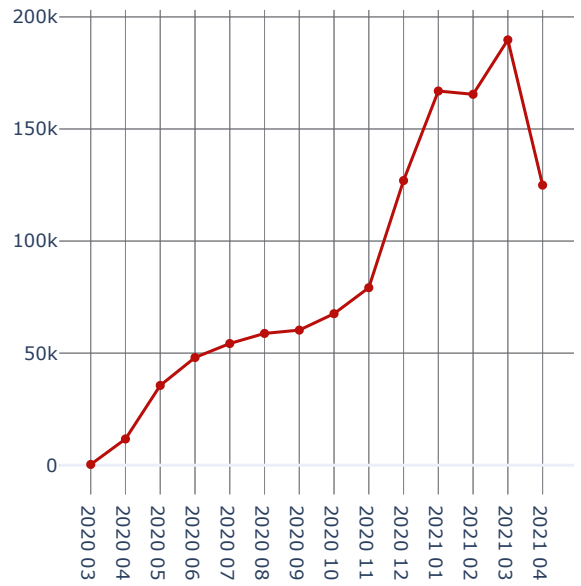
Covid Deaths/cases

◉deaths○cases

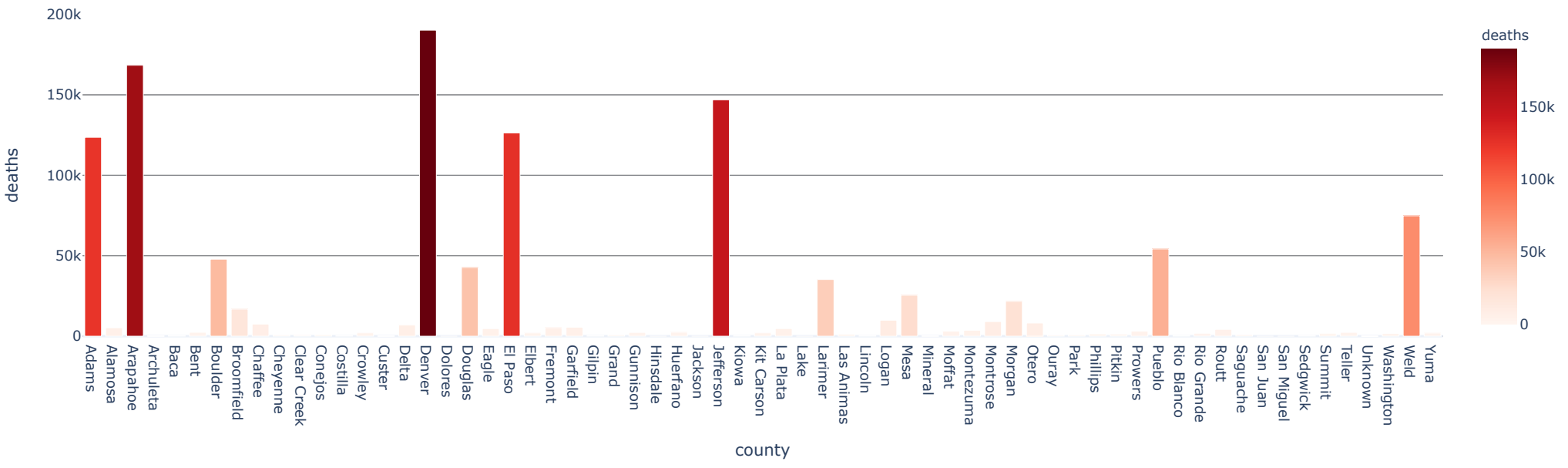
Covid-19 deaths in USA



Number of deaths through out 2020 in Colorado



Number of deaths in each county of Colorado

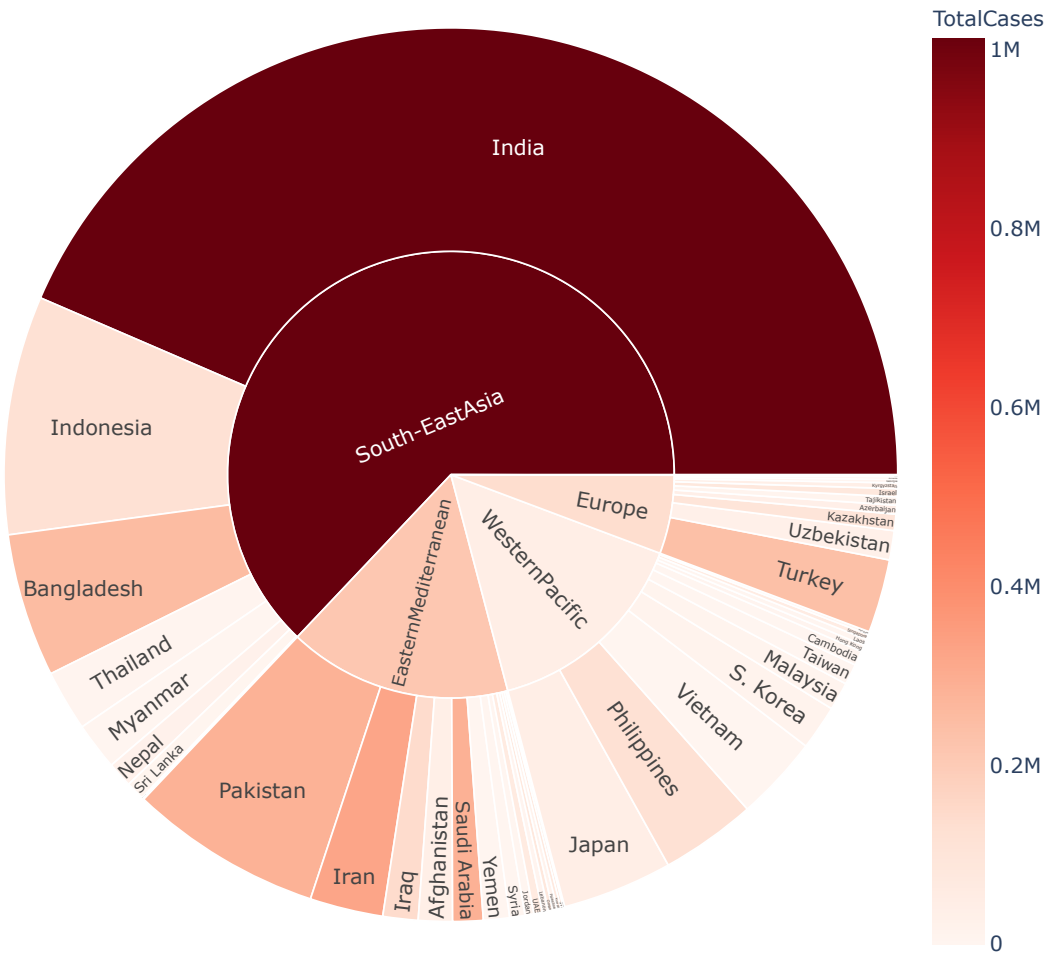


Asia covid-19 spread statistics

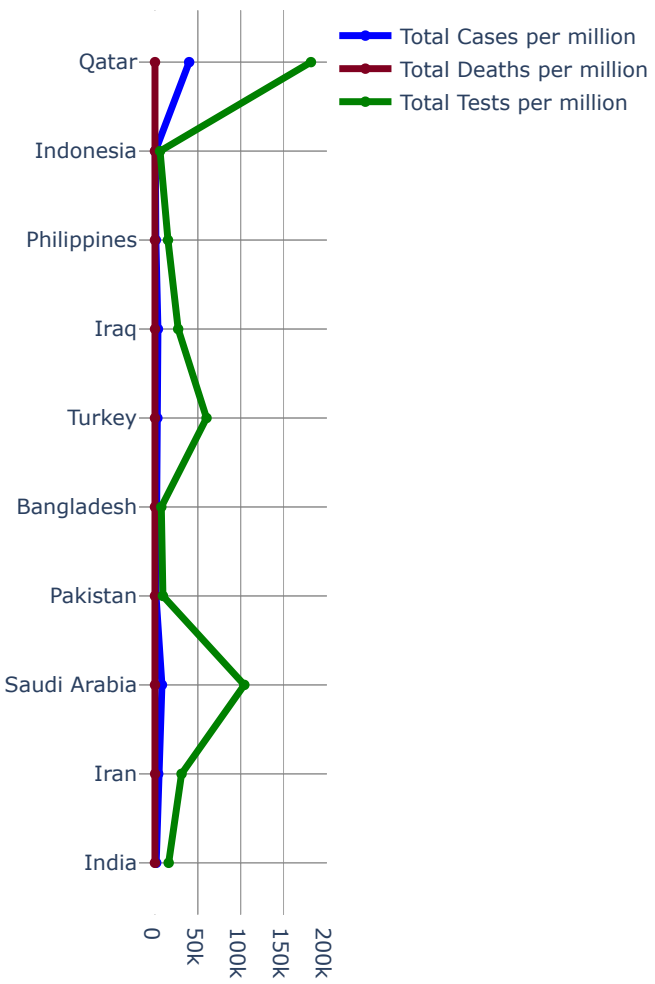
These graphs represents the cases, deaths, recoveries and tests from each continent, **Graph1** represents *the total cases and population based on each WHO region and countries of a continent*, **Graph2** represents *the most effected countries of each continent based on total cases*, **Graph3** represents *the serious and active cases of each WHO region for a continent* , **Graph4** represents *the total recovered, tested, deaths for different WHO regions for a continent*.

Continents

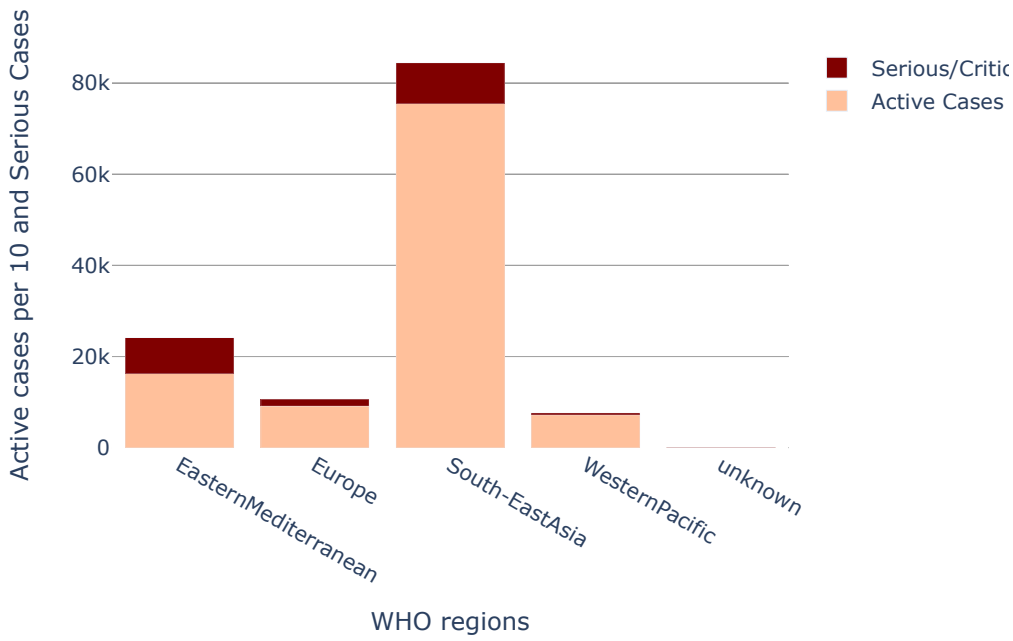
Asia



Most Affected Countries



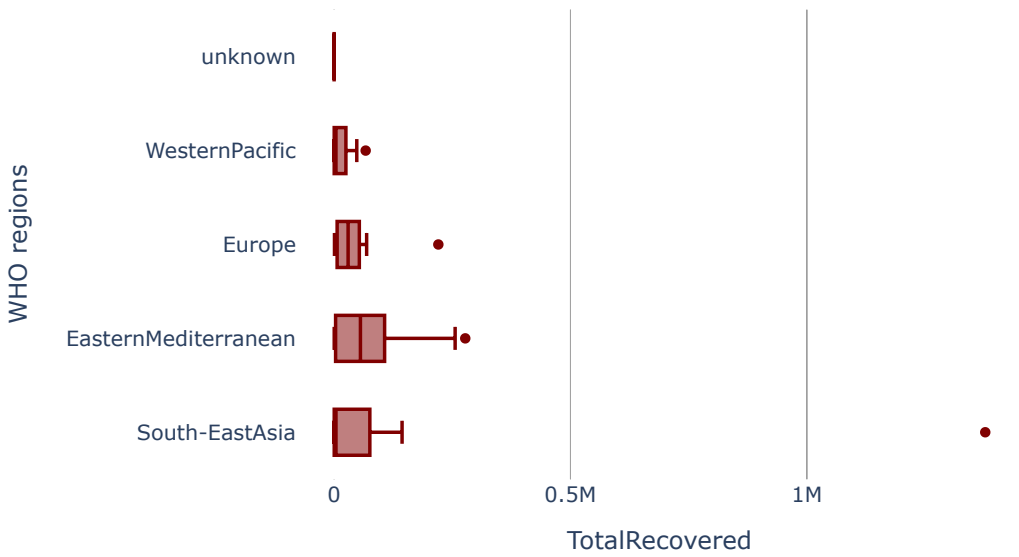
Total Active and Serious Cases



Types

TotalRecovered

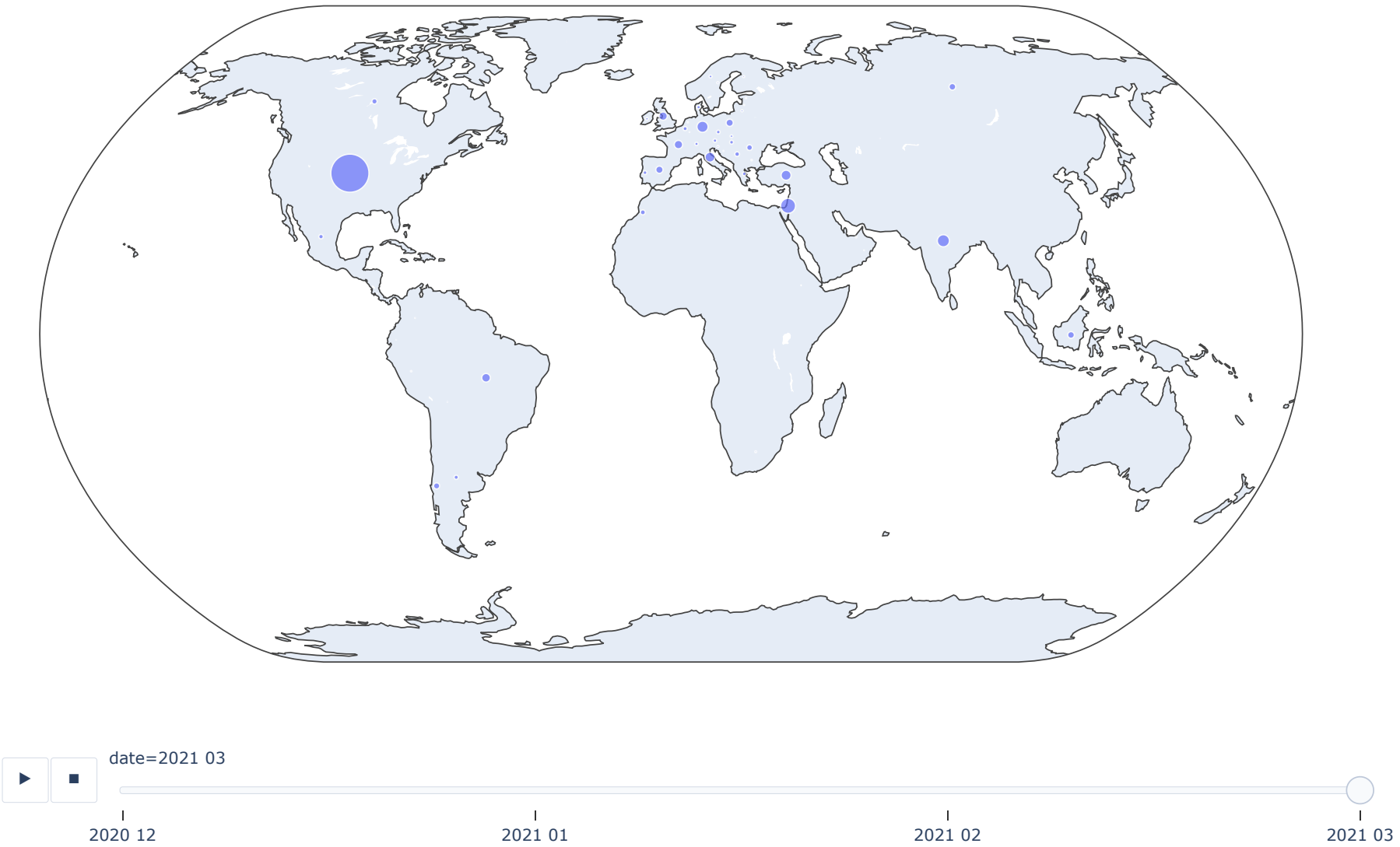
TotalRecovered



Covid-19 Vaccination progress through the World

These graphs represents the covid-19 vaccination progress for different countries in the world. **Graph1** represents *the world map describing the fully vaccinated people through the world based on timelines*. **Graph2** represents *the fully vaccinated people by the number of vaccinations present in each country and the type of vaccines*. **Graph3** represents *the total vaccines present per 100 and daily vaccinated people through the world based on different sources*.

Fully vaccinated population



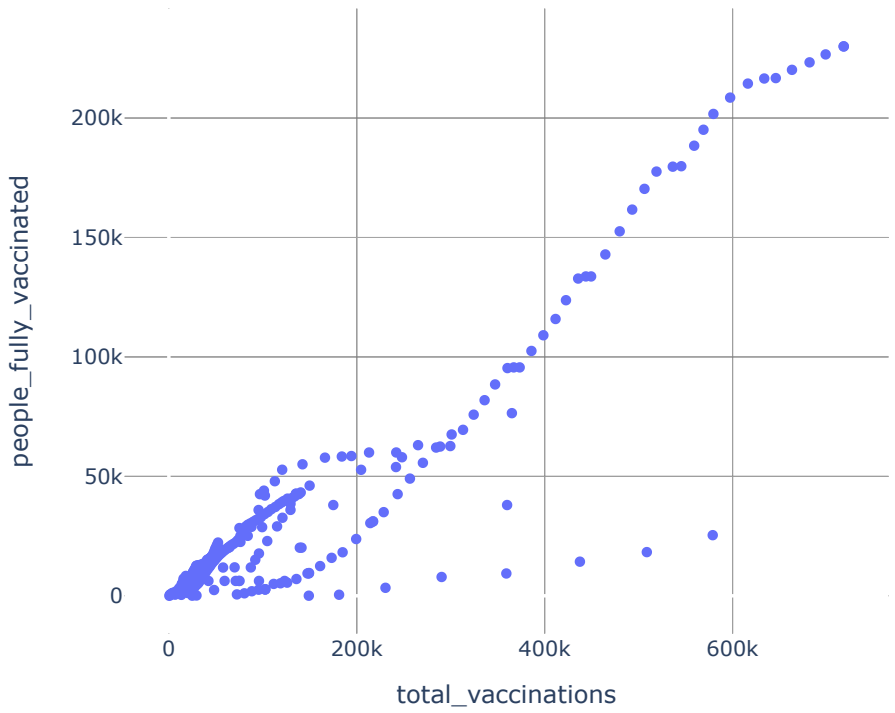
Combinations of Vaccines

Pfizer/BioNTech

Source Name

Ministry of Health

Total vaccinations and fully vaccinated people for vaccine Pfizer/BioNTech



Total vaccinations and daily vaccination by Ministry of Health