# Covid 19 US cases analysis

Authors: Dang Le and Vasu Dadhania
SOEN 6111 - Winter 20201
Present to Dr. Tristan Glatard

In this notebook, we will present the steps and results that are used to analyze COVIS 19 cases data set from New York Times. This dataset is updated daily on github and therefore represent an accurate number of confirmed cases in US counties over time.

# Abstract

The novel coronavirus (COVID-19) pandemic presents a severe threat to human health worldwide. The United States (US) has the highest number of reported COVID-19 cases, and over 16 million people were infected. To better understand and mitigate the spread of the disease, it is necessary to recognize the pattern of the outbreak. In this study, we explored the patterns of COVID-19 cases for the last 100 days. The county-level cases and rates of the disease were mapped using a geographic information system (GIS). The overall trend of the disease in the US, as well as in each of its 50 individual states, were analyzed by the seasonal-trend decomposition. The disease curve in each state was further examined using K-means clustering and hierarchical clustering. The results from this study provide insight in making disease control and mitigation strategies.

# Introduction

In this section, we will present the objective of our project as well as a quick presentation of the dataset.

## Objective

The novel coronavirus disease (COVID-19), first reported in Wuhan, China, presents a severe threat to global health. The disease caused by the novel severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2) is highly contagious and spreads rapidly. The most common transmission pathway is through face-to-face exposure of expelled droplets during coughing, sneezing or talking . Fever, cough and tiredness are common signs and symptoms of the infection. The symptoms of patients with the disease can be mild or severe, even deadly. The World Health Organization (WHO) declared the disease a global pandemic on 11 March 2020,

due to its infectivity and severity. By 12 December 2020, the disease had infected more than 72 million people and led to over 1.6 million deaths worldwide.
Since the first case was reported in late January 2020, the United States (US) has been suffering the impact of COVID-19 severely . Up to Mid-December 2020, the US reported more than 16 million cases, which is the highest in the world by country. The reported daily new cases vary greatly in space and time. Therefore, understanding the patterns of the disease occurrence is very important, because it helps to evaluate the effectiveness of mitigating strategies and policies in the past, and allocate new resources and implement updated control measures, as well as predict the future trend .

The objective of this study is to identify the patterns of the COVID-19 cases in the US. Specifically, we attempt to answer the following questions: What states have high risk trend of COVID-19 cases? What is the similarity of disease curves among these states? We applied time series analysis and K-means clustering analysis and hierarchical clustering to identify these patterns.

## Problem Presentation

The dataset we are using here is taken every day in 2020, thus the appropriate technique to use is time-series clustering, which is considered as clustering of a set of individual time series with respect to their similarity.

Recognizing a pattern is a process of discovering the regularities in data with statistical or machine learning algorithms and using these regularities to conduct further analysis such as clustering. For this analysis we have to use some unsupervised learning techniques because dataset is not labeled, and we have chosen time series clustering techniques to separate data point with their regarding features and also we take the dataset of the population of each county with the help of unique FISP (Federal Information Processing System)

# Materials and methods

In this section, we will introduce the data used in this experiement and also the clustering methods used, which are K-Means and Hierachical Clustering.

## Data preliminary analysis

First we would like to introduce the dataset, which can be obtained from https://raw.githubusercontent.com/nytimes/covid-19-data/master/us-counties.csv.

59

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
```

```
from sklearn.metrics import silhouette_score, davies_bouldin_score
from sklearn.cluster import AgglomerativeClustering
from sklearn.cluster import KMeans
import plotly.graph_objects as go
```

60

```
import json
from urllib.request import urlopen
with urlopen('https://raw.githubusercontent.com/plotly/datasets/master/geojson-counties-fips.json') as re
    counties = json.load(response)
```

41

```
total_live_data = "https://raw.githubusercontent.com/nytimes/covid-19-data/master/us-counties.csv"
liveData = pd.read_csv(total_live_data,  dtype={"fips": str})
liveData.head(5)
```

41

| | date | county | state | fips | cases | deaths |
|---|---|---|---|---|---|---|
| **0** | 2020-01-21 | Snohomish | Washington | 53061 | 1 | 0.0 |
| **1** | 2020-01-22 | Snohomish | Washington | 53061 | 1 | 0.0 |
| **2** | 2020-01-23 | Snohomish | Washington | 53061 | 1 | 0.0 |
| **3** | 2020-01-24 | Cook | Illinois | 17031 | 1 | 0.0 |
| **4** | 2020-01-24 | Snohomish | Washington | 53061 | 1 | 0.0 |

42

```
print(f'Number of rows are {len(liveData)}')
print(f'Number of counties are {len(liveData.county.unique())}')
print(f'Number of distinct dates are {len(liveData.date.unique())}')
print(f'Number of distinct fips are {len(liveData.fips.unique())}')
```

```
Number of rows are 1212589
Number of counties are 1930
Number of distinct dates are 447
Number of distinct fips are 3219
```

As shown above, the data we are using have 6 columns:

- **Date**: represent record date
- **County**: name of the county that report the case
- **State**: name of the state that this county belong
- **Fips**: represents the Federal Information Processing Standards number that is unique for each county
- **Cases**: record the number of confirmed cases
- **Deaths**: is the number of confirmed dead fromCOVID in each county

There are over 1 millions rows of data given, with 1,930 counties reporting covid cases over the time period of 441 days. Furthermore, we can see that there are over three thousands fips numbers, which is almost twice as many counties, meaning that each county can be broken further down into smaller regions represent by fips.

Because the given data is already sorted by date, we can see that Washing was the first state reported confirmed case, which is quite interesting. Furthermore, the data is very sparse in the early stage of the time interval, thus making it quite difficult for clustering because there are not enough information. Thus, we decided to transform the data into matrix form, with the rows represent each fips, the column represent the date and the value is number of confirmed cases for each county at a given day.

To start this transformation, we first observe that we only need 3 columns, date, fips and cases, because we can deduce the state and county from any given fips. Also the deaths confirmed number contains many zero, so we decided to drop that column.

```
43  covidData = liveData[['date','fips','cases']]
    groupByFips = covidData.set_index('date').groupby('fips')
```

```
44  groupByFips.cases.mean().describe()
```

```
44  count      3218.000000
    mean       3392.817402
    std       11981.411941
    min           1.000000
    25%         376.244668
    50%         874.886937
    75%        2222.621812
    max      393482.674208
    Name: cases, dtype: float64
```

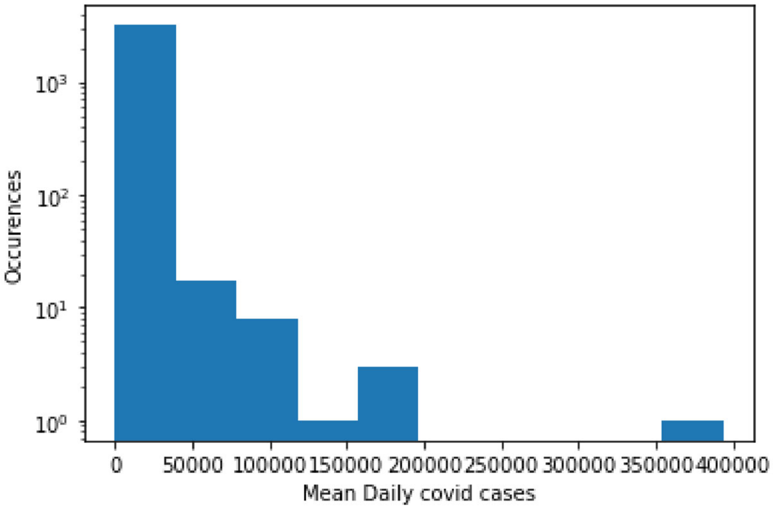
After grouping the data by fips number, we calculate the means value of cases reported and call the describe method to find some statistics. Looking the count of 3218 which equals to the number of fips we find above and confirm that the group by is correct.

Looking at the mean and standard deviation value, we can see that the data is more spread out, which indicate that clustering technique would be useful in this application to identify and group similar data points. Furthermore, from the distribution information, we can see that there are abnormally, where some group have mean value of only 1 cases, and some having over 300,000 cases.

```
45  plt.hist(list(groupByFips.cases.mean()))
    plt.rcParams['axes.facecolor'] = 'white'
    plt.ylabel('Occurences')
    plt.xlabel('Mean Daily covid cases')
    plt.yscale('log', nonposy='clip')
```

```
<ipython-input-45-a1c7b9b46d05>:5: MatplotlibDeprecationWarning:

The 'nonposy' parameter of __init__() has been renamed 'nonpositive' since Matplotlib 3.3; support for th
```



Using the histogram graph, we can see that the majority of counties have mean COVID-19 cases of less than 75,000. However, there are regions with significantly more daily cases, and we suspect these regions to be highly populated, however we will only be confirmed this hypothesis when doing the clustering. Also note that the graph is converted to log scale for cleaner representation.

# Clustering

In the section, we will discuss our implementation of K-Means and Hierarchical Clustering to the presented dataset above.

However, before we proceed with the clustering techniques, we need to identify how many clusters is suitable for our dataset. In order to do so, we pick out a sample data set to run k means clustering on, with k value set from 1 to 30 to find the best value. The number of clusters that optimizes the function convergence to the centroids, we plotted the elbow function which, paired with epidemiological knowledge from the dataset, supported the choose of five and six clusters. That is, five and six cluster classified. Although five and six clusters provided similar groups. Overall, the function cost (elbow plot, Figure 1), paired with the overall results (boxplots and maps), suggested that five or six cluster were a sensitive decision.

The Elbow method is one of the most popular ways to find the optimal number of clusters. This method uses the concept of WCSS value. WCSS stands for Within Cluster Sum of Squares, which defines the total variations within a cluster. The formula to calculate the value of WCSS (for 3 clusters) is given below:

WCSS= ∑Pi in Cluster1 distance(Pi C1)2 +∑Pi in Cluster2distance(Pi C2)2+ ∑Pi in CLuster3 distance(Pi C3)2

```
elbowTestData = groupByFips['cases'].apply(list).reset_index(name = 'new')
```

46

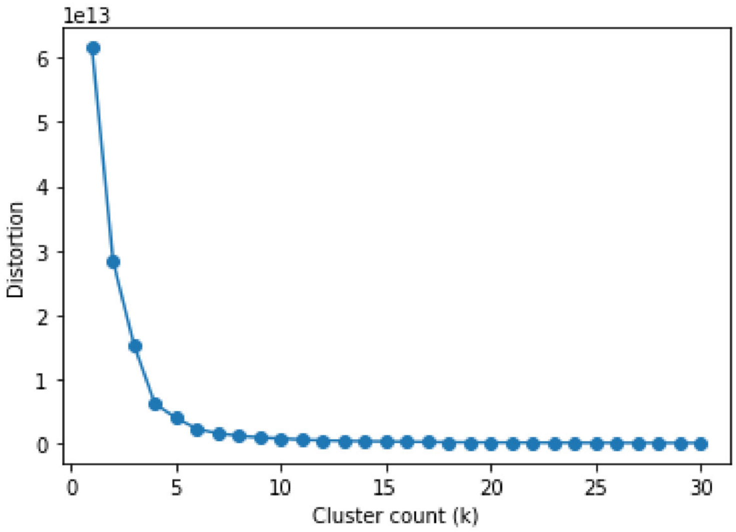
47

```
fips = list(groupByFips.fips.groups.keys())
resultgroupBy = groupByFips['cases'].apply(list).reset_index(name = 'new')
elbowData = resultgroupBy['new'].apply(lambda r: r[-100::5]).to_list()
```

48

```
distortions = []
max_k = 30
for i in range(1, max_k + 1):
    km = KMeans(n_clusters = i, init = 'k-means++', max_iter =300, n_init = 10, random_state = 0)
    km.fit(elbowData)
    distortions.append(km.inertia_)

plt.plot(range(1, max_k + 1), distortions, marker='o')
plt.xlabel('Cluster count (k)')
plt.ylabel('Distortion')
plt.show()
```



Using the elbow graph presented above, we can see that the best number of cluster should be between 6 and 8, therefore we decide to pick 7 as our number of clusters in both experiments.

## K Means Clustering

We applied K-means clustering to explore the pattern of disease curves in different states. K-means clustering is a simple and common unsupervised machine learning algorithm for exploratory data analysis to get an intuition about data structure. It divides these states into K subgroups (clusters). The states with a similar disease curve will be grouped into the same cluster. The K-means algorithm works as below:

1. Specify the number of clusters (K);
2. Initialize K centroids and calculate the distance between each centroid and each data point;

3. The data points that have the shortest distance to a centroid are grouped in the same cluster;
4. Calculate the new centroid based the data points in a cluster;
5. Repeat the process from (2) to (4) until the centroids are stable.

To find the K we used elbow method described as above.

49

```
testData = resultgroupBy['new'].apply(lambda r: r[-100:]).to_list()
import numpy as np
result = np.array(testData)

km = KMeans(n_clusters=7, init='k-means++', n_init=10, max_iter=300, random_state=0)
km.fit(result)
KmeansfinalDF = pd.DataFrame(data = fips)
KmeansfinalDF = KmeansfinalDF.rename(columns = {0:'fips'})
KmeansfinalDF['classification'] = km.labels_
silihouette = silhouette_score(testData, km.labels_, metric='euclidean')
print("silihouette_score "+str(silihouette))
davies_bouldin = davies_bouldin_score(testData, km.labels_)
print("davies_bouldin_score "+str(davies_bouldin))
```

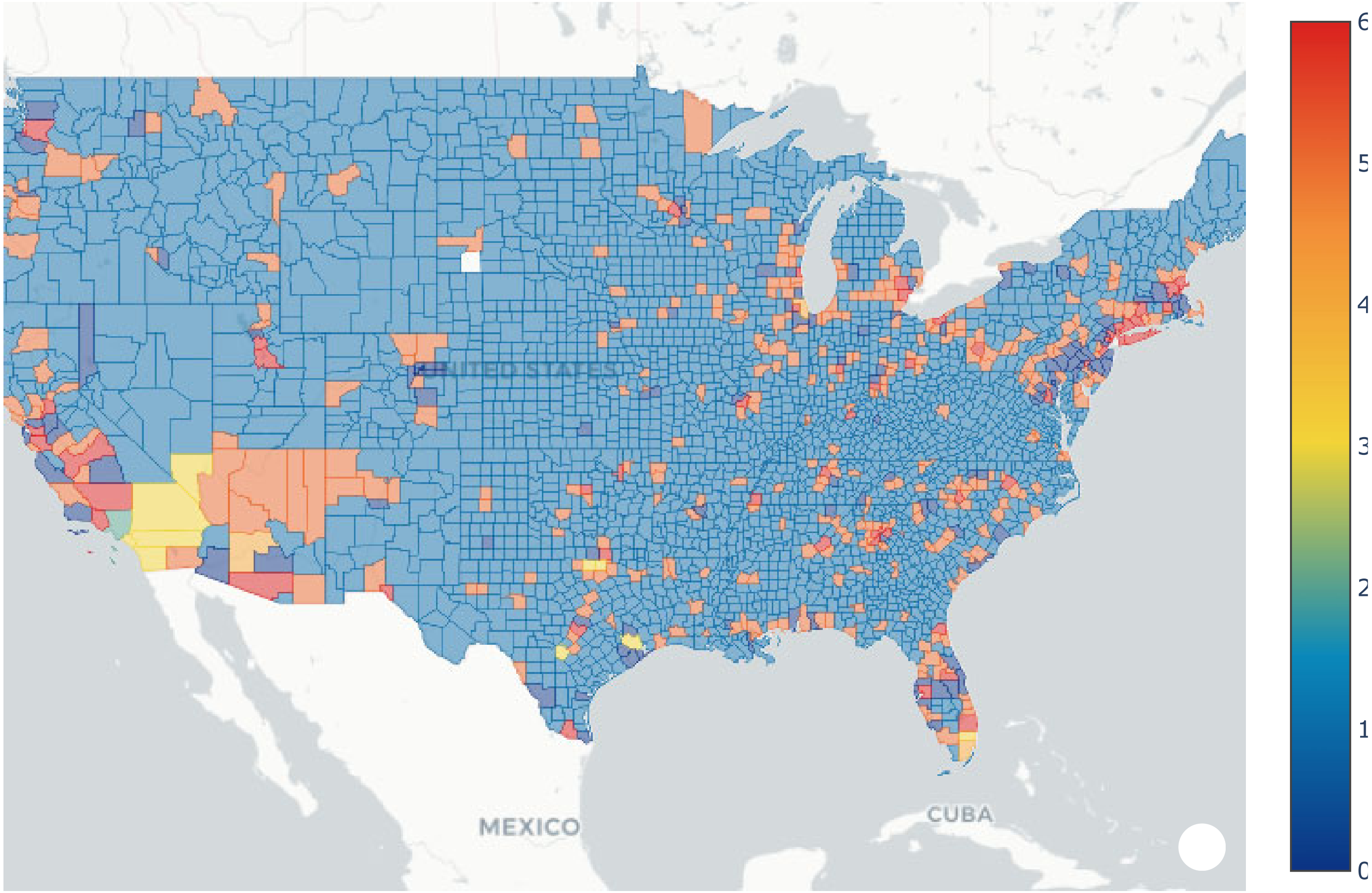
```
silihouette_score 0.76626110539648
davies_bouldin_score 0.3998848294417688
```

With the K means clustering algorithm, we are able to calculate the Silihouette and Davies Bouldin score, which confirm that the algorithm is working right. However, it would also be useful if we can see the result plot onto an US map.

61

```
print('Begin plotting')
fig = go.Figure(go.Choroplethmapbox(geojson=counties, locations=KmeansfinalDF['fips'], z=KmeansfinalDF['c
                                    colorscale="Portland", zmin=0, zmax=6,
                                    marker_opacity=0.5, marker_line_width=0))
fig.update_layout(mapbox_style="carto-positron",
                  mapbox_zoom=3, mapbox_center = {"lat": 37.0902, "lon": -95.7129})
fig.update_layout(margin={"r":0,"t":0,"l":0,"b":0})
fig.show()
```

```
Begin plotting
```



# Hierarchical Clustering

Similar to the k means clustering above, we then repeat the same experiment using Hierarchical clustering technique. Hierarchical clustering is another unsupervised machine learning algorithm, which is used to group the unlabeled datasets into a cluster and also known as hierarchical cluster analysis or HCA. In this algorithm, we develop the hierarchy of clusters in the form of a tree, and this tree-shaped structure is known as the dendrogram.Sometimes the results of K-means clustering and hierarchical clustering may look similar, but they both differ depending on how they work.

51

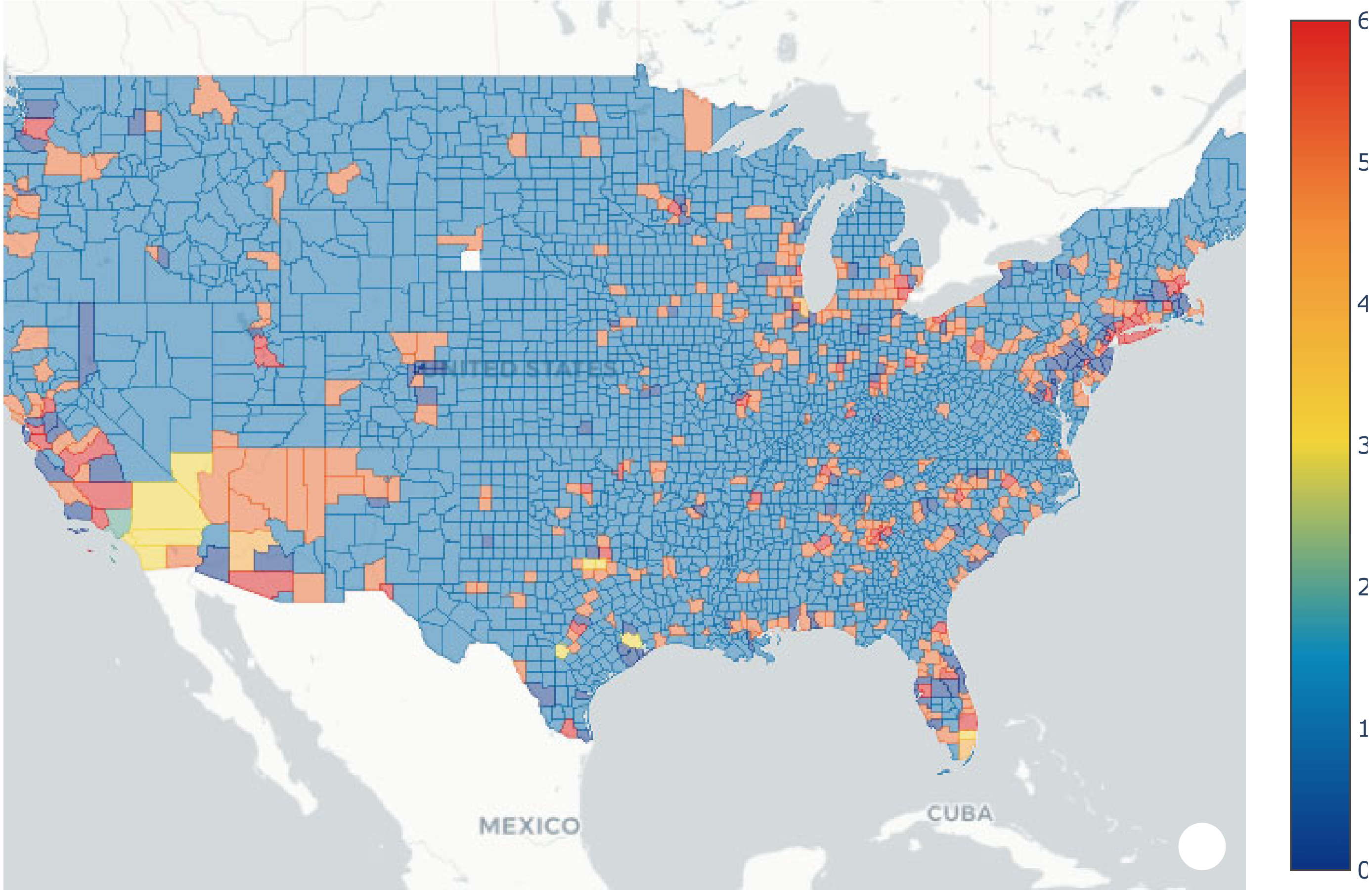
```
he =AgglomerativeClustering( n_clusters=7)
he.fit(result)
HEsfinalDF = pd.DataFrame(data = fips)
HEsfinalDF = HEsfinalDF.rename(columns = {0:'fips'})
HEsfinalDF['classification'] = km.labels_
silihouette = silhouette_score(testData, he.labels_, metric='euclidean')
print("silihouette_score "+str(silihouette))
davies_bouldin = davies_bouldin_score(testData, he.labels_)
print("davies_bouldin_score "+str(davies_bouldin))
```

```
silihouette_score 0.7334038369045663
davies_bouldin_score 0.4364573061972635
```

We then plot the result onto a US map.

```
52  print('Begin plotting')
    fig = go.Figure(go.Choroplethmapbox(geojson=counties, locations=HEsfinalDF['fips'], z=HEsfinalDF['classif
                                        colorscale="Portland", zmin=0, zmax=6,
                                        marker_opacity=0.5, marker_line_width=0))
    fig.update_layout(mapbox_style="carto-positron",
                      mapbox_zoom=3, mapbox_center = {"lat": 37.0902, "lon": -95.7129})
    fig.update_layout(margin={"r":0,"t":0,"l":0,"b":0})
    fig.show()
```

```
Begin plotting
```



# Results

In this section, we will discuss the results obtained from the experiment above, in which we will determine which clustering methods is better for this application and also the clusteres obtained.

## Analysis

First we would like to compare the correctiveness of the two clustering techniques, using the Silihouette and Davies Bouldin score.

## Silihouette Score

According to scikit-learn website, Silihouette score is calculated using the mean intra-cluster distance (a) and the mean nearest-cluster distance (b) for each sample. The best value is 1 and worst value is -1, with value near 0 indicates overlapping clusters. The Silhouette Coefficient for a sample is (b - a) / max(a, b). To clarify, b is the distance between a sample and the nearest cluster that the sample is not a part of.
Looking at our result from above, K-means seems to be a better algorithm, with Silihouette score of 0.7651, while the Hierarchical have the score of 0.7468.

## Davies Bouldin Score

Next we will analyze the two result using Davies Bouldin, which measure the by the average similarity measure of each cluster with its most similar cluster, where similarity is the ratio of within-cluster distances to between-cluster distances.The score is defined as the average similarity measure of each cluster with its most similar cluster, where similarity is the ratio of within-cluster distances to between-cluster distances. Thus, clusters which are farther apart and less dispersed will result in a better score.
The minimum score is 0, with lower value indicate better clustering. The score we obtain for K mean is 0.4032, while the score for Hierarchical is 0.4329. This result again confirms that K Means is a better algorthm between the two.

Because the given data is not extremely large, comparing the run time of the two algorithms is not feasible, thus we would like to keep this for future work, where dataset size is more suited for performance conparision.
Next, we would like to point out some interesting facts from our result above.

## Clustering Result

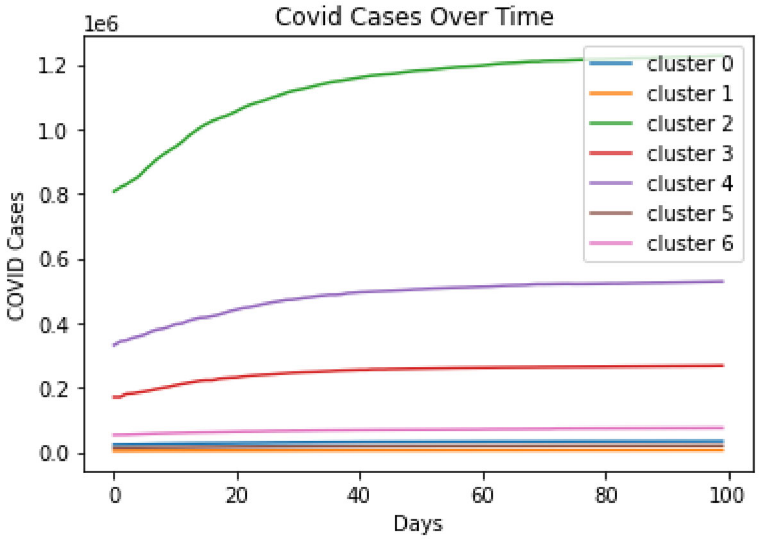
First of all, the results seems to be very similar, which confirm the facts even though K means is a more effective algorithm, Hierarchical is still correct and product reliable result.

Next, we can see that highly populated areas such as Los Angeles, New York and Miami are grouped into similar cluster, which confirm our hypothesis that cities with high population are more likely to have similar COVID 19 progression. However, to further valid our hypothesis and see which cluster can be classified as high risk versus low risk, we would take randomly one sample from each cluster and graph it to see the progression over time.

53

```
sample = {}
fipstList = {}
for i in range (0,7):
    index = HEsfinalDF[HEsfinalDF['classification'] == i].iloc[0]['fips']
    sample[i] = resultgroupBy[resultgroupBy['fips'] == index].iloc[0]['new'][-100:]
    fipstList[i] = index
```

54
```
x = list(range(0, 100))
import matplotlib.pyplot as plt
for i in range (0,7):
    plt.plot(x, sample[i], label = f"cluster {i}")
plt.xlabel('Days')
# Set the y axis label of the current axis.
plt.ylabel('COVID Cases')
plt.title('Covid Cases Over Time ')
# show a legend on the plot
plt.legend()
# Display a figure.
plt.show()
```



55
```
for key, value in fipstList.items():

    temp = liveData[liveData['fips'] == value].iloc[0]
    countyName = temp['county']
    stateName = temp['state']
    print(f'Cluster {key} county is {countyName} in {stateName}')
```

```
Cluster 0 county is Madison in Alabama
Cluster 1 county is Autauga in Alabama
Cluster 2 county is Los Angeles in California
Cluster 3 county is Orange in California
Cluster 4 county is Maricopa in Arizona
Cluster 5 county is Baldwin in Alabama
Cluster 6 county is Jefferson in Alabama
```

After plotting out the cases progression over time, we can see that cluster 2 has the highest number of daily covid cases, follow by cluster 3 and 4. Cluster 6 can be considered as medium risk, where as cluster 1 and 5 is relatively low risk.
Furthermore, looking at the counties that belong to each cluster, it validates the hypothesis that counties with high and dense population is more likely to transmit COVID-19.

56
```
HEsfinalDF['classification'].value_counts()
```

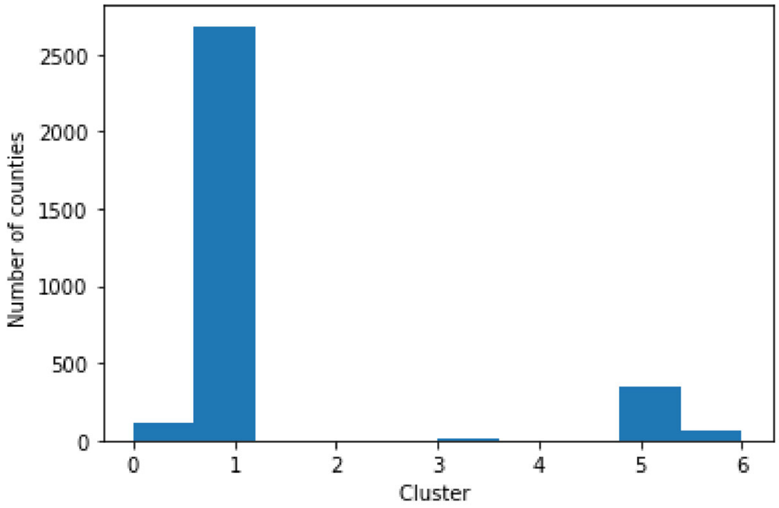
56
```
1    2684
```

```
5     348
0     116
6      56
3      10
4       3
2       1
Name: classification, dtype: int64
```

However, looking at the value counts, we can see that most of US counties can be considered as low risks, within cluster 1 and 5, with only 1 counties considered as maximum risk level, which is Los Angeles.

Finally, to show a more informative insight about our result, we will present a histogram shows number of counties in each cluster.

57
```
plt.hist(HEsfinalDF['classification'])
plt.rcParams['axes.facecolor'] = 'white'
plt.ylabel('Number of counties ')
plt.xlabel('Cluster ')
# plt.yscale('log', nonposy='clip')
plt.show()
```



# Discussion

This study has a few limitations. First, we only obtained COVID-19 data at the county level. Therefore, we cannot explore clusters at smaller spatial units. Second, K-means clustering has its own weakness. For example, the linear separability of data and the selection of K value are likely to influence the results of clustering. Finally, we did not investigate risk factors or covariates for COVID-19. Therefore, it is difficult to accurately interpret the patterns observed. Further studies to explore factors associated with the pattern of the diseases are needed.

# Conclusion

Overall, this project have successfully demonstrated the ability and versatile of clustering technique in analyzing the COVID-19 dataset provided from New York Times. We are able to identify regions with high, medium and low risks and therefore would make a good vaccination plan, where high risks areas are prioritized, then medium and low risks regions would follow.

Furthermore, as a future work, it would be interesting to repeat the same experiment with different time interval to identify regions that move from one cluster to another. This would identify states/counties with good preventive measurement in place and therefore allows stakeholders to make an effective preventative plan.
Also, we can cluster regions based on number of confirmed death and see if there are regions that belong to high risk cluster in terms of daily cases but lower risk in terms of death. This would mean that they have an effective treatment plan for infected patients.

# Reference

- https://scikit-learn.org/stable/modules/generated/sklearn.metrics.silhouette_score.html
- https://scikit-learn.org/stable/modules/generated/sklearn.metrics.davies_bouldin_score.html
- https://plotly.com/python/county-choropleth/#fips-and-values
- https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7308996/
- https://www.javatpoint.com/k-means-clustering-algorithm-in-machine-learning
- https://www.javatpoint.com/hierarchical-clustering-in-machine-learning
- https://www.scikit-yb.org/en/latest/api/cluster/elbow.html#:~:text=K%2Dmeans%20is%20a%20simple,number%20(k)%20of%20clusters.&text=The%20elbow%20method%20runs%20k,average%20score%20for%20all%20clusters.