Table of Contents

Import relevant stuff

```
data = load('COVIDbyCounty.mat');
[dates, CNTY_CENSUS,CNTY_COVID, divisionLabels, divisionNames] =
 deal(data.dates, data.CNTY_CENSUS, data.CNTY_COVID, data.divisionLabels,
 data.divisionNames);

figure(1)
plot(CNTY_COVID(10,:))
title('Plot of Individual County')
ylabel('Covid Cases/100,000 People')
xlabel('Week #')
```

```
Error using load
'COVIDbyCounty.mat' is not found in the current folder or on the MATLAB path,
 but exists in:
    C:\Users\Ryan\OneDrive\Documents\GitHub\Case-Study-1

Change the MATLAB current folder or add its folder to the MATLAB path.

Error in MasterFile (line 3)
data = load('COVIDbyCounty.mat');
```

Divide covid data by region

Initial data represents 9 regions, each with 25 counties. They are in the initial data file chronologically, with the 1st 25 making up region 1, the 2nd 25 region 2, etc.

```
% Here we just divide up the data by region, creating 9 25x130 matrices.

region1 = CNTY_COVID(1:25,:);
region2 = CNTY_COVID(26:50,:);
region3 = CNTY_COVID(51:75,:);
region4 = CNTY_COVID(76:100,:);
region5 = CNTY_COVID(101:125,:);
region6 = CNTY_COVID(126:150,:);
region7 = CNTY_COVID(151:175,:);
```

```
region8 = CNTY_COVID(176:200,:);
region9 = CNTY_COVID(201:225,:);
```

Calc average of each region

Here we use the matrices from above to calculate the average for each region at each timepoint. We don't actually use this data for our method, but reasoned a visual representation of this data could be handy.

```
ave1 = mean(region1,1);
ave2 = mean(region2,1);
ave3 = mean(region3,1);
ave4 = mean(region4,1);
ave5 = mean(region5,1);
ave6 = mean(region6,1);
ave7 = mean(region7,1);
ave8 = mean(region8,1);
ave9 = mean(region9,1);

aves = cat(1,ave1,ave2,ave3,ave4,ave5,ave6,ave7,ave8,ave9);
figure(2)
plot(transpose(aves))
title('Averages for 9 Target Regions')
ylabel('Covid Cases/100,000 People')
xlabel('Week #')
legend('region1','region2','region3','region4','region5','region6','region7','region8','re
```

Calc standard deviation of each timepoint by region

The first step of our method, which is based entirely on standard deviation. Here, though, we are just using the region matrices to generate a new 9x130 matrix composed of the STDVs of each region at each time point. Row 1 represents region 1's 130 STDVs, and so on.

```
S1 = std(region1,0,1);
S2 = std(region2,0,1);
S3 = std(region3,0,1);
S4 = std(region4,0,1);
S5 = std(region5,0,1);
S6 = std(region6,0,1);
S7 = std(region7,0,1);
S8 = std(region8,0,1);
S9 = std(region9,0,1);

comp = cat(1,S1,S2,S3,S4,S5,S6,S7,S8,S9);
all = std(comp,0,1);
[maxes,idxs] = maxk(all,10);
```

METHOD 1

Generate ratio proportional to STDV of region 'x' divided by sum of STDVs of all other regions. Idea is small STDV makes ratio smaller, large STDVs for other regions makes ratio smaller. We want the indices with small values.

```
targetIDXs = zeros(9,130);                                                  %
 Initialize a matrix to deposit the ratios

i = 1;
while i <= 9                                                                %
 Iterate over all rows
    j = 1;
    while j <= 130                                                          %
 Iterate over all columns in the ith row
        targetIDXs(i,j) = comp(i,j) * 8 / (sum(comp(:,j)) - comp(i,j)); %
 Ratio calculation, multiplied by 8 to keep closer to 1.
        j = j + 1;
    end
    i = i + 1;
end

[targets,mins] = mink(targetIDXs,40,2);                                     % Find
 the k smallest ratios, to be used in the next step

uniques = unique(mins);                                                     %
 Repeats not necessary
taking = [];
i = 1;
while i <= 125
    if size(find(mins == uniques(i,1))) < 4                                 % This
 whole loop is designed to ELIMINATE overly-commmon indices. If an index is
        taking(end + 1) = uniques(i,1);                                     % good
 for every region, its good for no region.
    end
    i = i + 1;
end
```

Generating identity matrix that targets particular indeces

```
oddID = zeros(130,130);

i = 1;
while i <= 19                                                               % The
 first few identified points only work well. Too many points and the data all
 misclusters
    oddID(taking(i),taking(i)) = 1;                                         %
 together. Too much noise and it can't distingush based on markers anymore.
    i = i + 1;
end
```

Multiplies data by this new identity matrix

```
newData = CNTY_COVID;                                                       %
 Copied so we don't ruin base data
```

```
sz = 225;
i = 1;
while i <= sz
    ele = newData(i,:);
    newData(i,:) = ele * oddID;                                                    %
 Multiply every row in base data by the augmenting identity matrix
    i = i + 1;
end
```

Generate test and sample matrices

```
split = newData;
sample = zeros(180,130);                                                           %
 Initialize matrices for two data sets; one helps identify centroids, other is
 clustered
test = zeros(45,130);                                                              % using
 those.

i = 1;
j = 1;
k = 1;
while i <= 225
    if rem(i,5) ~= 0                                                               % Every
 5th element is put into test data set, while the rest goes to sample set.
 Data is
        sample(j,:) = split(i,:);                                                  %
 ordered by population, so taking data all throughout the whole is necessary
 for test diversity.
        j = j + 1;
    end
    if rem(i,5) == 0
        test(k,:) = split(i,:);
        k = k + 1;
    end
    i = i + 1;
end
```

Run Kmeans, get centroids to test on test group

```
[indeces, centroids] = kmeans(sample,9);

ref = zeros(1,9);
i = 1;
while i <= 9
    ref(1,i) = mode(indeces(((i-1)*20 + 1):i*20,1));                               % Loop
 that checks results of kmeans, identifying which cluster each region appears
 to align w/
    i = i + 1;
end
```

Test function on 45 county test group

```
[IDXresult, C1] = kmeans(test,9,'start',centroids);

i = 1;
j = 1;
total = 0;
while i <= 45
    if IDXresult(i,1) == ref(1,j)                                          % Loop
 that checks results of test kmeans against ref array, counting the number of
 successes
        total = total + 1;
    end
    if rem(i,5) == 0
        j = j + 1;
    end
    i = i + 1;
end

%silhouette(test,IDXresult)
figure(3)
plot(transpose(centroids(6:8,:)))
hold on
xlim([0 40])
```

On all 225 given counties

```
[finalResults, C2] = kmeans(newData,9,'start',centroids);
silhouette(newData,finalResults)

i = 1;
j = 1;
total2 = 0;
while i <= 225
    if finalResults(i,1) == ref(1,j)
  % Loop that checks results of test kmeans against ref array, counting the
 number of successes
        total2 = total2 + 1;
    end
    if rem(i,25) == 0
        j = j + 1;
    end
    i = i + 1;
end
```

Published with MATLAB® R2022a