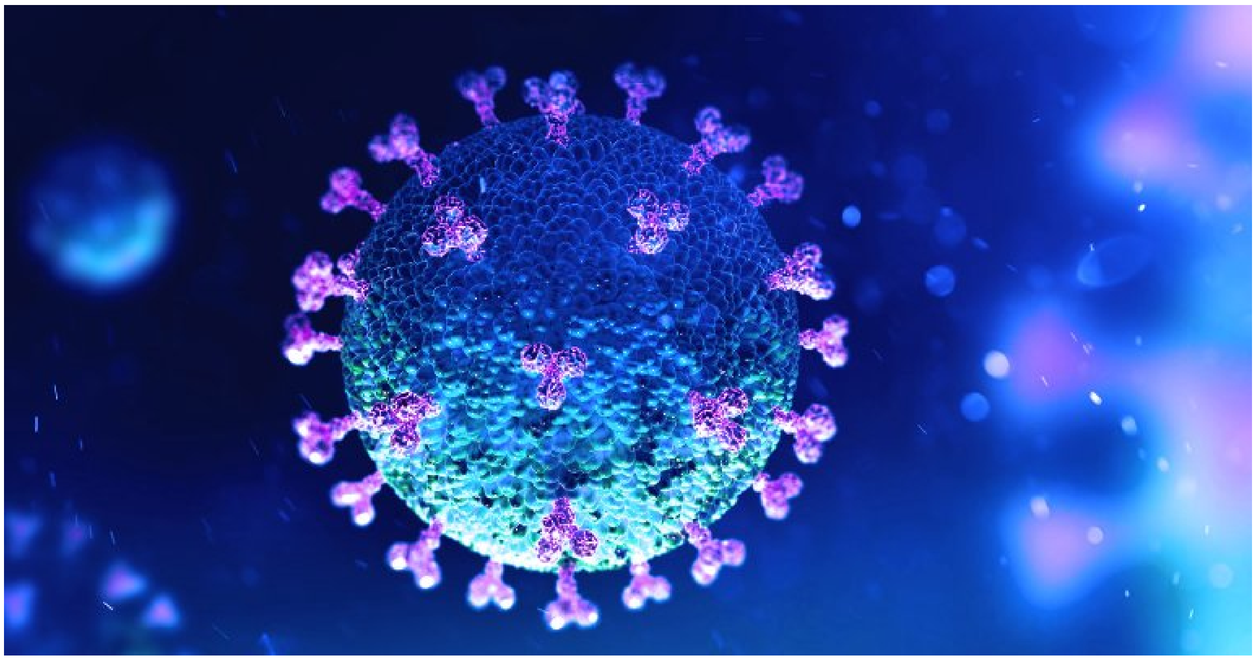


Covid-19 Data Analysis Final Project Report

03.26.2021

—

Eric Alexander Zair
Introduction To Data Science
Syracuse University
Winter 2021

Introduction & Project Overview

The goal of this project is analyze a dataset containing information about covid-19 cases and deaths by US states. The dataset I am using for analysis can be found here: https://data.cdc.gov/api/views/9mfq-cb36/rows.csv. Using this dataset I will be able to take a look at the correlation between covid-19 deaths and covid-19 case for a given state and during a specific time period. The end goal is to see if we can get an idea of what states are doing a "good" job at handling covid-19 as well as cluster the data to figure out what states are similar with respect to covid. We can then determine what states are "good", "average", or "bad" with respect to the pandemic. This will be done using K-means clustering. If I have enough time I will also consider looking into time series forecasting on the covid case case count per state and see if I can factor in a dataset containing information on if a vaccine is out during a certain time period.

Business Questions

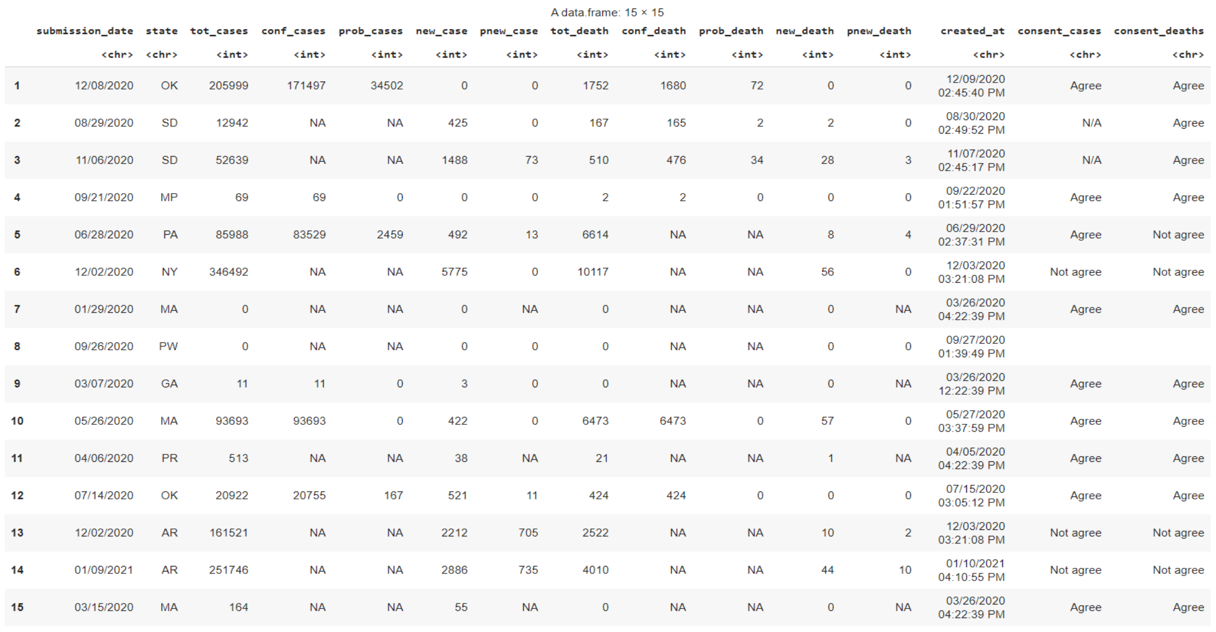
1. What state has the most covid cases?
2. What state has the most covid deaths?
3. What state has the least covid deaths?
4. What state has the least covid deaths?
5. Is there a pattern between deaths and states that we can find without the need for 3rd party data such as age and pre-existing conditions of victims.
6. Construct visualizations (where possible) to display the above metrics.
7. Find out the mean and standard deviation of each state with respect to covid-19.
8. Compare state population to covid cases.
9. Compare state population to covid deaths.
10. Create a metric to represent the efficiency of a state with respect to covid-19.
11. Run K-Means cluster on the dataset to figure out which states are most similar in regards to covid-19 related metrics.

The Dataset

For this project I will specifically be taking a look at the columns in the https://data.cdc.gov/api/views/9mfq-cb36/rows.csv online dataset using the R programming language. In conjunction with the columns/features contained in the dataset, I will also be constructing my own columns and appending them to the original dataset. I will also ensure that the dataset is cleaned e.g. not missing row/column data, the data labels are accurate, time ranges make sense, etc…

Data Cleaning & Transformation

When I first loaded in the covid-19 state dataset it look like the following:



A data.frame: 15 × 15

	submission_date	state	tot_cases	conf_cases	prob_cases	new_case	pnew_case	tot_death	conf_death	prob_death	new_death	pnew_death	created_at	consent_cases	consent_deaths
	<chr>	<chr>	<int>	<int>	<int>	<int>	<int>	<int>	<int>	<int>	<int>	<int>	<chr>	<chr>	<chr>
1	12/08/2020	OK	205999	171497	34502	0	0	1752	1680	72	0	0	12/09/2020 02:45:40 PM	Agree	Agree
2	08/29/2020	SD	12942	NA	NA	425	0	167	165	2	2	0	08/30/2020 02:49:52 PM	N/A	Agree
3	11/06/2020	SD	52639	NA	NA	1488	73	510	476	34	28	3	11/07/2020 02:45:17 PM	N/A	Agree
4	09/21/2020	MP	69	69	0	0	0	2	2	0	0	0	09/22/2020 01:51:57 PM	Agree	Agree
5	06/28/2020	PA	85988	83529	2459	492	13	6614	NA	NA	8	4	06/29/2020 02:37:31 PM	Agree	Not agree
6	12/02/2020	NY	346492	NA	NA	5775	0	10117	NA	NA	56	0	12/03/2020 03:21:08 PM	Not agree	Not agree
7	01/29/2020	MA	0	NA	NA	0	NA	0	NA	NA	0	NA	03/26/2020 04:22:39 PM	Agree	Agree
8	09/26/2020	PW	0	NA	NA	0	0	0	NA	NA	0	0	09/27/2020 01:39:49 PM		
9	03/07/2020	GA	11	11	0	3	0	0	NA	NA	0	NA	03/26/2020 12:22:39 PM	Agree	Agree
10	05/26/2020	MA	93693	93693	0	422	0	6473	6473	0	57	0	05/27/2020 03:37:59 PM	Agree	Agree
11	04/06/2020	PR	513	NA	NA	38	NA	21	NA	NA	1	NA	04/05/2020 04:22:39 PM	Agree	Agree
12	07/14/2020	OK	20922	20755	167	521	11	424	424	0	0	0	07/15/2020 03:05:12 PM	Agree	Agree
13	12/02/2020	AR	161521	NA	NA	2212	705	2522	NA	NA	10	2	12/03/2020 03:21:08 PM	Not agree	Not agree
14	01/09/2021	AR	251746	NA	NA	2886	735	4010	NA	NA	44	10	01/10/2021 04:10:55 PM	Not agree	Not agree
15	03/15/2020	MA	164	NA	NA	55	NA	0	NA	NA	0	NA	03/26/2020 04:22:39 PM	Agree	Agree

Removing NA Records

After querying the data frame to get a count of the amount of NAs I found that the "state" column had 3,798 missing records at the time (keeping in mind that this dataset is still being updated to this day). In total the dataset contained 24,898 rows in it, implying that we would be losing a large portion of data by removing all of the NA fields in the "state" column. Despite that fact however, there was no other option but to remove the records, being that the state field is not numeric. There is no way to figure out which state is associated with each NA row, as the state fields themselves can not be averaged. After removing the 3,798 records from the dataset I was left with 21,100. Furthermore, 15.52% if the original data was removed, leaving us with 85.74% percent of the data to work with. Unfortunately there was no other option here, as the "state" field is extremely important to have for our future analysis.

Changing Column Types

Looking at the above data frame image, we can see that the “submission_date” colum is set a <str> type, however for future queries and plots it would be best to change the column to a <date> type.

Remove columns

The “created_at” column does not provide any important information with respect to the questions/analysis that we are trying to perform on the data, and thus it was removed.

Replace state abbreviations with full names

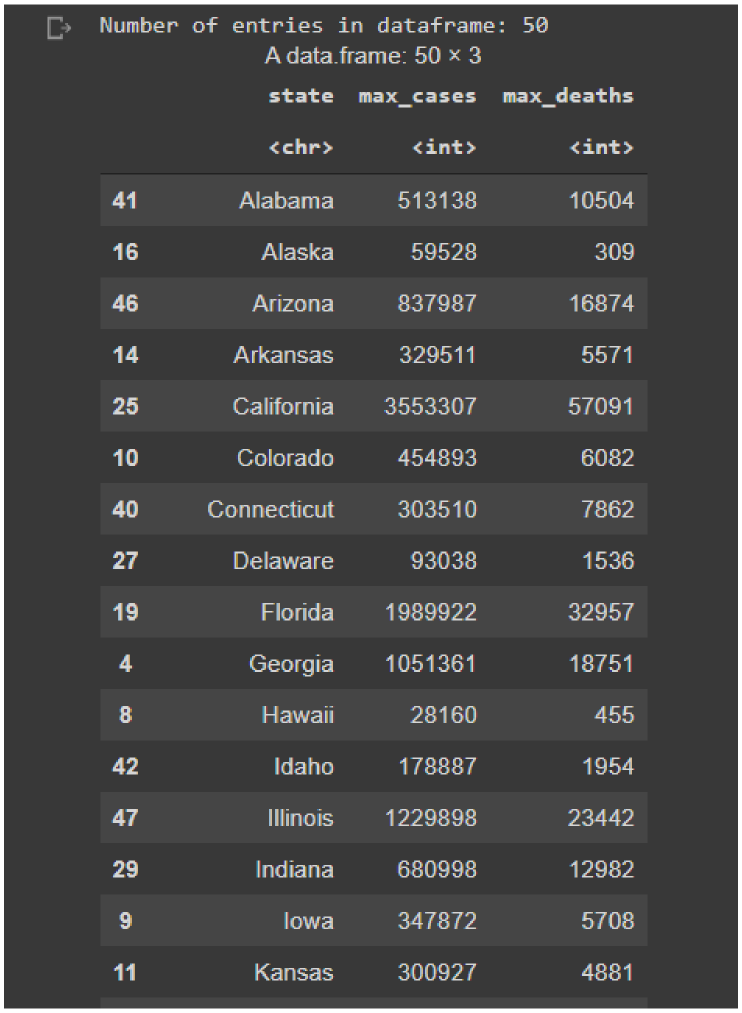
For the sake of future visualizations and ease of read, I decided to replace each record in the “state” column from a state abbreviation, to the full name of the state. For example, “NY” became “New York”. This was very easy to do thanks to the help of the “usdata” library in R. Doing this actually makes interpreting the data a lot easier (as some people do not know all state abbreviations).

Remove Records That Are Not 50 States

In the case of the research that we are doing, we are only interested in the main 50 states in the United States. Moreover, I decided to remove any records for “district of columbia” or “washington dc”, as they would strongly skew our future visualizations and averages (when we add population into our dataframe).

Generating a New Dataframe From the Original

By taking the original dataframe from above (now cleaned and transformed) I was able to query through all 50 unique States and grab the record containing the max number of covid deaths, max number of covid cases, and the name of the name of the state. I then took these records and placed them into a brand new dataframe. The logic here is that using this newly constructed dataframe, we can very easily generate bar charts of each state's deaths and cases. And moving forward, we can generate new columns from the max_cases and max_deaths, which might be useful for our clustering model in the future.
Example view of new dataframe on next page. I then sorted the dataframe by state name so that the visualizations we produce are much easier to find. The states are now in alphabetical order from A - Z for visual aesthetics.



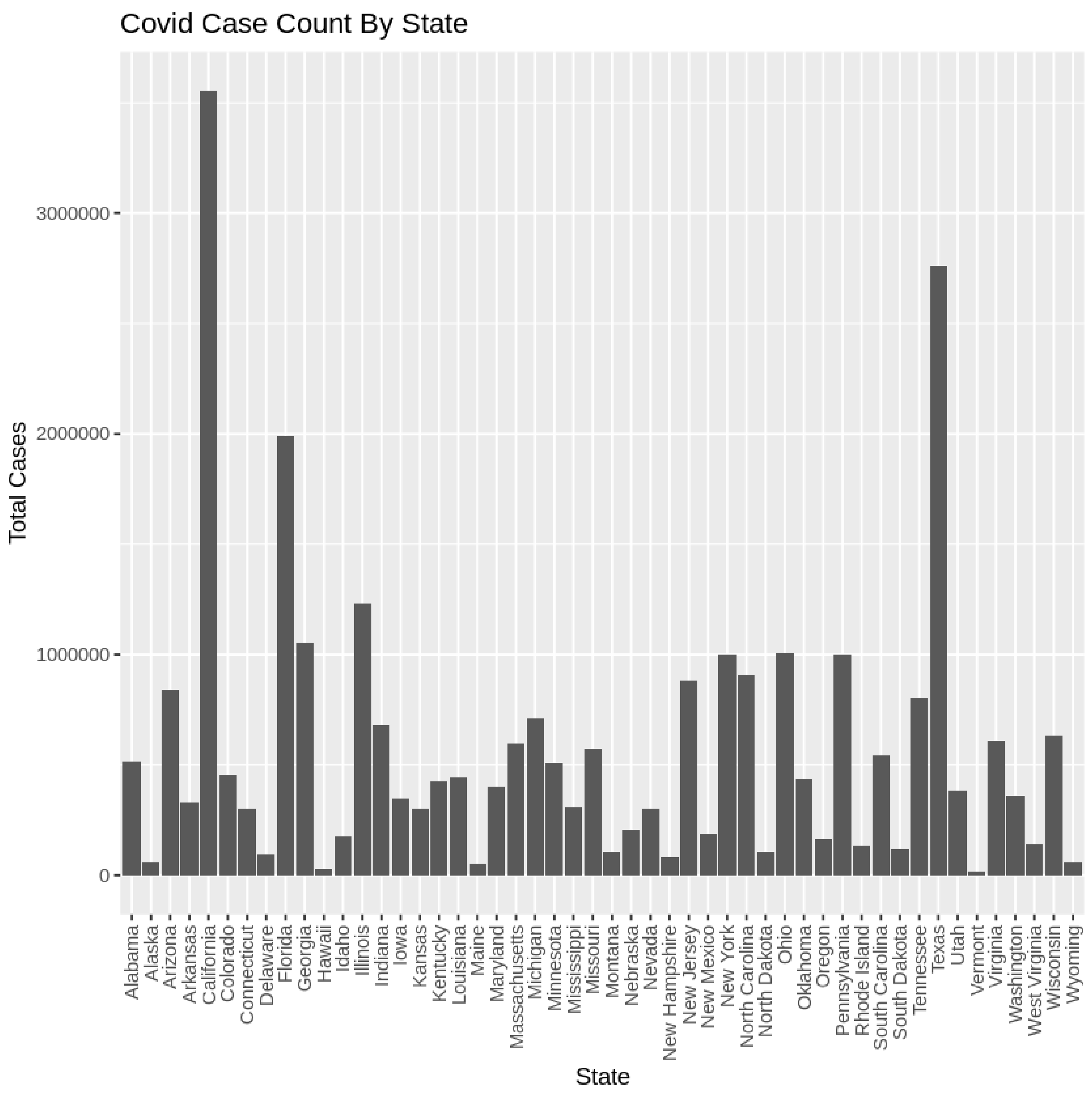
```
Number of entries in dataframe: 50
```

A data.frame: 50 × 3

	state	max_cases	max_deaths
	<chr>	<int>	<int>
41	Alabama	513138	10504
16	Alaska	59528	309
46	Arizona	837987	16874
14	Arkansas	329511	5571
25	California	3553307	57091
10	Colorado	454893	6082
40	Connecticut	303510	7862
27	Delaware	93038	1536
19	Florida	1989922	32957
4	Georgia	1051361	18751
8	Hawaii	28160	455
42	Idaho	178887	1954
47	Illinois	1229898	23442
29	Indiana	680998	12982
9	Iowa	347872	5708
11	Kansas	300927	4881

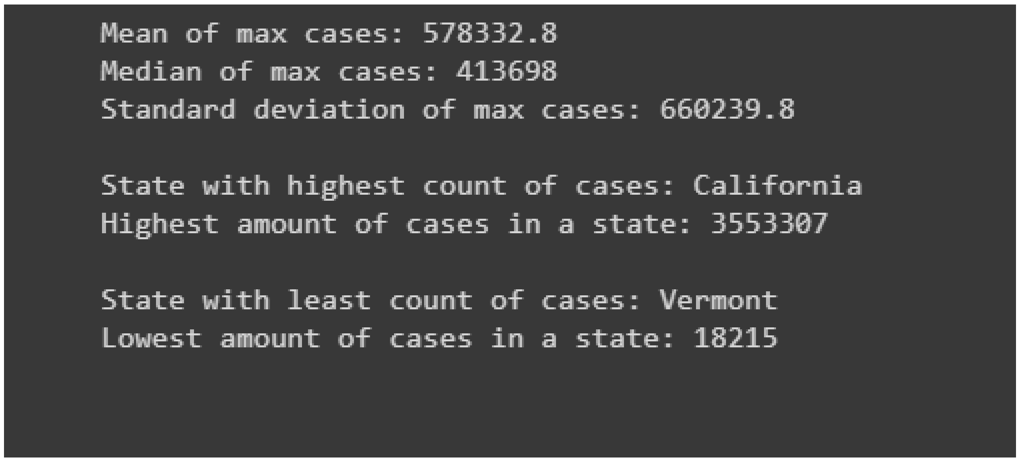
Covid-19 Cases & Deaths by State

Once the data was cleaned up and transformed, I decided to create a view visualizations to help better understand just how many cases and deaths each individual state had in comparison with one another. To do this I generated two different bar graph plots. One representing all states with respect to their covid death count. The other represents all states with respect to their covid death count.



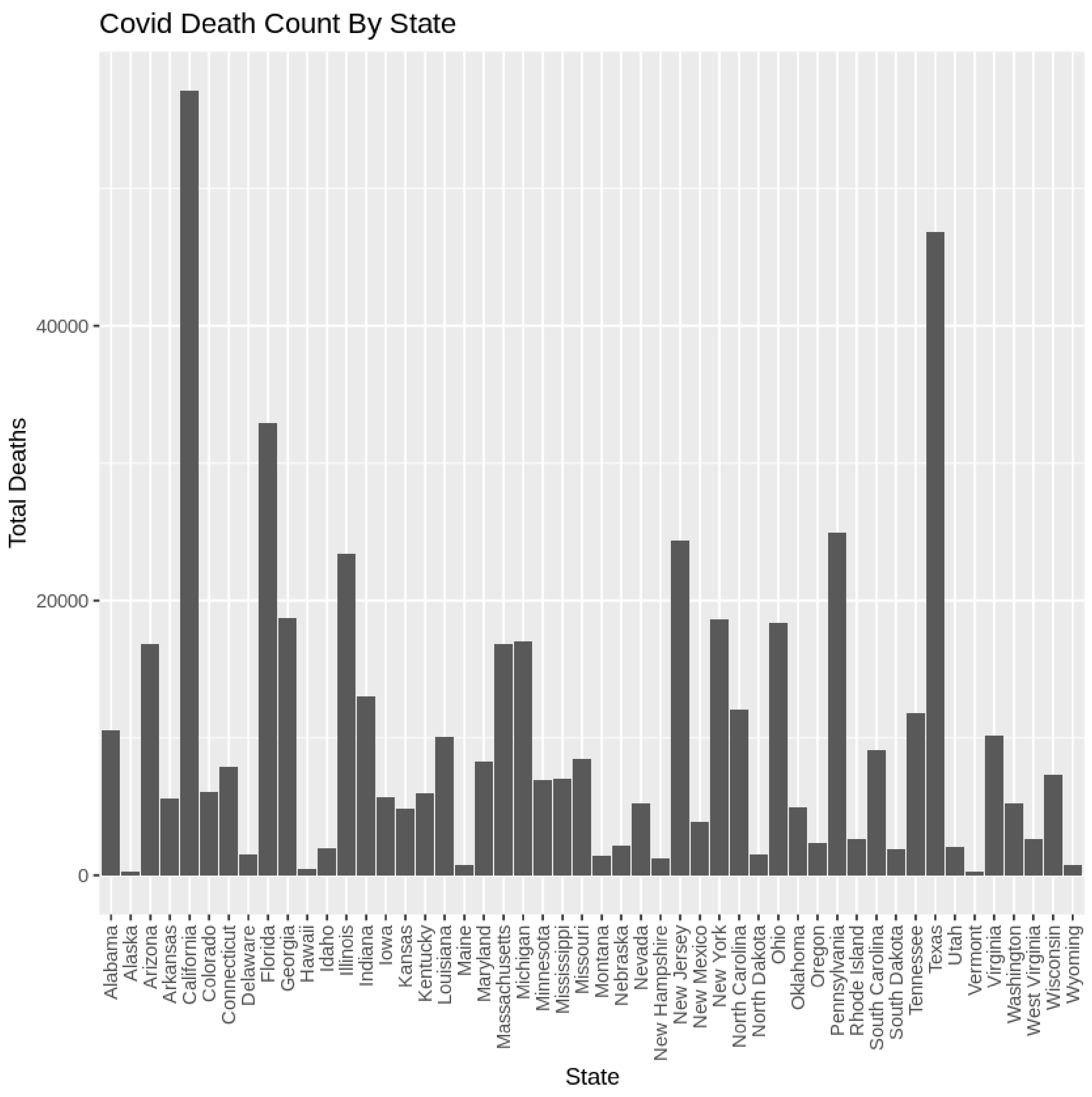
Looking at this data we can immediately point out that California has the highest amount of covid cases by a significant amount. Second in line with respect to covid cases is Texas, and then Florida following behind. Interestingly enough, the news was so highly concerned of NY doing extremely poorly, but looking at other states you can see that NY is actually not significantly worse than the majority of the other states.

General metrics with respect to max covid cases by state:

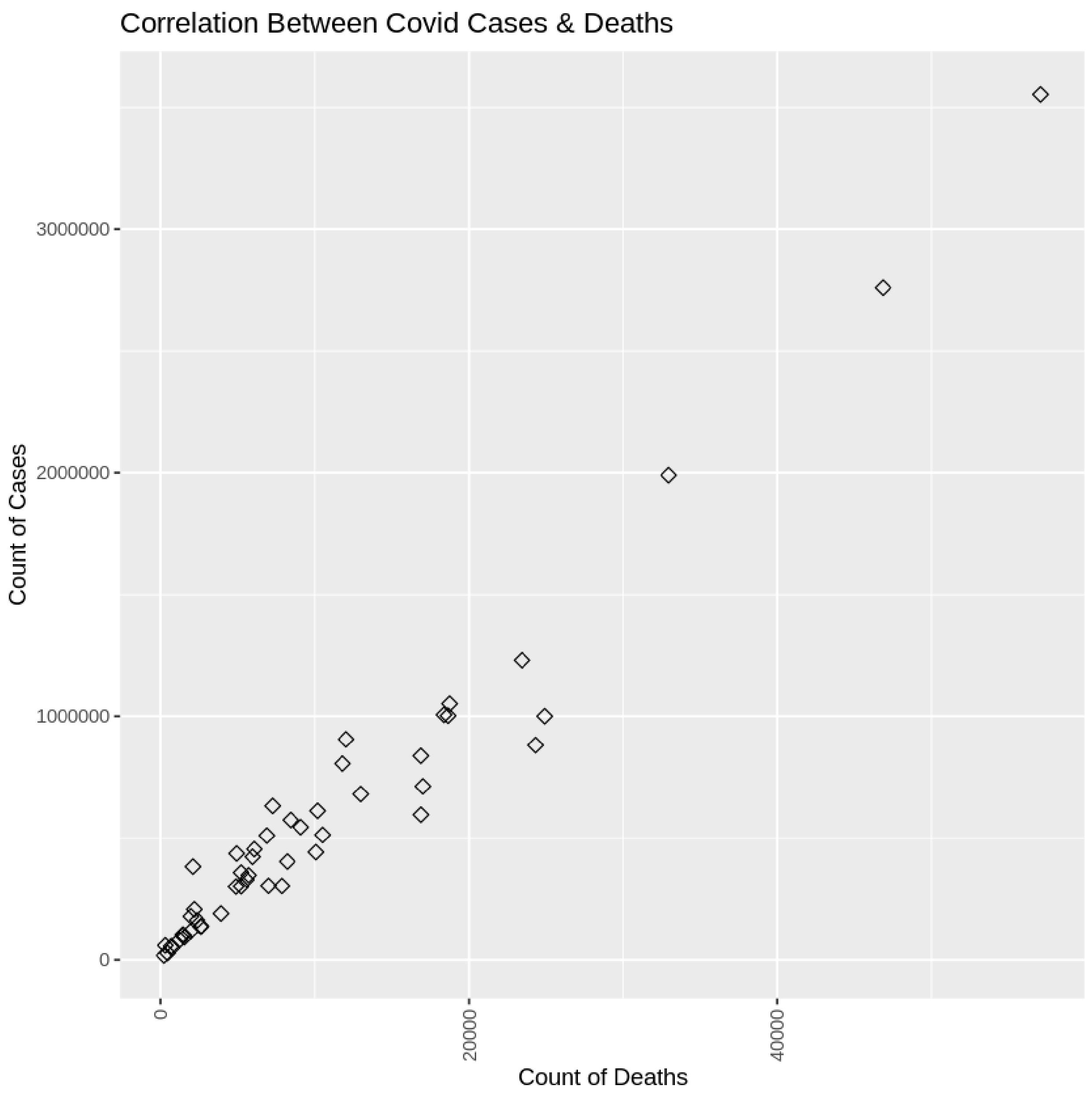


```
Mean of max cases: 578332.8
Median of max cases: 413698
Standard deviation of max cases: 660239.8

State with highest count of cases: California
Highest amount of cases in a state: 3553307

State with least count of cases: Vermont
Lowest amount of cases in a state: 18215
```

Looking at the above numbers we can see that the average number of cases by state is 578,332.8 and the standard deviation of the max cases is a very large 660,239.8, implying that majority of the states have a max_number of cases that is significantly far from their total mean average. And if we were to take the range of that data (660239.8 - 18215) = 642,024.8 then we can clearly see that the spread of cases by state is very different.

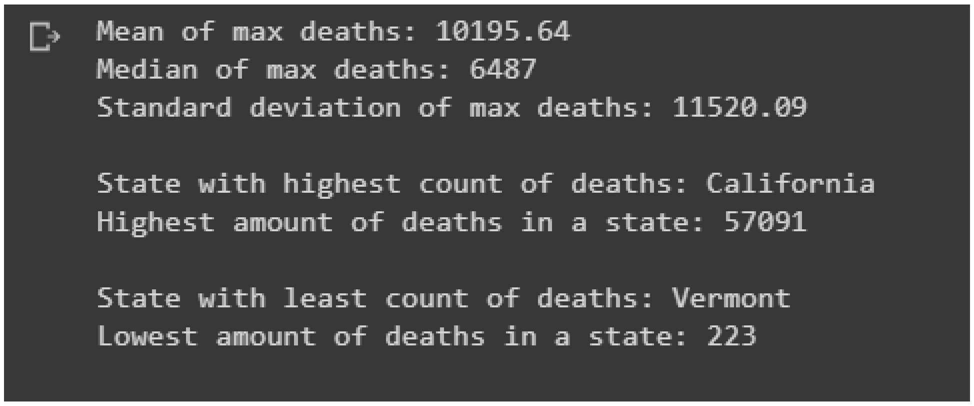


Looking at the above visualization we can actually see that the number of deaths is significantly less than that of the number of cases per each state. However it is very interesting that even though the deaths and cases are very different, if you look at each state you can actually see that the states with the most deaths align with the states that have the most cases. For example, if we look at California, Texas, and Florida, then we can see that this mirrors what we saw in the previous plot. This implies that (to no surprise) there is a correlation between number of cases and number of deaths.



Note that as the number of covid cases increases, so does the number of deaths. This implies that these two variables are highly correlated, and are linearly independent. Though I did not generate a heatmap of the variables, we can pretty much assume strong linear dependence here.

After doing some further calculations of the data the following was discovered:



```
Mean of max deaths: 10195.64
Median of max deaths: 6487
Standard deviation of max deaths: 11520.09

State with highest count of deaths: California
Highest amount of deaths in a state: 57091

State with least count of deaths: Vermont
Lowest amount of deaths in a state: 223
```

We can see that yet again, the standard deviation is rather sparse with respect to the max number of covid cases by each state. The mean is significantly smaller than previous, since our range is much smaller with covid deaths than it was with covid cases e.g. the max number of deaths by state was 10,195, as opposed to the max number of cases by state, which was 578,332.8.

Interestingly enough, these statistics also mirror what we saw previously with the covid cases. Moreover, the state with the least deaths was Vermont. And as it turns out Vermont was also the state with the most covid cases, just showing that strong correlation between cases and deaths again.

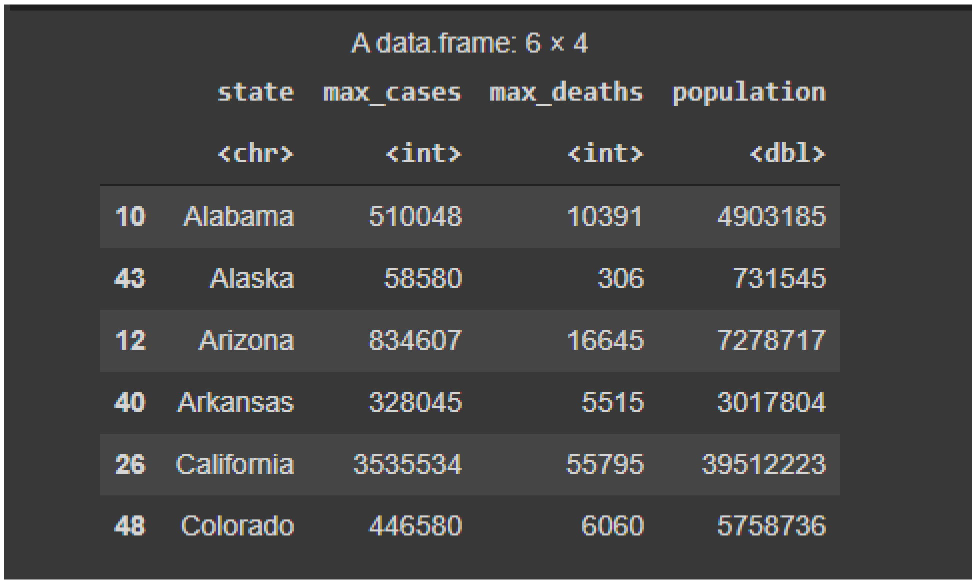
Generating New Columns Using State Population

Even though we have the count of covid-19 cases and deaths by state, that does not necessarily give us an accurate representation of how well states are doing in comparison to one another. Each state has a different population size. It is rather expected that states with large populations have a larger amount of cases. More people would imply more chance for a higher infection rate (in a sense). Therefore, what really determines how well a state is doing in comparison to another state is the percentage of deaths and percentage of cases for each individual state. This will tell us that out of N amount of people percentage P were affected or died in state S. This gives significantly better metrics to make good comparisons and will certainly be a good variable to provide to our clustering model.

Appending Population Column To The Current Dataframe

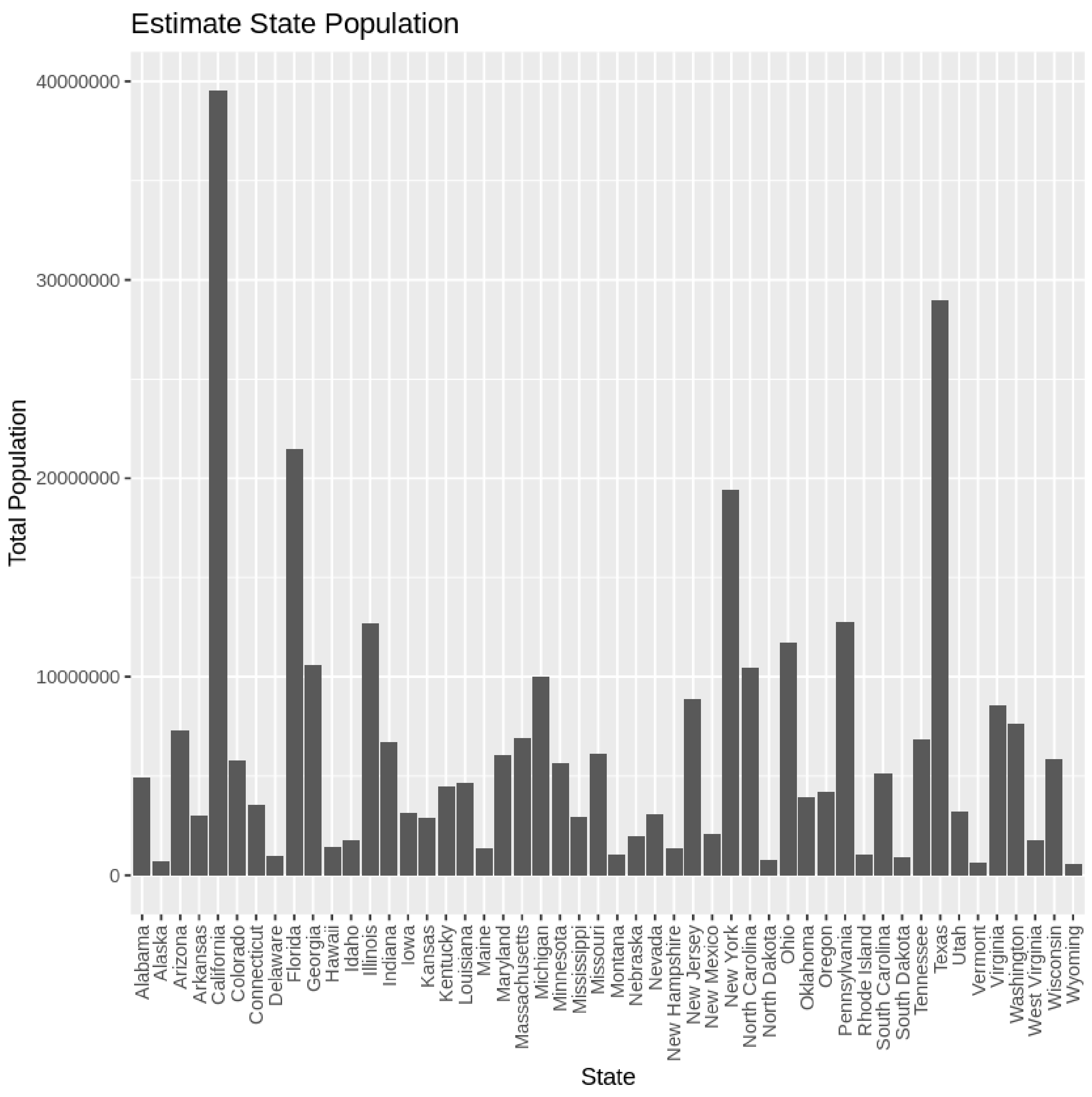
Using the “tidycensus” API and “tidyverse” library, I was able to write some code to output the estimated population for each individual state into a separate data frame. Unfortunately the most recent year that the data API was able to pull population data from was 2019, however, this is the same data that most respectable data scientists use with respect to 2021 U.S. population matters, so this data should be efficient, as population does not grow exponentially, and generally scales at the same rate per state (from my understanding). I was able to sort both data frames by “State” and then merge the population column into the data frame I was currently using.

Furthermore, the data frame being used now looks like this:



A data.frame: 6 × 4

	state	max_cases	max_deaths	population
	<chr>	<int>	<int>	<dbl>
10	Alabama	510048	10391	4903185
43	Alaska	58580	306	731545
12	Arizona	834607	16645	7278717
40	Arkansas	328045	5515	3017804
26	California	3535534	55795	39512223
48	Colorado	446580	6060	5758736

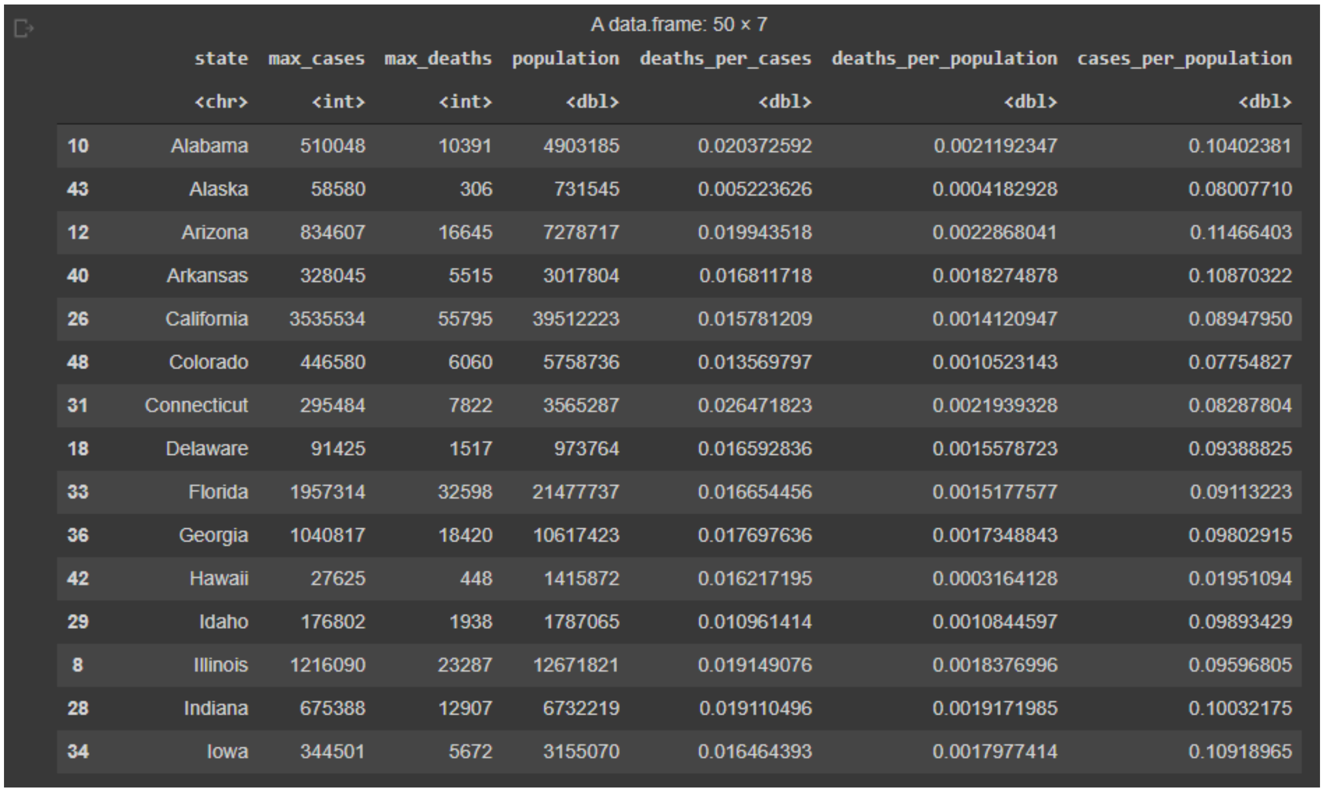


Notice again that California, Texas, and Florida are very large on this graph. This mirrors the previous plots as well, essentially showing us that the population variable is highly correlated with cases and deaths. Looking at a heatmap of this would better show us this.

Generating Death/Population & Death/Cases Columns

The next step done was to take the Population and generate a percentage of deaths and cases by state and then append those variables as columns in our dataframe. Then using these new metrics we can generate some more visualizations on which states have the most deaths and cases by population. We also create an additional column containing the percentage of deaths by cases. This will tell us out of the number of people that had covid-19, what percentage died in a given state. We can feed this data into a clustering model and see the similarities of state efficiency during the covid-19 pandemic.

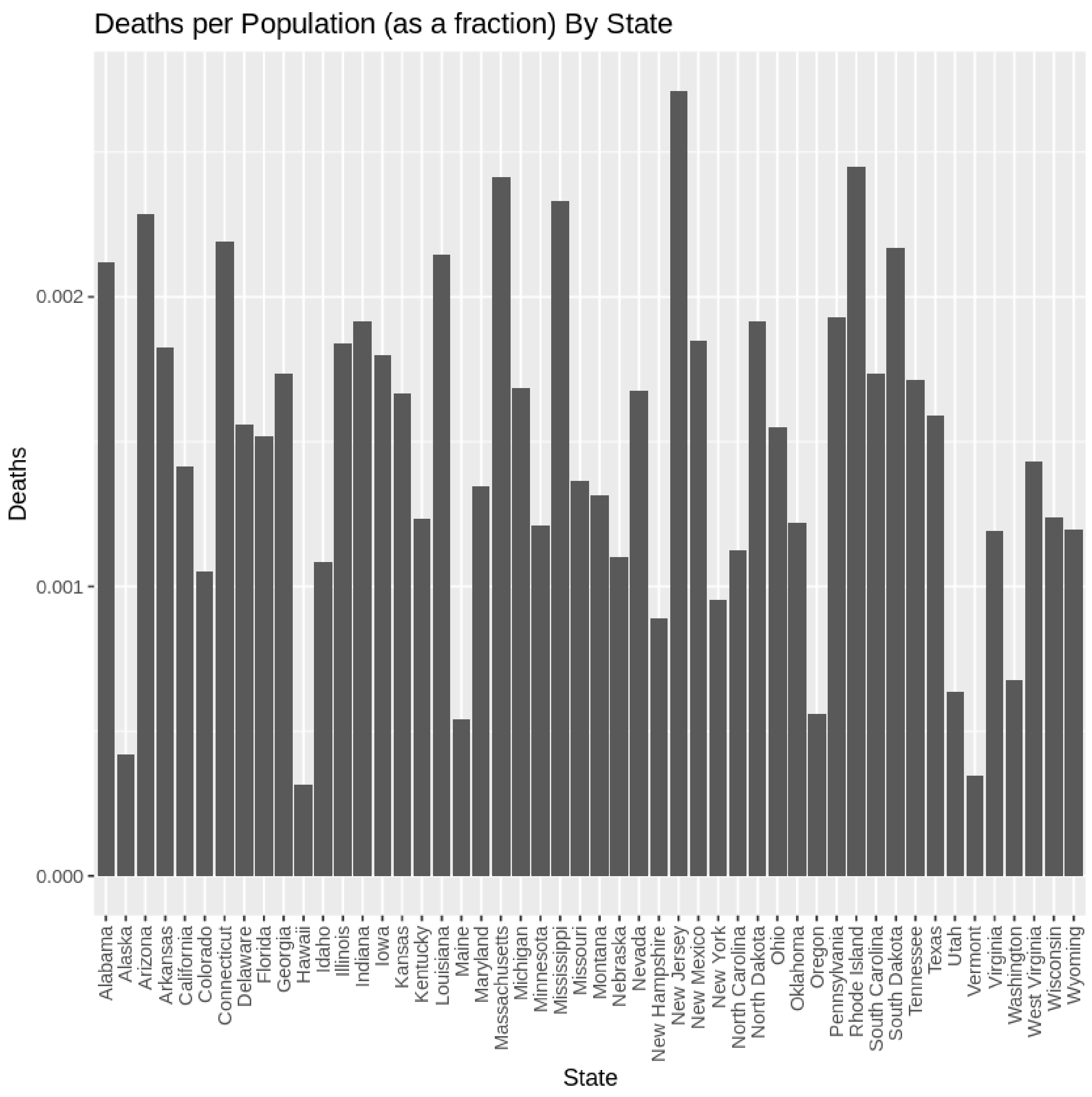
View of new dataset after generating new metrics:

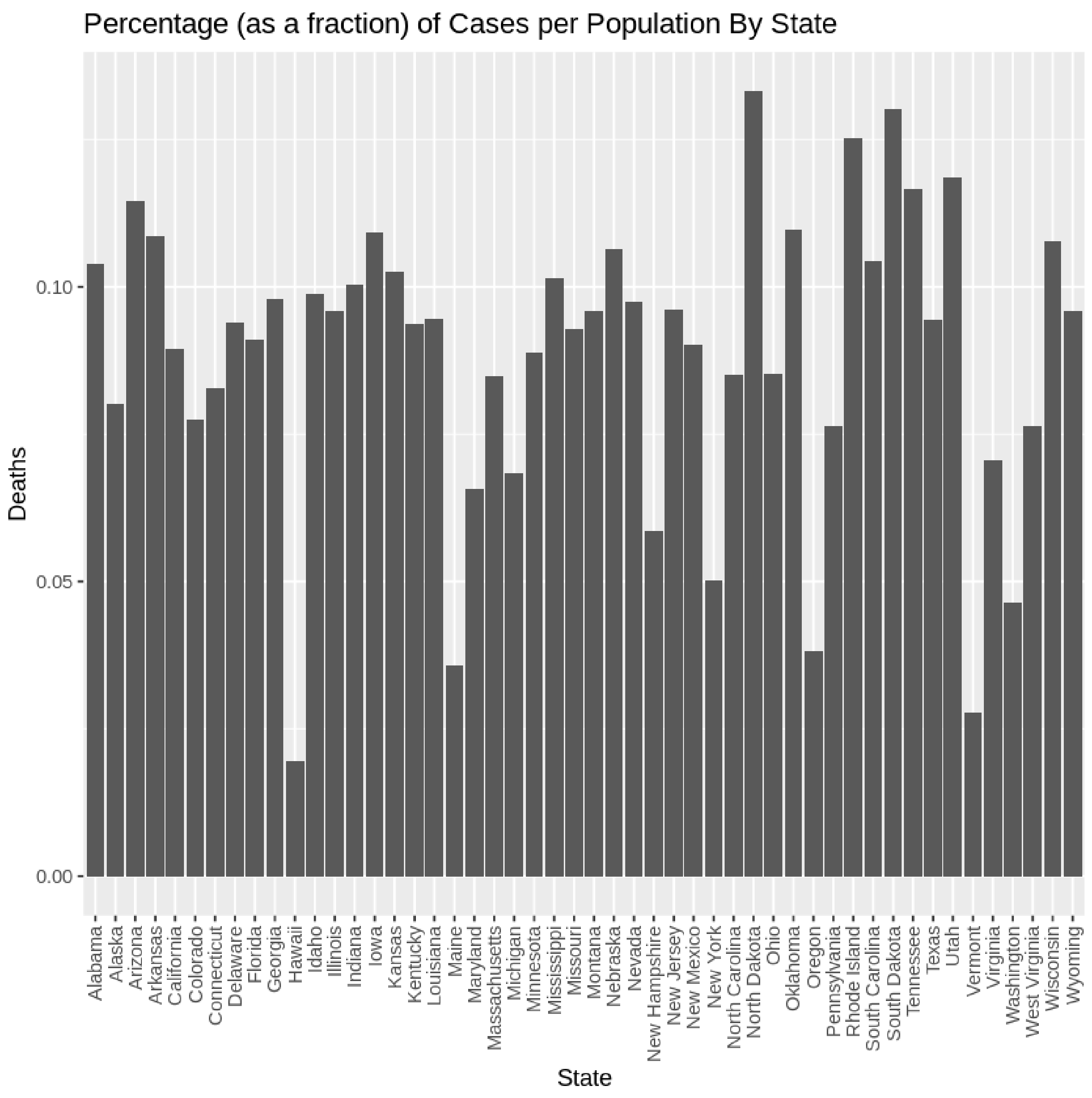


A data.frame: 50 × 7

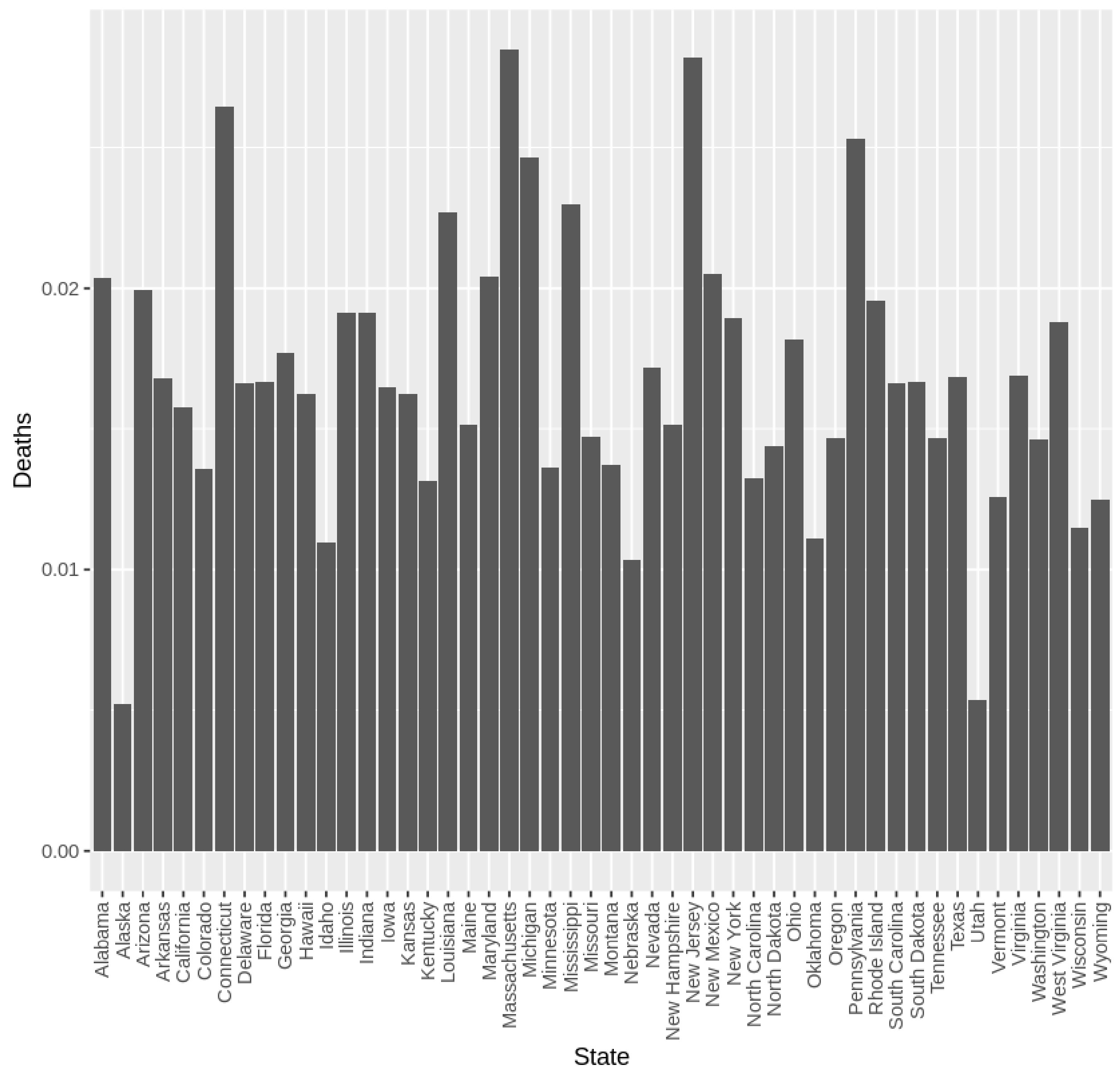
	state	max_cases	max_deaths	population	deaths_per_cases	deaths_per_population	cases_per_population
	<chr>	<int>	<int>	<dbl>	<dbl>	<dbl>	<dbl>
10	Alabama	510048	10391	4903185	0.020372592	0.0021192347	0.10402381
43	Alaska	58580	306	731545	0.005223626	0.0004182928	0.08007710
12	Arizona	834607	16645	7278717	0.019943518	0.0022868041	0.11466403
40	Arkansas	328045	5515	3017804	0.016811718	0.0018274878	0.10870322
26	California	3535534	55795	39512223	0.015781209	0.0014120947	0.08947950
48	Colorado	446580	6060	5758736	0.013569797	0.0010523143	0.07754827
31	Connecticut	295484	7822	3565287	0.026471823	0.0021939328	0.08287804
18	Delaware	91425	1517	973764	0.016592836	0.0015578723	0.09388825
33	Florida	1957314	32598	21477737	0.016654456	0.0015177577	0.09113223
36	Georgia	1040817	18420	10617423	0.017697636	0.0017348843	0.09802915
42	Hawaii	27625	448	1415872	0.016217195	0.0003164128	0.01951094
29	Idaho	176802	1938	1787065	0.010961414	0.0010844597	0.09893429
8	Illinois	1216090	23287	12671821	0.019149076	0.0018376996	0.09596805
28	Indiana	675388	12907	6732219	0.019110496	0.0019171985	0.10032175
34	Iowa	344501	5672	3155070	0.016464393	0.0017977414	0.10918965

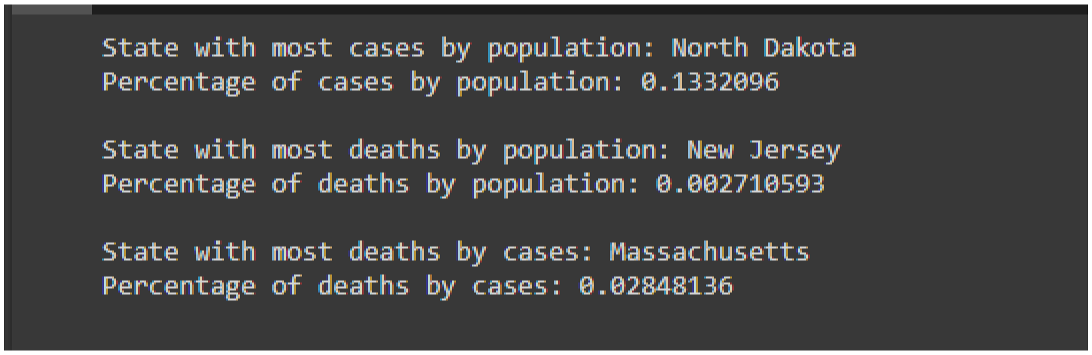
Visualizations of Each State with Respect to New Columns Added


Deaths per Population (as a fraction) By State
Deaths
0.000
0.001
0.002
State
Alabama
Alaska
Arizona
Arkansas
California
Colorado
Connecticut
Delaware
Florida
Georgia
Hawaii
Idaho
Illinois
Indiana
Iowa
Kansas
Kentucky
Louisiana
Maine
Maryland
Massachusetts
Michigan
Minnesota
Mississippi
Missouri
Montana
Nebraska
Nevada
New Hampshire
New Jersey
New Mexico
New York
North Carolina
North Dakota
Ohio
Oklahoma
Oregon
Pennsylvania
Rhode Island
South Carolina
South Dakota
Tennessee
Texas
Utah
Vermont
Virginia
Washington
West Virginia
Wisconsin
Wyoming

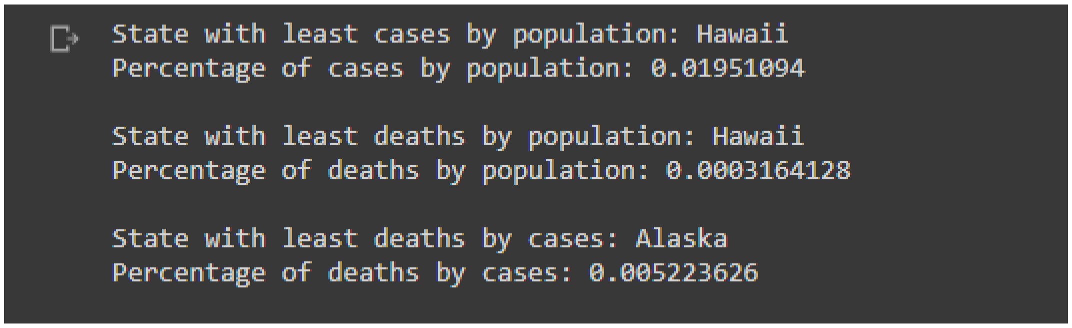

Percentage (as a fraction) of Cases per Population By State
Deaths
0.00
0.05
0.10
State
Alabama
Alaska
Arizona
Arkansas
California
Colorado
Connecticut
Delaware
Florida
Georgia
Hawaii
Idaho
Illinois
Indiana
Iowa
Kansas
Kentucky
Louisiana
Maine
Maryland
Massachusetts
Michigan
Minnesota
Mississippi
Missouri
Montana
Nebraska
Nevada
New Hampshire
New Jersey
New Mexico
New York
North Carolina
North Dakota
Ohio
Oklahoma
Oregon
Pennsylvania
Rhode Island
South Carolina
South Dakota
Tennessee
Texas
Utah
Vermont
Virginia
Washington
West Virginia
Wisconsin
Wyoming

Deaths Per Cases (as a fraction) By State





```
State with most cases by population: North Dakota
Percentage of cases by population: 0.1332096

State with most deaths by population: New Jersey
Percentage of deaths by population: 0.002710593

State with most deaths by cases: Massachusetts
Percentage of deaths by cases: 0.02848136
```



```
State with least cases by population: Hawaii
Percentage of cases by population: 0.01951094

State with least deaths by population: Hawaii
Percentage of deaths by population: 0.0003164128

State with least deaths by cases: Alaska
Percentage of deaths by cases: 0.005223626
```

Very interesting! When plotting the visualizations of our new columns by state as well as doing some quick statistics, we are finally getting different results. In other words, even though California has been shown as the state performing the worst in all of the previous charts, we actually see that North Dakota is the state that has the most cases by population, New Jersey is the state performing the worst with respect to deaths by population, and Massachusetts has the most deaths by covid cases.

These variables are significantly more meaningful to view as opposed to looking at the number of deaths and number of cases without any account for population, as it is extremely biased on the data. This approach however makes much more sense, as it takes into account the size of the state and outputs a percentage with respect to that individual state's population. Thus not being affected by the bias of the size of the state.

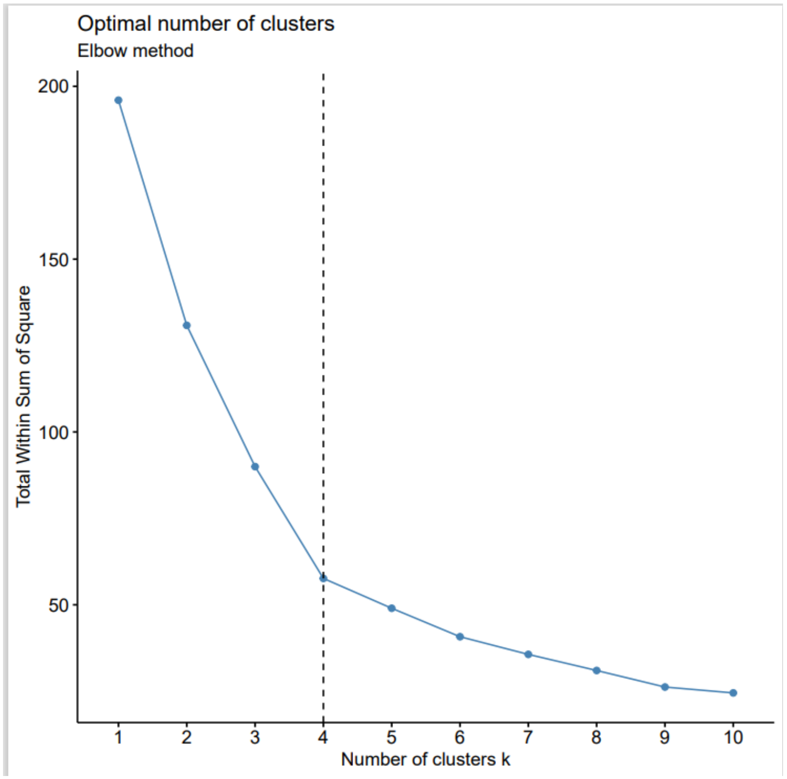
Modeling Techniques -- KMeans Clustering

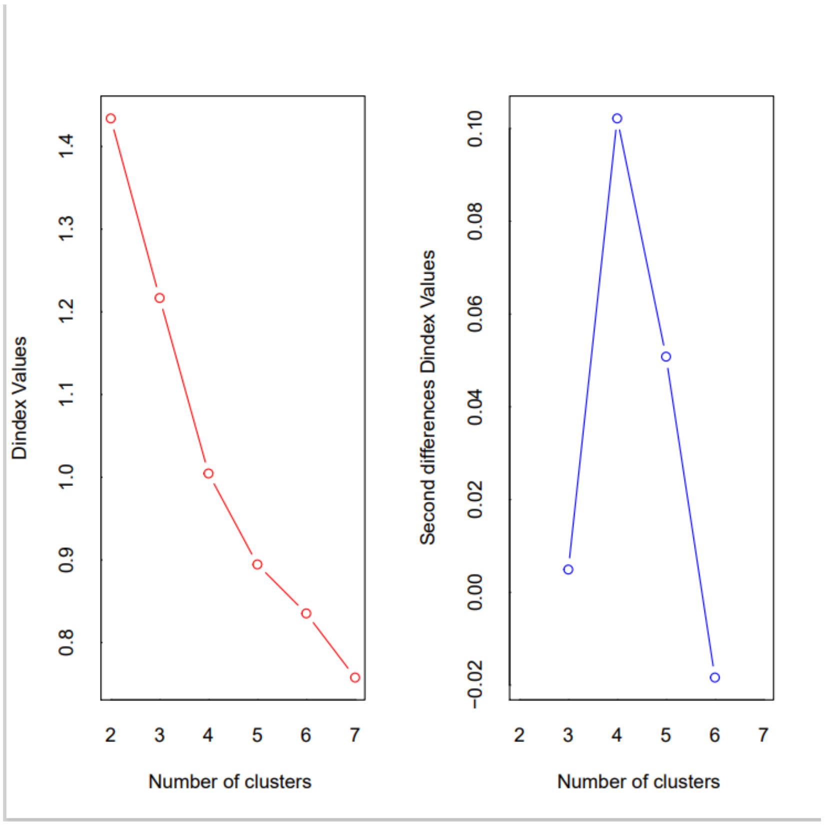
In order to determine how well states performed with respect to covid-19 I decided to construct a Kmeans clustering model. We use this model to attempt to group states into the 3 different groups that we can identify. In other words we group states into bad, average, and good with respect to covid-19 performance.

Determining The Number of K Clusters for Kmeans

Generally speaking when K Means clustering is used, we often try to find the optimal K number. K implies the number of clusters that we should group our data into. There are multiple ways to do this such as the “Elbow method” or the “Silhouette method”. We visually plot the results K produces overtime and then take the cluster that has the least change between steps (essentially).

In the case of the problem we are working on now I was contemplating if I should just decide to use 3 clusters (to represent bad, average, and good) or if I should use N clusters, where N is the output of an algorithm for finding the optimal number of clusters. At the end of the day I decided that since I specifically want to group bad, average, and good performance, that I will hardset my K to be 3 clusters. However I was rather curious what the optimal number of clusters would be so I decided to look into it. Below are multiple plots to help show the optimal number of K clusters.


Optimal number of clusters
Elbow method
Total Within Sum of Square
200
150
100
50
1
2
3
4
5
6
7
8
9
10
Number of clusters k


Dindex Values
1.4
1.3
1.2
1.1
1.0
0.9
0.8
2
3
4
5
6
7
Number of clusters
Second differences Dindex Values
0.10
0.08
0.06
0.04
0.02
0.00
−0.02
2
3
4
5
6
7
Number of clusters

Interestingly enough the output of most all of the algorithms I used to find the optimal number of clusters revealed that 4 would be the optimal amount of clusters to use.

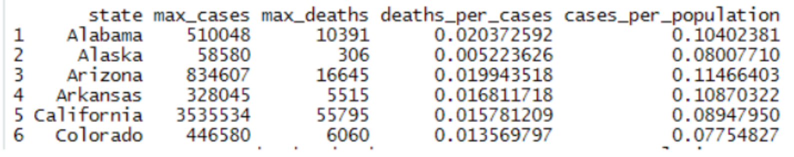
As stated earlier however, we will be using 3 clusters for the sake of simplicity and for my specific curiosity of identifying bad, average, and good.

Dataframe Used For Clustering

Before I began running any clustering on our data it was very important that I cleaned it up and ensured that it would not be very biased for the clustering. I decided to remove the population field from the dataframe completely, as I figured that the K Means model would strongly weight that variable more than the other ones, since it's values are significantly larger than any other field and the population of the large states are much larger than that of the small states. (Later testing proved that this hypothesis was correct).

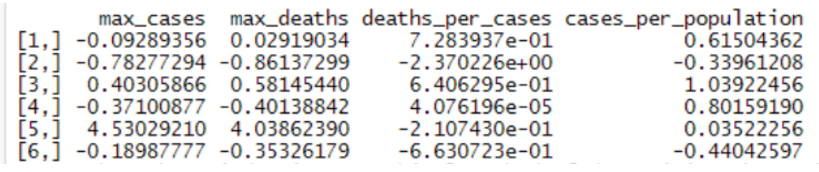
I also decided to take all of the numeric columns in the dataframe and standardized/scaled them prior to passing them to our model, as I knew that in order to visualize the result of the clustering I would need to use PCA (Principal Component Analysis) an PCA can be strongly affected by the scaling of features. Also note that in general, it is never a bad idea to scale our data down for any machine learning model, as it helps to allow the model to converge quicker regardless (especially when the model uses gradient descent as its optimization algorithm).

Before Standardizing the data:



```
        state max_cases max_deaths deaths_per_cases cases_per_population
1     Alabama    510048      10391      0.020372592           0.10402381
2      Alaska     58580        306      0.005223626           0.08007710
3     Arizona    834607      16645      0.019943518           0.11466403
4    Arkansas    328045       5515      0.016811718           0.10870322
5  California   3535534      55795      0.015781209           0.08947950
6    Colorado    446580       6060      0.013569797           0.07754827
```

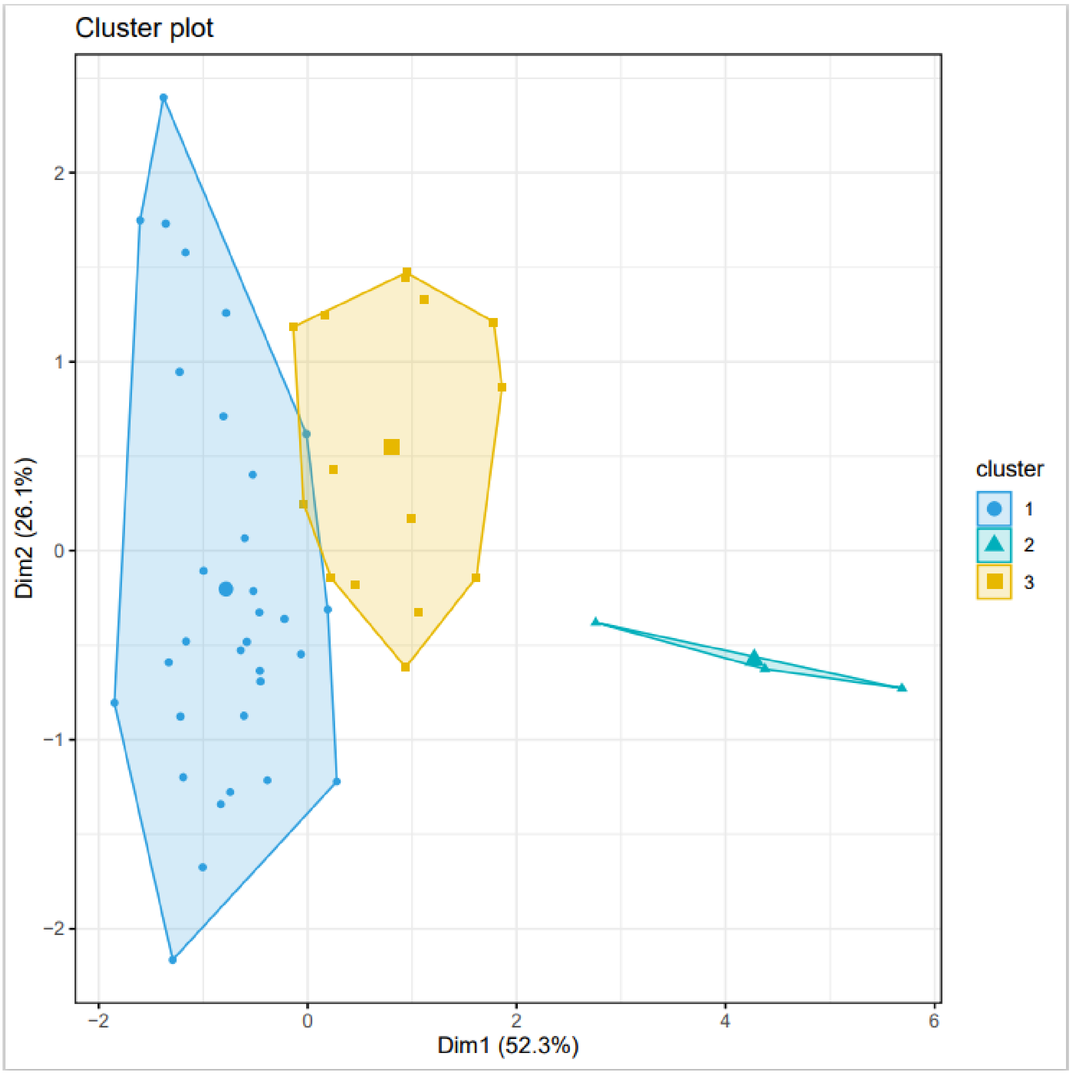
After Standardizing the data:

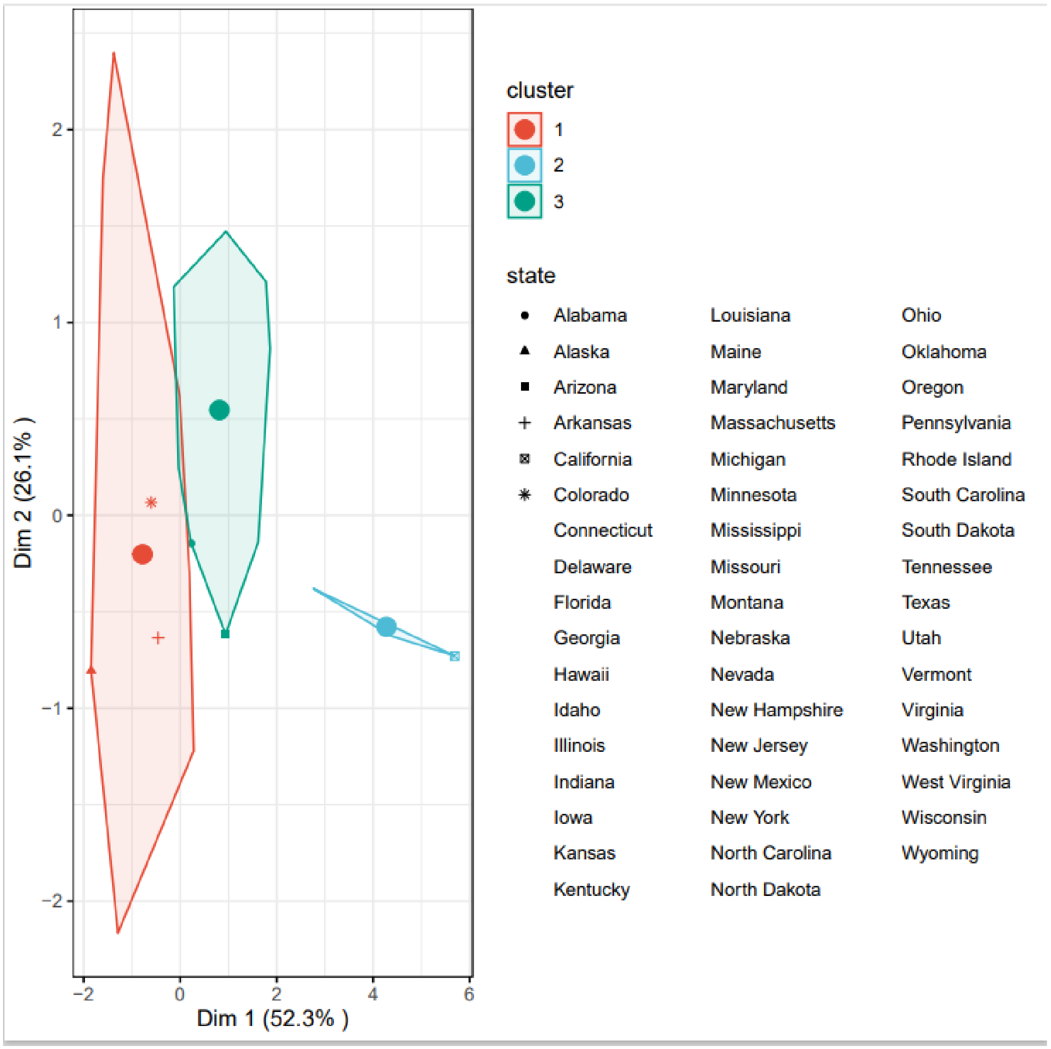


```
        max_cases  max_deaths deaths_per_cases cases_per_population
[1,] -0.09289356  0.02919034     7.283937e-01           0.61504362
[2,] -0.78277294 -0.86137299    -2.370226e+00          -0.33961208
[3,]  0.40305866  0.58145440     6.406295e-01           1.03922456
[4,] -0.37100877 -0.40138842     4.076196e-05           0.80159190
[5,]  4.53029210  4.03862390    -2.107430e-01           0.03522256
[6,] -0.18987777 -0.35326179    -6.630723e-01          -0.44042597
```

K Means Results & Visualizations

Now that the dataset is cleaned up we run it through the K Means model with K=3 clusters and then use PCA to visualize the clusters on a 2-Dimensional plane.

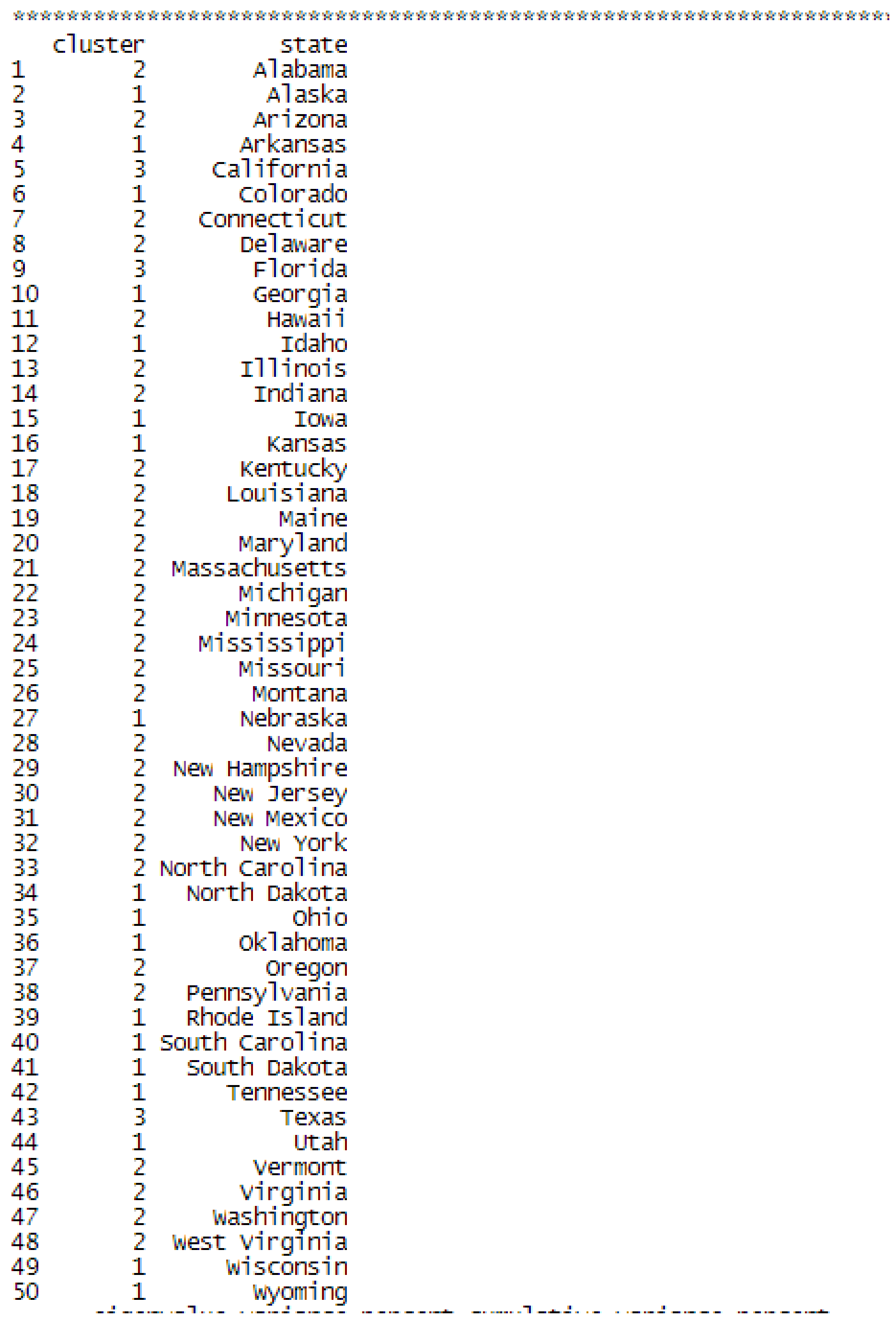




Taking a look at the above clustering images we can immediately see that cluster number 1 (the red cluster) contains the majority of the states in it, cluster 2 contains the second highest amount of states, and cluster 3 contains the least amount of states in it.

After doing a lot of research into these outputted visualizations, as well as cross comparing our previous barplot visualizations against the list of states assigned to their outputted K Means cluster, it can be revealed that cluster 1 represents “good” performance, cluster 2 represents “average” performance, and cluster 3 represents “bad” performance, all with respect to covid-19 deaths and cases per each individual state.

List of all states and their assigned K Means cluster:



```
*****************************************************************
   cluster          state
1        2        Alabama
2        1         Alaska
3        2        Arizona
4        1       Arkansas
5        3     California
6        1       Colorado
7        2    Connecticut
8        2       Delaware
9        3        Florida
10       1        Georgia
11       2         Hawaii
12       1          Idaho
13       2       Illinois
14       2        Indiana
15       1           Iowa
16       1         Kansas
17       2       Kentucky
18       2      Louisiana
19       2          Maine
20       2       Maryland
21       2  Massachusetts
22       2       Michigan
23       2      Minnesota
24       2    Mississippi
25       2       Missouri
26       2        Montana
27       1       Nebraska
28       2         Nevada
29       2  New Hampshire
30       2     New Jersey
31       2     New Mexico
32       2       New York
33       2 North Carolina
34       1   North Dakota
35       1           Ohio
36       1       Oklahoma
37       2         Oregon
38       2   Pennsylvania
39       1   Rhode Island
40       1 South Carolina
41       1   South Dakota
42       1      Tennessee
43       3          Texas
44       1           Utah
45       2        Vermont
46       2       Virginia
47       2     Washington
48       2  West Virginia
49       1      Wisconsin
50       1        Wyoming
```

Modeling Techniques -- Classification using Naive Baye's & SVM

Once the cluster was completed, I figured that it would be rather interesting to see if we could produce an accurate classification model that could predict (with good accuracy) the results of the K Means clustering labeling. In other words, can we create a model that can predict the clustering number that a state was assigned to using other variables?

If the model is good at predicting the correct clustering label, that would imply that the clustering results are really accurate. Keep in mind however that the dataset we constructed does not contain that many rows, but does contain a decent amount of columns. Furthermore it is expected that the model will most likely be inaccurate.

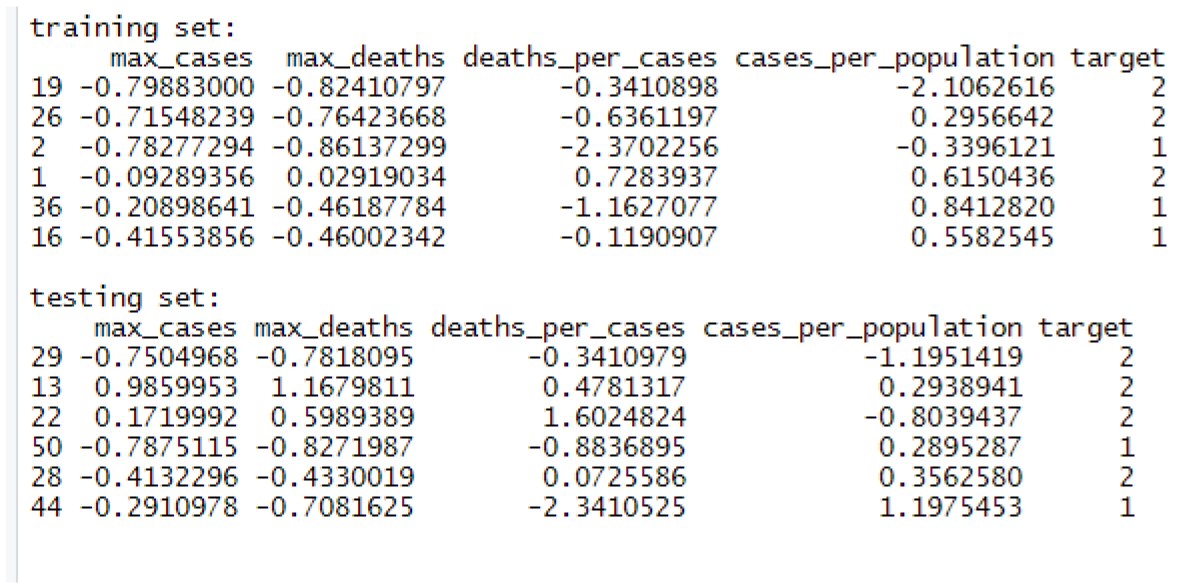
Classification using SVM

Since we do not have a large amount of data, I am going to stick to using the SVM model, as it tends to perform very well on small datasets and also does a great job at handling non-linear classification predictions on multiple different classes.

In the case of this problem we have three different classes we are trying to predict [1, 2, 3] respectively. These three classes represent the three possible clusters that a state can be assigned to.

For classification in general we will use the same exact dataset as with the clustering, however we will ensure to remove the state column from it for many reasons. 1) The state column would have to be One Hot Encoded or numerically encoded and 2) The state column would cause our model to overfit extremely since there is only one record of each different state in the current dataframe we are using.

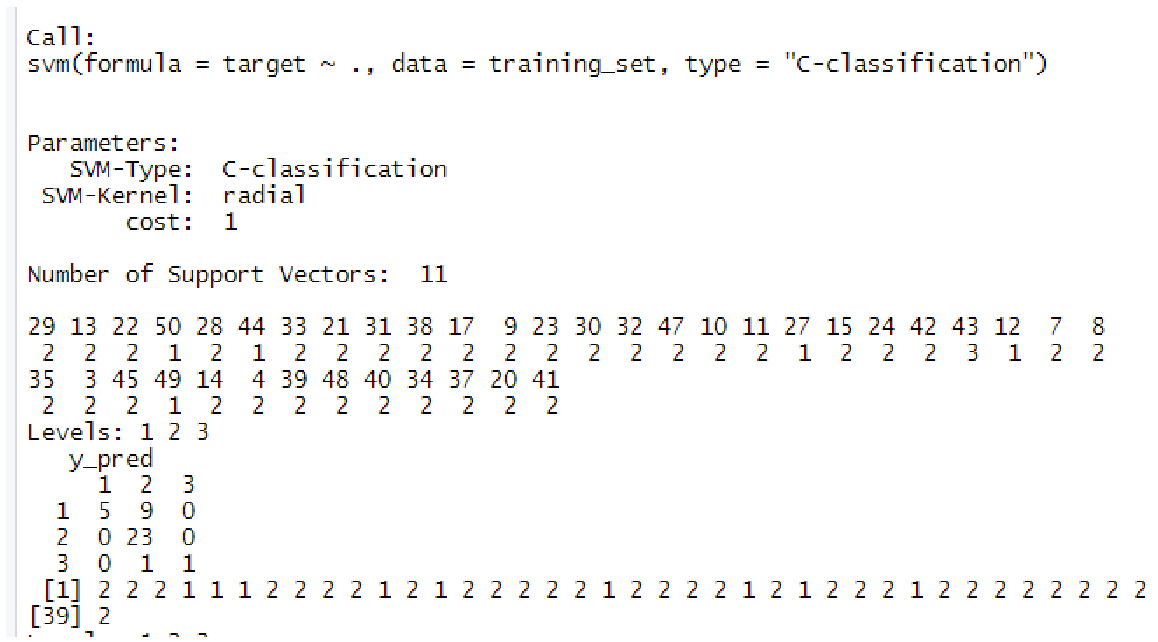
View of the testing and training dataframes:



```
training set:
     max_cases  max_deaths deaths_per_cases cases_per_population target
19 -0.79883000 -0.82410797       -0.3410898           -2.1062616      2
26 -0.71548239 -0.76423668       -0.6361197            0.2956642      2
2  -0.78277294 -0.86137299       -2.3702256           -0.3396121      1
1  -0.09289356  0.02919034        0.7283937            0.6150436      2
36 -0.20898641 -0.46187784       -1.1627077            0.8412820      1
16 -0.41553856 -0.46002342       -0.1190907            0.5582545      1

testing set:
    max_cases max_deaths deaths_per_cases cases_per_population target
29 -0.7504968 -0.7818095       -0.3410979           -1.1951419      2
13  0.9859953  1.1679811        0.4781317            0.2938941      2
22  0.1719992  0.5989389        1.6024824           -0.8039437      2
50 -0.7875115 -0.8271987       -0.8836895            0.2895287      1
28 -0.4132296 -0.4330019        0.0725586            0.3562580      2
44 -0.2910978 -0.7081625       -2.3410525            1.1975453      1
```

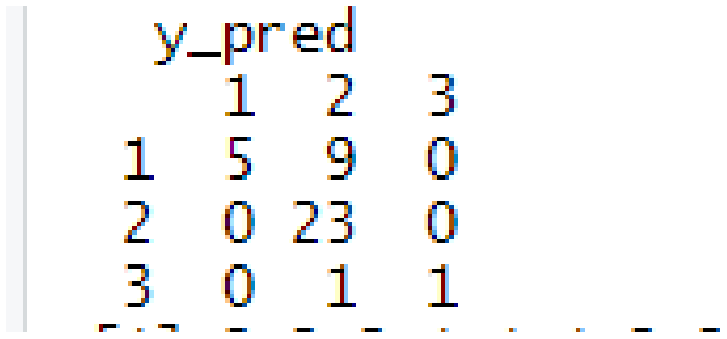
For this classification problem, we will use 20% of the data for testing and the other 80 percent for training. Note that the data is already standardized above.

With respect to the results of this model, I am not interested in how well the training did compared to how well the testing did, simply because there was just not a lot of data. Instead what I wanted to look at was specifically the confusion matrix of the data to see how many things overall it classified correctly. Again I must note that this is a very odd classification problem because there are only 50 records that we are running this model on and thus 20% percent of the states were thrown into the testing set, but we do in fact care about if our classification model correctly predicts these states. Furthermore looking at the confusion matrix itself will tell us exactly how well our model did with respect to this specific problem.



```
Call:
svm(formula = target ~ ., data = training_set, type = "C-classification")


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  1

Number of Support Vectors:  11

29 13 22 50 28 44 33 21 31 38 17  9 23 30 32 47 10 11 27 15 24 42 43 12  7  8
 2  2  2  1  2  1  2  2  2  2  2  2  2  2  2  2  2  2  1  2  2  2  3  1  2  2
35  3 45 49 14  4 39 48 40 34 37 20 41
 2  2  2  1  2  2  2  2  2  2  2  2  2
Levels: 1 2 3
   y_pred
     1  2  3
  1  5  9  0
  2  0 23  0
  3  0  1  1
 [1] 2 2 2 1 1 1 2 2 2 2 1 2 1 2 2 2 2 2 1 2 2 2 2 1 2 1 2 2 2 1 2 2 2 2 2 2 2 2
[39] 2
```

View of confusion matrix for SVM model:



```
   y_pred
     1  2  3
  1  5  9  0
  2  0 23  0
  3  0  1  1
```

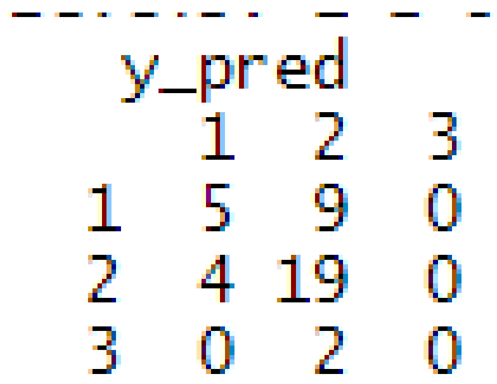
We can see by the below confusion matrix that the SVM model got a total of 10 results wrong in total. This indicates that the model was 75 percent accurate (in general). The model did a great job classifying cluster 2 data. This mostly has to do with the fact that cluster 2 is the largest cluster of them all, and thus the model is used to seeing 2. One thing that is certain, since cluster 1 and cluster 3 appear less in the dataset than cluster 2, the data we are feeding the model is skewed and thus results in cluster 2 being correctly predicted more often than cluster 3 and 1. However there is not exactly a fix to the skew of the data because there are exactly 50 states, and thus one record per each state. Since we cannot append anymore records to the dataset, there is no way to balance it. Furthermore, we cannot improve the performance of this model other than using some form of regularization (which will potentially help with the overfitting of the model).

Classification using Naive Baye's

We will be using the exact same dataset as with the SVM model. In addition we note that the data is already standardized here as well.

Generally speaking the Naive Baye's classifier works very well for classification problems where we have a very small dataset. However Naive Baye's also has a big flaw...it tends to perform significantly poorly on data that is highly linearly dependent. If we recall the way that majority of the data columns we have were generated by doing column X column calculations, implying that there is a strong relationship between some of our features. In addition to this we saw previously that there is strong collinearity between number of cases and number of deaths with respect to individual states. Furthermore, it is not at all expected that this model will perform well on the data.

View of confusion matrix for Naive Baye's model:



```
   y_pred
     1  2  3
  1  5  9  0
  2  4 19  0
  3  0  2  0
```

Looking at this confusion matrix above, it is very clear that the NB model classified the data significantly worse than that of the SVM.

Actionable Insights (Results & Discussion)

With the SVM we only got 10 records predicted wrong, however, in the case of the NB model, we actually mis-classified 15 records wrong. We can also take note that this model handled cluster 2 moderately well, but did a horrible job classifying for cluster 1 and for cluster 3. Again this is most likely a product of the skew of the dataset being passed into it. Not only do we have a small amount of records, but we also have a very uneven balance of classes to classify, thus causing suveer overfitting for this model. In addition to this, this model also suffers from the co-linear dependence that was spoken about previously. This most likely has to do with the fact that Naive Baye's assumes that there is no linear dependence in the features that we have. In the case of our data, many of those columns are highly correlated, as some columns are actually generated directly off of one another.

Furthermore, it appears that our SVM model performed the best at predicting the expected cluster that each state would be assigned into (indicating the efficiency of the state during the covid-19 pandemic).

Potential Future Experiments & Expectations

I think that a future experiment of interest would be attempting to take the same dataset and run it through a KNN model, as that model tends to perform very well when we have a dataset with a low dimensionality (very few features). As stated before, we do not have a lot of data to run in the training for this dataset (because of the nature of the problem itself), and thus it is difficult for any model in general to avoid overfitting.

If I had to guess at the model that would perform the best with the specific dataset we are using, it would either be gradient boosting trees or a random forest. Both of these algorithms are ensemble methods built upon regular classification trees. The main flaw of these models is that they cannot "extrapolate", however, in the case of this classification problem that we are currently running the extrapolation would not be a problem. Random forests are also really great on both high dimensional datasets as well as datasets that do not contain a lot of rows in them. Thus in the case of the dataset we are currently using, I expect significantly better results from one of these decision tree based ensemble methods.

The model that I believe would perform the worst would be a Neural Network. Generally

NNs perform with extremely high accuracy results, but again with so little data to train a NN on, we expect that it will overfit massively. NNs require a huge amount of data in order to be correctly trained.

Appendix - Code

Majority of the code for this assignment can be found in google colab notebook here. Access most likely needs to be requested in order to view the code. I will grant access if requested.

Additional .R scripts were written to pull population data and do classification/clustering. These files will be attached in the submission along with this document.