Project Assignment 3

by Alisiya Balayan, Bishoy Abdelmalik, Arian Dehghani, Ariel Kohanim, Sergio Ramirez, Natalie Weingart

1. Datasets Used

1.1. **Dataset A** - COVID-19 World Vaccination Progress
- https://www.kaggle.com/gpreda/covid-world-vaccination-progress

1.2. **Dataset B** - Novel CoronaVirus 2019 Dataset
- https://www.kaggle.com/sudalairajkumar/novel-corona-virus-2019-dataset

1.3. **Dataset C** - Countries population by year 2020
- https://www.kaggle.com/eng0mohamed0nabil/population-by-country-2020

1.4. **Dataset D** - COVID-19 daily vaccination
- https://github.com/owid/covid-19-data/tree/master/public/data/vaccinations

2. Data Mining Goals

2.1. Using datasets acquired, we are predicting when will 70% of the U.S. population achieve herd immunity (hypothesis).

2.2. Identify which entities and attributes in the datasets are needed to make statistical conclusions and will be used for visualization.

2.3. Show clustering, skewness, kernel density estimation for the datasets.

3. Statistical Techniques

3.1. Clustering of vaccination data (figure 4.1).
- Figure 4.1 shows clustering of vaccination data per state where we can observe that California, Texas, Florida and New York have the most prominent clusters (i.e., shown in darker blue compared to the greyed out states). This clustering occurred for a few reasons, one being that these states have high populations and more vaccines are being delivered. Therefore, more people have access to vaccinations. However, to reach 70% immunity in these states, vaccination should continue to increase. For example, California's population is 39 million, and total vaccination (on 4/23/2021) was 26 million.

3.2. Datasets visualizations (figures 4.2 - 4.5).
- Figures 4.2-4.5 represent visualizations of our datasets where we are able to see total covid-19 cases, death, recoveries and vaccinations. Our datasets and graphs show data from January 21, 2020 until February 25, 2021 for Covid-19 cases, as well as vaccination data ranging from January 14, 2021 until April 23, 2021. Later on, we will build skewness graphs and heatmaps based on these figures and code.

3.3. Skewness of Covid-19 cases data (figures 5.6-5.8).

- **Mean, mode, median and standard deviation.**

3.3.■.1. Mean of a dataset is found by adding all numbers in the dataset and dividing it by the total count of numbers.

3.3.■.2. Mode is the number that often occurs in the dataset.

3.3.■.3. Median is the middle value of the dataset when ordered from greatest to least.

3.3.■.4. Standard deviation is a measure of how spread out the numbers in the dataset are.

3.3.■.5. We obtained standard deviation and mean for total covid-19 cases, recovered, and death cases by using *describe()* method in python. Median and mode numbers were obtained by using the *statistics module* in python (as shown in section 6).

- **Positive Skewness**

3.3.■.1. Figures 4.6 and 4.8 display positive skewness after graphing it where it shows that mean comes after median and mode. Positive skew can be identified by its looks (i.e., “fatter tail on the right), as well as verifying it with mean, median and mode where the sequence of values should be as follows: mode, median, mean in this exact order. For example, figure 4.6 shows a positive skew with a mean of 1.24 which is a positive number, so it means that there are no outliers. Skewness of recovered cases was approximately 1.1 using *skew()* method which means that it is not a normal distribution. Positive skew in recovery cases (figure 4.6) and death cases (figure 4.8) show that although a lot of people recovered from disease, many people also died because of it.

- **Negative Skewness**

3.3.■.1. Figure 4.7 displays negative skewness where it shows that mean, median and mode come after each other in this exact order. Although we can observe that it is a negative skew simply by identifying the “fatter tail on the left”, mean, median and mode values are used to check the conclusion. Negative skew is observed when we graphed total covid-19 confirmed cases which means that cases were rapidly growing and haven’t gone down to achieve normal distribution. When total covid-19 confirmed cases skew will be equal to 0, we can say that the situation has been normalized and it is safe to return back in person.

3.4. **Kernel Density Estimate (figure 4.9-4.11)** is used to to estimate the probability density function of the random variables to make inferences about the population based on the finite data sample that we have. Tp graph kernel density estimate we used *kde()* method in python as shown in section 6.

3.5. Correlation between attributes by using heatmap (figures 4.12-4.13).

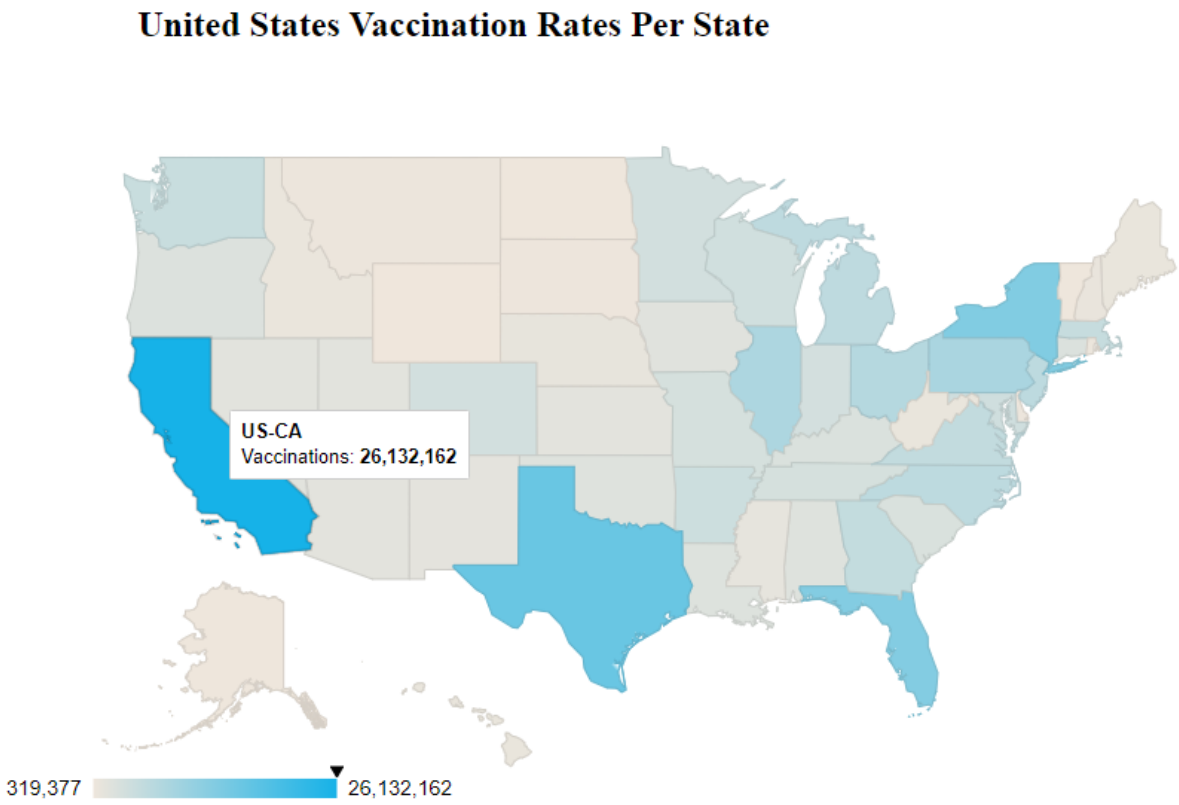
- A correlation heatmap is a visual representation of 2D matrix between two discrete dimensions where color scale defines the color for each cell

depending on the values (i.e., red and orange show states with more vaccinations than blue). The color of each cell is proportional to the number of measurements that is in the dimensional value. Correlation heatmaps are used in data analysis to identify patterns between many attributes easier since it makes the patterns more readable and highlights the differences and variation in the same data. Correlation matrix gives an indication of how related are the changes between 2 attributes. For example in our heatmap, orange and red means higher numbers of people are vaccinated, whereas blue shades represent low vaccination rates.

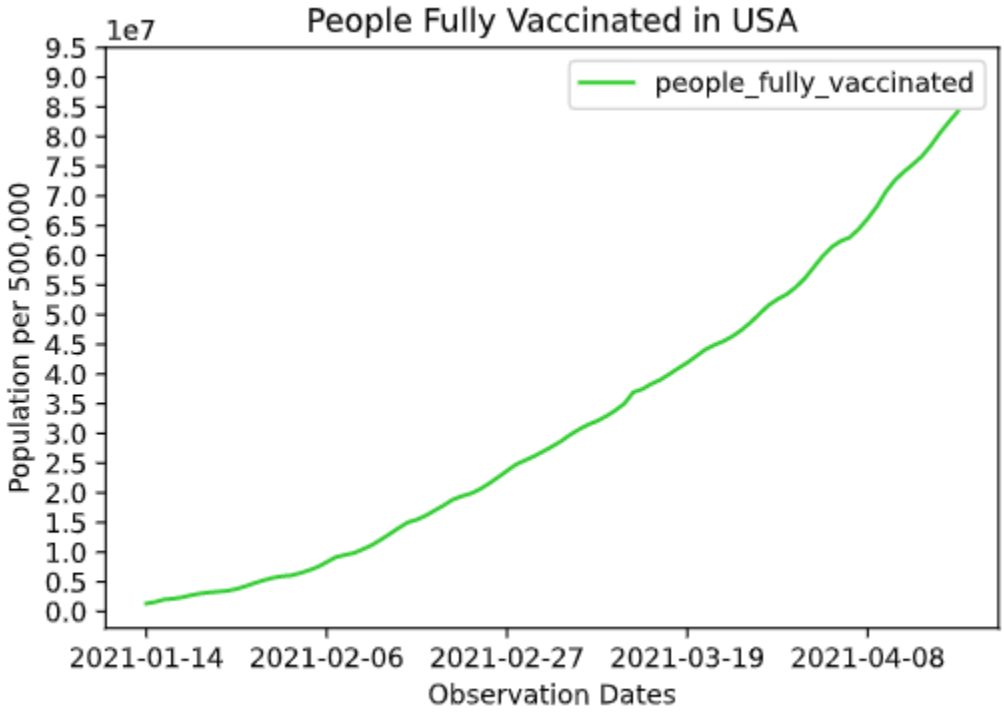
- Figure 4.12 shows a correlation matrix using *seaborn* library between vaccinations in each state and the quantity of vaccine distribution over time. We can observe that in states like California and Texas where most cells are light and dark orange, vaccinations have been occurring a lot more compared to states like Alabama where most cells are blue. As of now, California vaccinated 26 million people, therefore there is a correlation between California vaccination and some of the recent months (March and April) where more people are vaccinated and it is continuing to increase.

4. Visualization

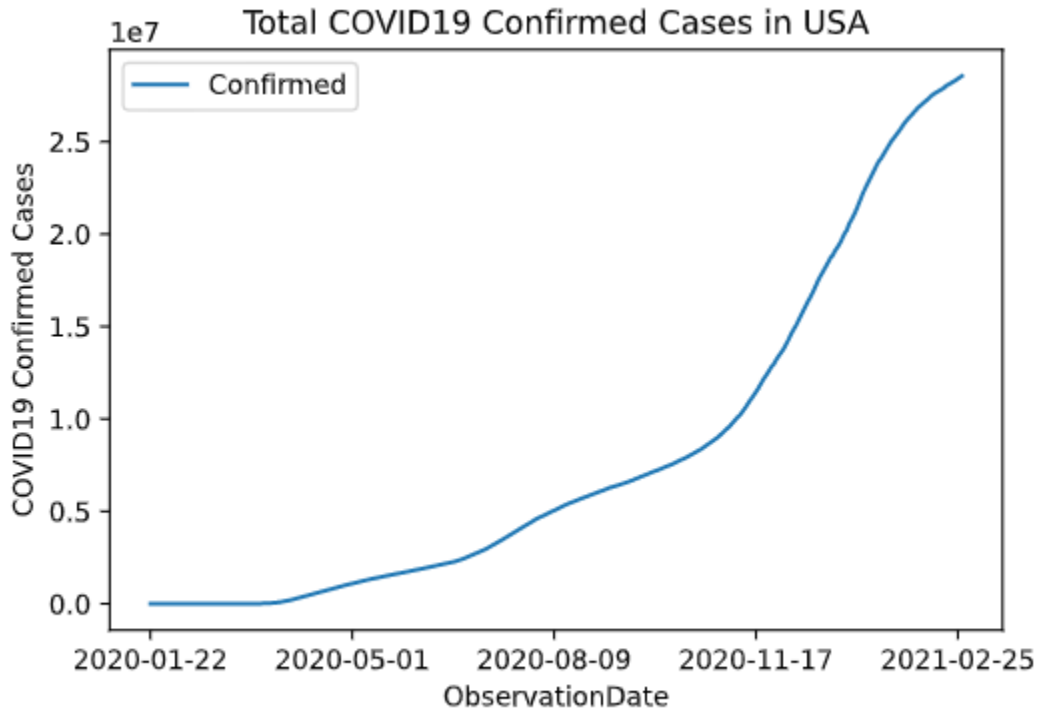
4.1. **Figure 4.1: Clusters of vaccinations per state.**



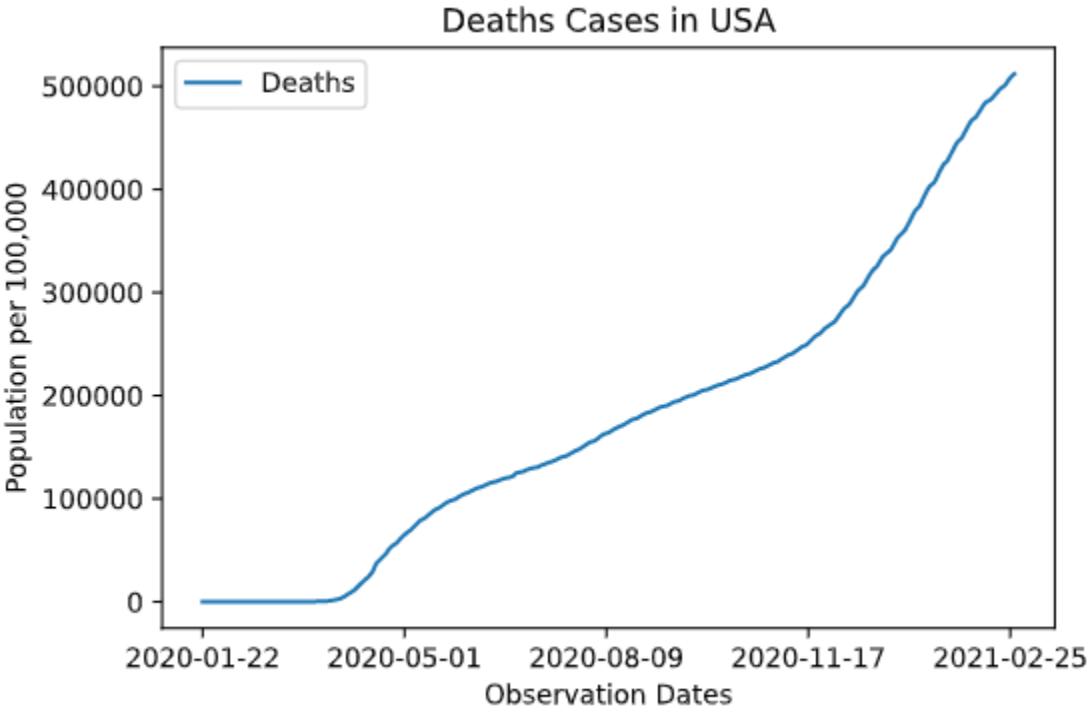
4.2. **Figures 4.2 - 4.5 are visualizations of datasets.**



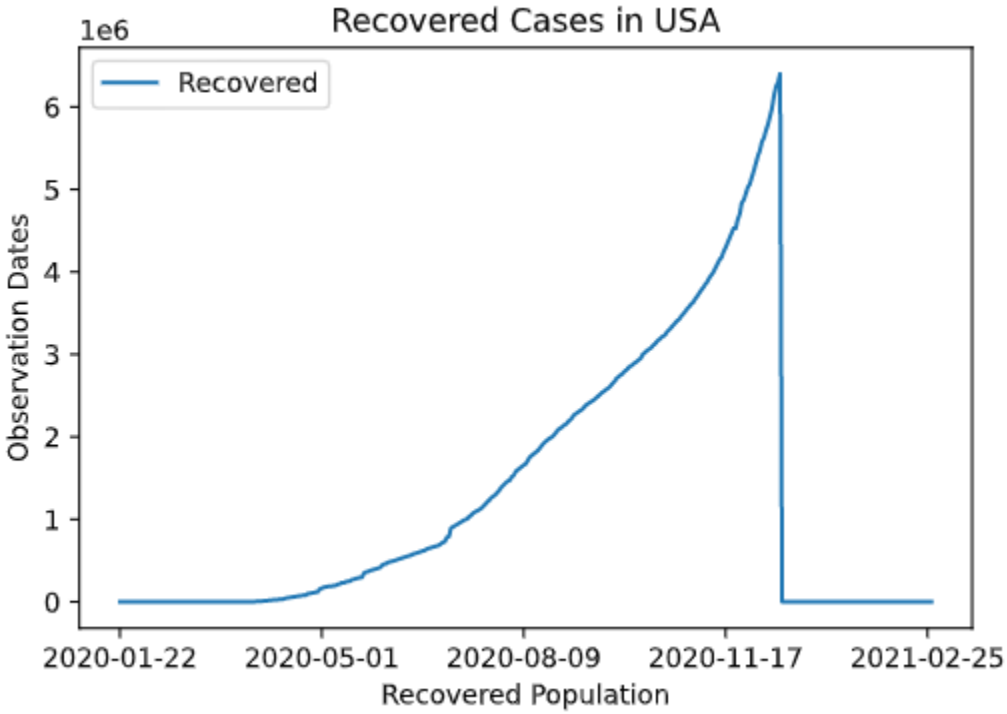
4.3.



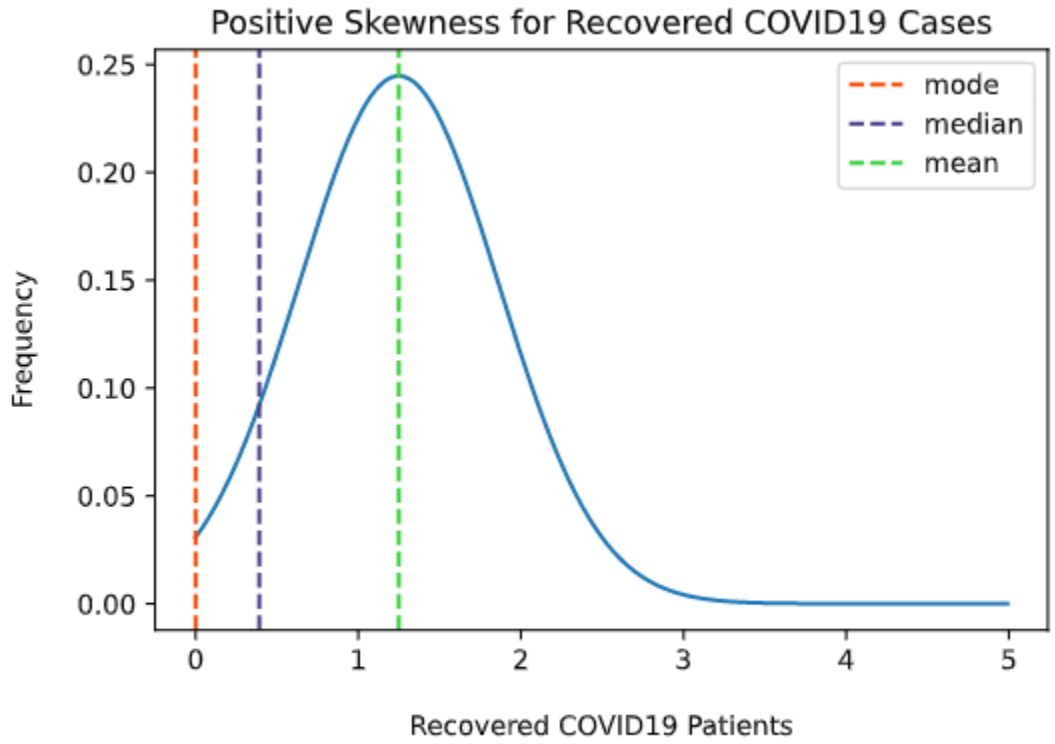
4.4.


Deaths Cases in USA
Deaths
500000
400000
300000
200000
100000
0
Population per 100,000
2020-01-22
2020-05-01
2020-08-09
2020-11-17
2021-02-25
Observation Dates

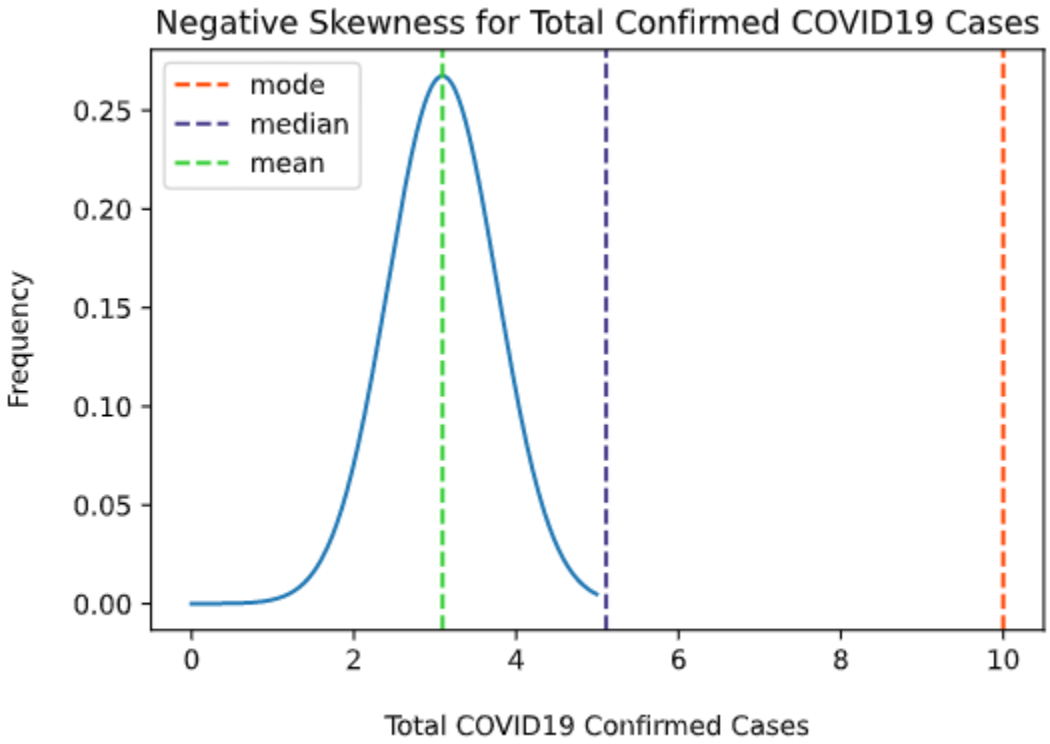
4.5.


Recovered Cases in USA
1e6
Recovered
6
5
4
3
2
1
0
Observation Dates
2020-01-22
2020-05-01
2020-08-09
2020-11-17
2021-02-25
Recovered Population

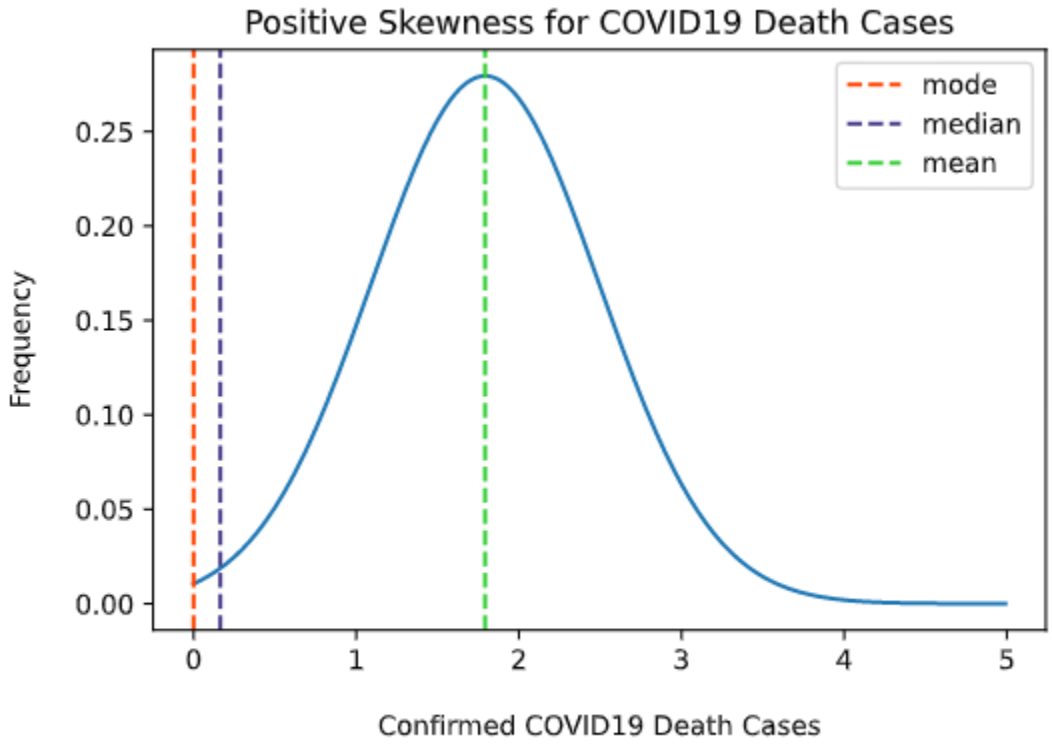
4.6. Figures 4.6 - 4.8 represent skewness of datasets.



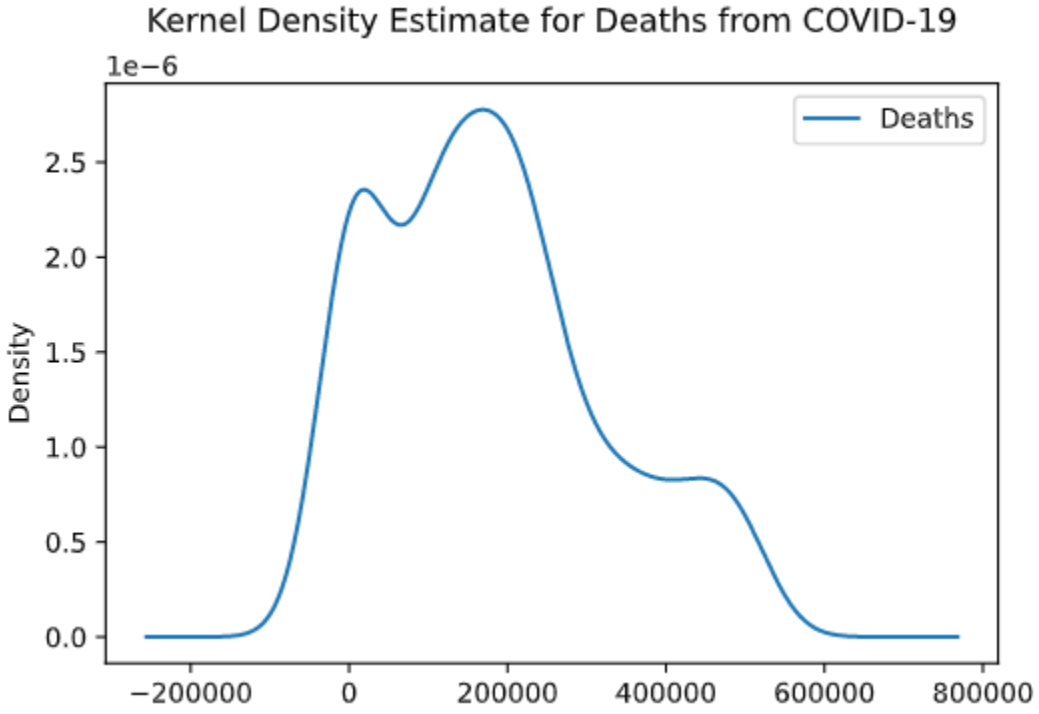
4.7.



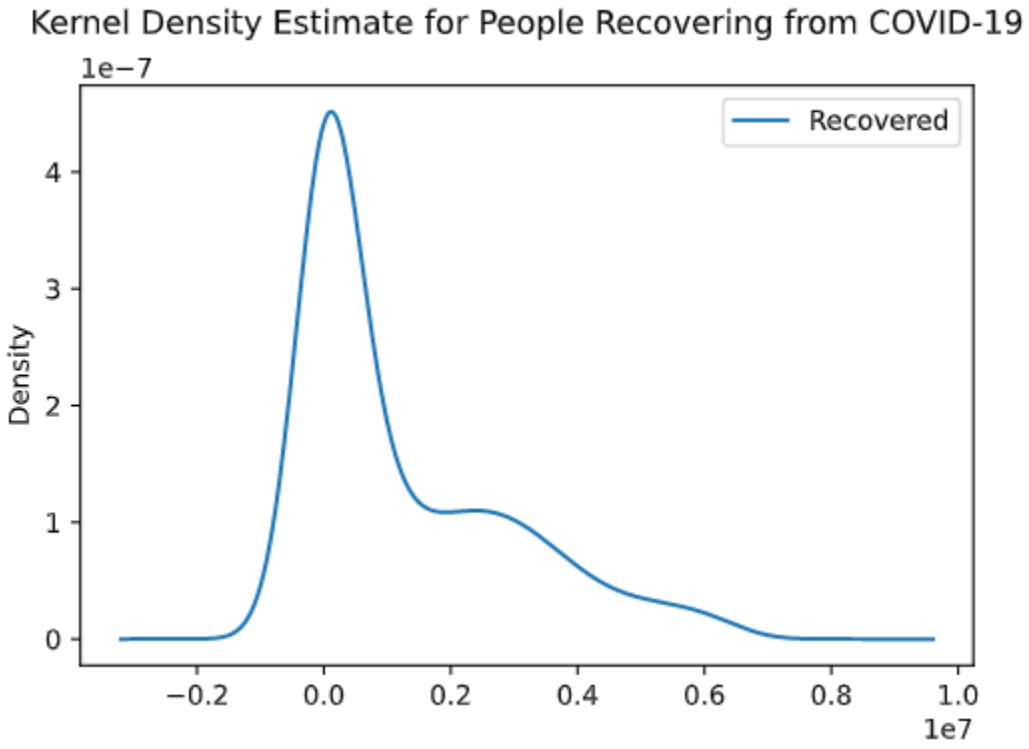
4.8.



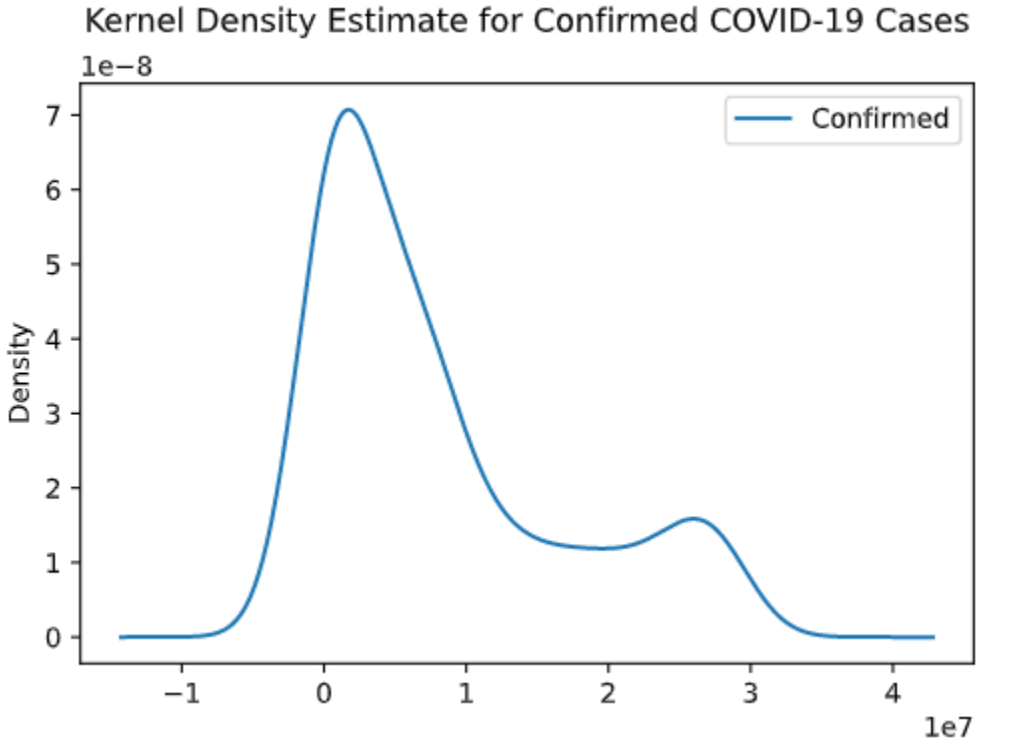
4.9. Figures 4.9-4.11 show kernel density estimates for datasets.



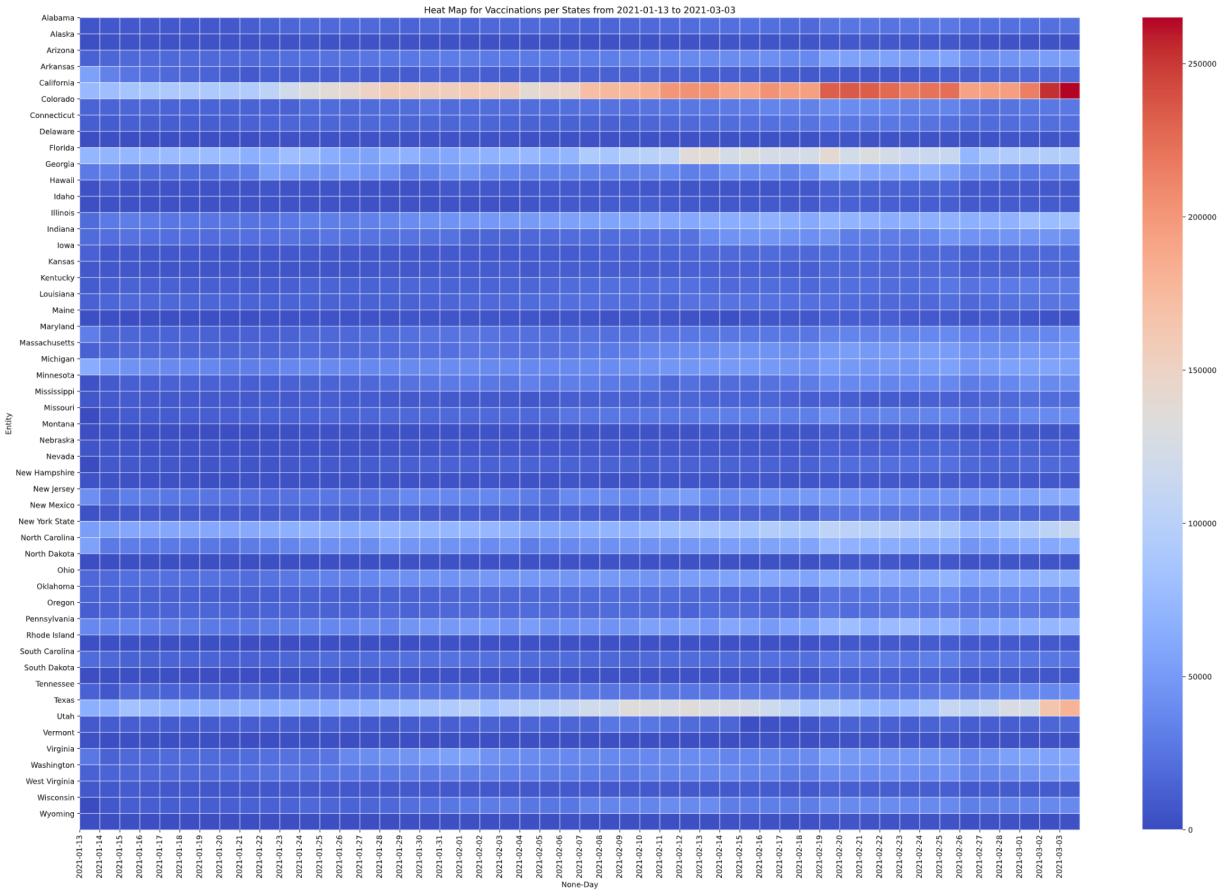
4.10.


Kernel Density Estimate for People Recovering from COVID-19
1e−7
Recovered
Density
0
1
2
3
4
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
1e7

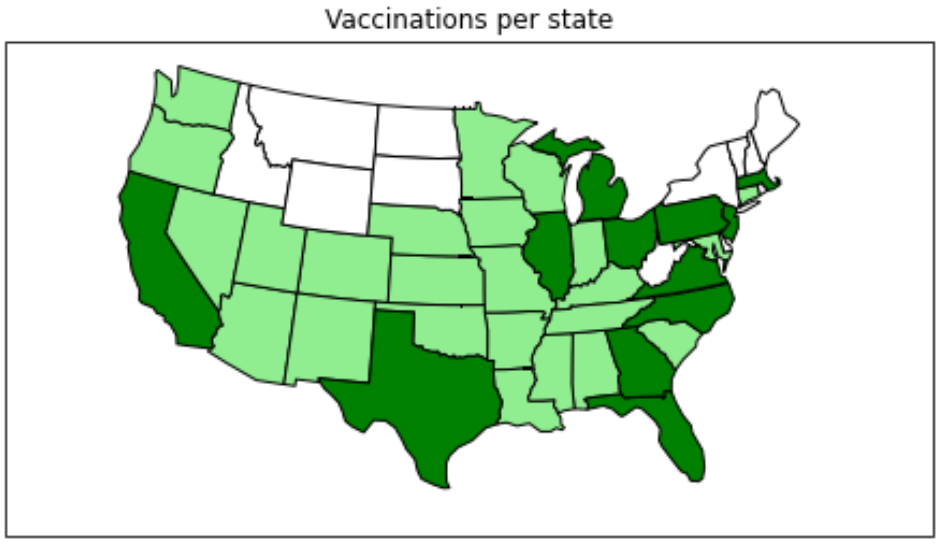
4.11.


Kernel Density Estimate for Confirmed COVID-19 Cases
1e−8
Confirmed
Density
0
1
2
3
4
5
6
7
−1
0
1
2
3
4
1e7

4.12. Figure 4.12 displays a heatmap between attributes (50 states and time periods) for vaccinations.



4.13. **Figure 4.13 is another variation of the heatmap above.**



5. Code

5.1. What we used:

- Our team wrote the code to process the datasets using the Python programming language and a number of data processing libraries such as pandas, numpy, seaborn, matplotlib and statistics. Our team is collaborating on a cloud hosted Jupyter Notebook.
- Below are the screenshots of the code, in addition with a zip file that has code and the datasets used will be attached.

5.2. Some of our screenshots from the Jupyter Notebook:

5.3. Figure 5.2 shows how we displayed the datasets:

```
from datetime import datetime

dates=list(set(covid_19_data_usa_confirmed['ObservationDate']))
dates.sort(key=lambda date: datetime.strptime(date, "%m/%d/%Y"))

confirmed_cases_usa = pd.DataFrame({'ObservationDate':[],'Confirmed':[]})
for d in dates:
    temp=covid_19_data_usa_confirmed[covid_19_data_usa_confirmed["ObservationDate"]==d]
    d=datetime.strptime(d, "%m/%d/%Y")
    d=d.strftime('%Y-%m-%d')
    confirmed_cases_usa.loc[len(confirmed_cases_usa.index)] = [d, temp['Confirmed'].sum()]

plot = confirmed_cases_usa.plot(x='ObservationDate',y='Confirmed',kind='line',xticks=list(range(0,500,100)),title='Total COVID19 Confirmed Cases in USA')
plot.set_ylabel("COVID19 Confirmed Cases")
```

5.4. Figures 5.4 and 5.5 show how we got the std and mean values of the datasets.

5.5.

```
# to get the std and percentiles
covid_19_data_usa_confirmed.describe().apply(lambda x: '%.5f' % x, axis=1)
```

```
count      21462.00000
mean      149160.90355
std       309144.87948
min            0.00000
25%         2710.25000
50%        35865.50000
75%       156865.25000
max      3563578.00000
dtype: object
```

5.6.

```
# to get std and percentiles
recovered_cases_usa.describe().apply(lambda x: '%.5f' % x, axis=1)
```

```
count        403.00000
mean     1249059.44417
std      1630170.41782
min            0.00000
25%            3.00000
50%       391508.00000
75%      2292820.50000
max      6399531.00000
dtype: object
```

5.7. Figure 5.7 shows how we got median and mode for the skewness.

```
# Median
# 5.1, 0.16, 0.3
from statistics import median, mode
print(median(confirmed_cases_usa['Confirmed']))
print(median(deaths_cases_usa['Deaths']))
print(median(recovered_cases_usa['Recovered']))
```

```
5107195.0
164041.0
391508.0
```

5.8.

5.9. Figures 5.9 and show how we graphed skewness of the datasets.

```
# Total Confirmed COVID19 Cases
# Negative Skew
# mean, median, mode
mean = 3.091449; std = 1.491609; median = 5.107195; mode = 10; variance = np.square(std)
x = np.arange(0,5, .01)
f = np.exp(-np.square(x-mean)/2*variance)/(np.sqrt(2*np.pi*variance))

plt.plot(x,f)
plt.axvline(x=mode, linestyle='--', linewidth=1.5, label='mode', c='orangered')
plt.axvline(x=median, linestyle='--', linewidth=1.5, label='median', c='darkslateblue')
plt.axvline(x=mean, linestyle='--', linewidth=1.5, label='mean', c='limegreen')
plt.title('Negative Skewness for Total Confirmed COVID19 Cases')
plt.xlabel('Total COVID19 Confirmed Cases', labelpad=15)
plt.ylabel('Frequency', labelpad=15)
plt.legend()
plt.show()
```

5.10.

```
# Recovered COVID19 - Positive Skewness
# mode, median, mean
mean = 1.249059; std = 1.630170; median =  0.391508; mode = 0.0; variance = np.square(std)
x = np.arange(0,5,.01)
f = np.exp(-np.square(x-mean)/2*variance)/(np.sqrt(2*np.pi*variance))

plt.plot(x,f)
plt.axvline(x=mode, linestyle='--', linewidth=1.5, label='mode', c='orangered')
plt.axvline(x=median, linestyle='--', linewidth=1.5, label='median', c='darkslateblue')
plt.axvline(x=mean, linestyle='--', linewidth=1.5, label='mean', c='limegreen')
plt.title('Positive Skewness for Recovered COVID19 Cases')
plt.xlabel('Recovered COVID19 Patients', labelpad=15)
plt.ylabel('Frequency', labelpad=15)
plt.legend()
plt.show()
```

5.11.

5.12. Figure 5.12 shows kde() method in python to graph kernel density of datasets (death cases example):

```
plot = deaths_cases_usa.plot.kde()
plot.set_ylabel("Density")
plot.set_title("Kernel Density Estimate for Deaths from COVID-19\n")
```

5.13.

5.14. Figure 5.14 shows seaborn library being used to graph heatmap for vaccination dataset:

```
import seaborn as sns

data= data_cleaned.iloc[:, : 50]
ax = sns.heatmap( data , linewidth =0.5 , cmap = 'coolwarm',cbar_kws={'ticks': list(range(0,300000,50000))})
fig = plt.gcf()
figsize = fig.get_size_inches()
fig.set_size_inches(figsize * 5)
dates=[]
for col in data.columns:
    dates.append(col[1])
plt.yticks(list(range(0,50)),states)
plt.xticks(list(range(0,50)),dates)
plt.title( "Heat Map for Vaccinations per States from "+dates[0]+" to "+dates[len(dates)-1])
plt.show()
```

5.15.

5.16. Figure 5.17 shows interactive map about vaccinations datasets in Javascript using Map API:

```
<script type="text/javascript" src="https://www.gstatic.com/charts/loader.js"></script>
<script type="text/javascript">
    google.charts.load('current', {
        'packages': ['geochart'],
        // keys
        'mapsApiKey': ''
    });
```

5.17.