Table of Contents

 1
Creating the Training and Testing Sets 1
Get rid of Outliers 2
Optomizing the Clusters 3
Values to Keep 4
Functions 5

```
load COVIDbyCounty.mat
```

Creating the Training and Testing Sets

USER: change this to use a different training/testing ratio

```
percent_training = 0.8;

% Initializing the trainingData matrix and the trainingDataLabels vectors
% so that they can be concatenated onto
% This will hold each vector in the training data that we produced and its
% 130 data points for covid data
training_data = zeros(1,130);
% this will hold the divsion number that each row vector belongs to
training_data_labels = zeros(1,1);

for num = 1:9 % for each division
%   extract the data and divsion labels cooresponding to the row vectors
%   in that division (i.e. every row vector in CNTY_COVID in division 1
%   if num = 1)
    data = CNTY_COVID((CNTY_CENSUS.DIVISION == num),:);
    labels = divisionLabels(divisionLabels == num);
%   This vector holds the indices that we want to extract from data and
%   labels, which are randomly selected. int32 is used to cast in the case
 that
%   percentTraining*size(data,1) is a decimal.
%   first param gives the range of random numbers selected
%   second param gives the number of numbers we are going to generate
%   (i.e. size of training set for each division)
    selected_indices = randperm(size(data,1), ...
        int32(percent_training*size(data,1)));
%   extracts the labels and data row vectors only cooresponding to the
%   indices which were randomly selected above
    div_training_labels = labels(selected_indices);
    div_training = data(selected_indices,:);
%   Concatenates the rows from division num that we want to use for
%   training onto the ones that we have already extracted
    training_data = [training_data; div_training];
%   Same as above, but for the division labels
    training_data_labels = [training_data_labels; div_training_labels];
```

```
end
clear num data labels div_training div_training_labels selected_indices

% get rid of column of zeros (initialized each vector/matrix with a column of
% zeros so I could concatenate onto that)
training_data = training_data(2:end,:);
training_data_labels = training_data_labels(2:end);
% Concatenates the training data labels onto training data so we know which
% vectors belong in each division
training_data = [training_data_labels training_data];
clear("training_data_labels");

% Creates the testing set (i.e. the data points remaining in CNTY_COVID
% that are not in the training set
testing_index = ~ismember(CNTY_COVID,training_data(:,2:end), 'rows');
temp_CNTY_COVID = [divisionLabels CNTY_COVID];
testing_data = temp_CNTY_COVID(testing_index,:);
clear temp_CNTY_COVID testing_index
```

Get rid of Outliers

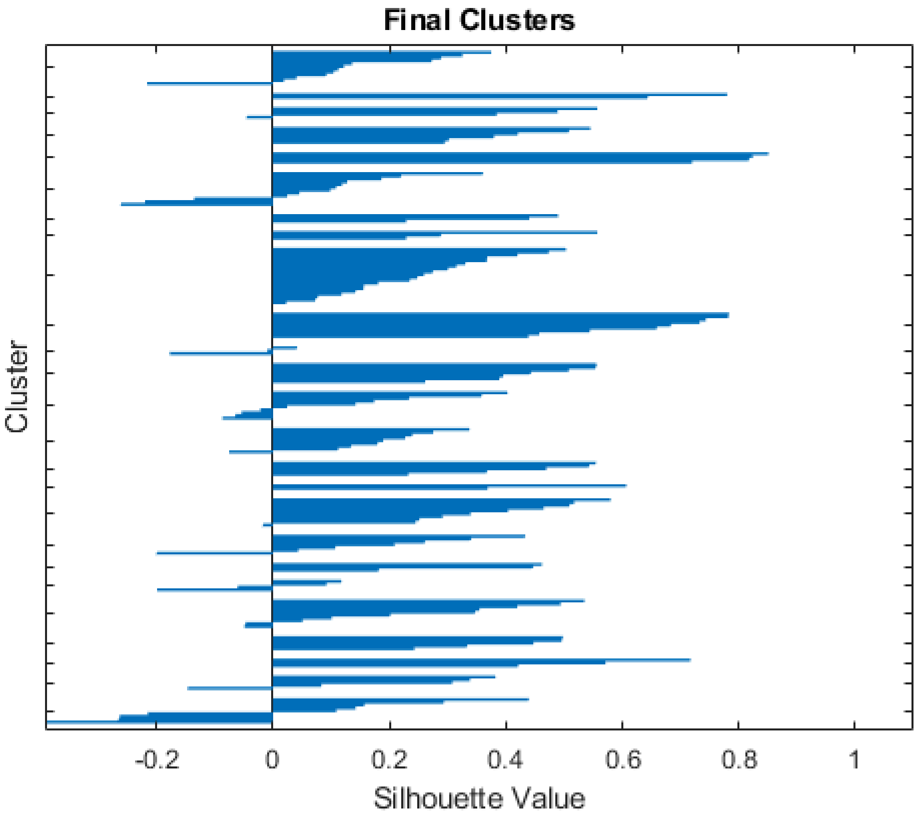
```
training_data_before_while = training_data;
outliers_idx = ones(180);


while any(outliers_idx)
    %disp('start');

    k = 30;
    clust_idx = kmeans(training_data(:,2:end),k,'replicates', 200);

    % USER: change this number to change the amount of clusters
    training_data = [clust_idx training_data];
    % sorts training data according to the cluster values so you can scroll
 and
    % visually see which division regions are part of each cluster
    %training_data = sortrows(training_data, 1);
%     figure
%     silhouette(training_data(:,3:end),clust_idx);
    s_vals = silhouette(training_data(:,3:end),clust_idx);

    % Find outliers aka when silhouette val = 1
    outliers_idx = s_vals(:,1) == 1;

    %Remove outliers
    training_data(outliers_idx,:) = [];

    %Get rid of cluster label so it doesn't affect next round of kmeans
    training_data(:,1) = [];
    %disp('end')
end
%disp('outliers done');
```

Optomizing the Clusters

The following code runs k-means with k values ranging from 9 to a max_k_val value. For each k value, kmeans is run multiple times, as each run of kmeans can lead to different results. The centroids are then assigned to divisions using the centroid_division method, and those that yeild the highest success rate of assigning the training data to the correct cluster are kept to be used as the final set of centroids.

```
max_k_val = 30;
clusters_max = 0;
percent_max = 0;
k_max = 0;
centroid_division_assignments_max = 0;

for k = 9:max_k_val
    for g = 1:20
        %Run K means
        %disp('k means')
        [clust_idx,clusters] = kmeans(training_data(:,2:end),k,...
            'replicates', 200);
        training_data = [clust_idx training_data];
        [centroid_division_assignments, table] = centroid_division(k,...
            training_data);

        %Assign Centroids
        %disp('centroids')
        nearest_neighbors_idx = knnsearch(clusters,testing_data(:,2:end));
        testing_data_results = [nearest_neighbors_idx ...
        centroid_division_assignments(nearest_neighbors_idx, 1) ...
        testing_data(:,1)];

        training_data_results_table = array2table(testing_data_results, ...
            "VariableNames",{'Cluster', 'Assigned Division', ...
            'Actual Division'});

        num_correct = testing_data_results(:,2) == ...
            testing_data_results(:,3);
        num_correct = sum(num_correct);
        percent_numCorrect = num_correct/size(testing_data_results,1);
        %disp(percent_numCorrect)
        if (percent_numCorrect > percent_max)
            percent_max = percent_numCorrect;
            %disp(percent_max)
            k_max = k;
            %disp(k)
            clust_idx_max = clust_idx;
            clusters_max = clusters;
            centroid_division_assignments_max = ...
                centroid_division_assignments;
        end
        %Get rid of cluster label so it doesn't affect next round of kmeans
        training_data(:,1) = [];
    end
end
```

```
figure
silhouette(training_data(:,2:end),clust_idx_max);
s_vals = silhouette(training_data(:,2:end),clust_idx_max);
title('Final Clusters')
```



Values to Keep

Program saves the training data set, the testing data set, and the centroids in training_data, testing_data, and clusters respectively.

```
clust_idx = clust_idx_max;
training_data = [clust_idx training_data];
clusters = clusters_max;
centroid_division_assignments = centroid_division_assignments_max;

save training_data.mat
save testing_data.mat
save clusters.mat
save centroid_division_assignments.mat
```

%%%

Functions

This function assigns centroid numbers to a division based on what division makes up the majority of a cluster INPUTS: num_clusters (number of clusters, scalor) and data (matrix of sorted COVID time series data with cluster assignments in the first row) OUTPUT: division_number, a num_clusters x 6 matrix that holds the top 3 clusters to appear in each cluster in rows 1-3, and the names of those clusters in rows 4-6. table is the same as division_number but formatted nicely

```
function [division_number,table] = centroid_division(num_clusters, data)
    x = zeros(num_clusters,3);

    % Most common division
    for clust = 1:num_clusters
        subdata = data(data(:,1) == clust, 2);
        [mode1,freqMode1] = mode(subdata);
        x(clust,1) = mode1;
        mean_clust = mean(subdata);
        x(clust,7) = mean_clust;
        x(clust,4) = freqMode1;
        x(clust,8) = size(subdata,1);

    end
    % Second most common division
    for clust = 1:num_clusters
        subdata = data(data(:,1) == clust, 2);
        [mode2, freqMode2] = mode(subdata(subdata ~= mode(subdata)));
        x(clust,2) = mode2;
        x(clust,5) = freqMode2;

    end
    % Third most common division
    for clust = 1:num_clusters
        subdata = data(data(:,1) == clust, 2);
        first = mode(subdata);
        second = mode(subdata(subdata ~= mode(subdata)));
        [mode3, freqMode3] = mode(subdata(subdata ~= first & ...
            subdata ~= second));
        x(clust,3) = mode3;
        x(clust,6) = freqMode3;
    end
    % Setting return value
    division_number = x;
    table_labels = {'Most Common Div', '2nd Most Common Div',...
    '3rd Most Common Div','FreqMode1', 'FreqMode2', 'FreqMode3',...
    'Mean','ClusterLength'};
    centroid_division_assignments_table = array2table(x,'VariableNames',...
        table_labels);
    clear table_labels
    table = centroid_division_assignments_table;

end
```

Published with MATLAB® R2022a