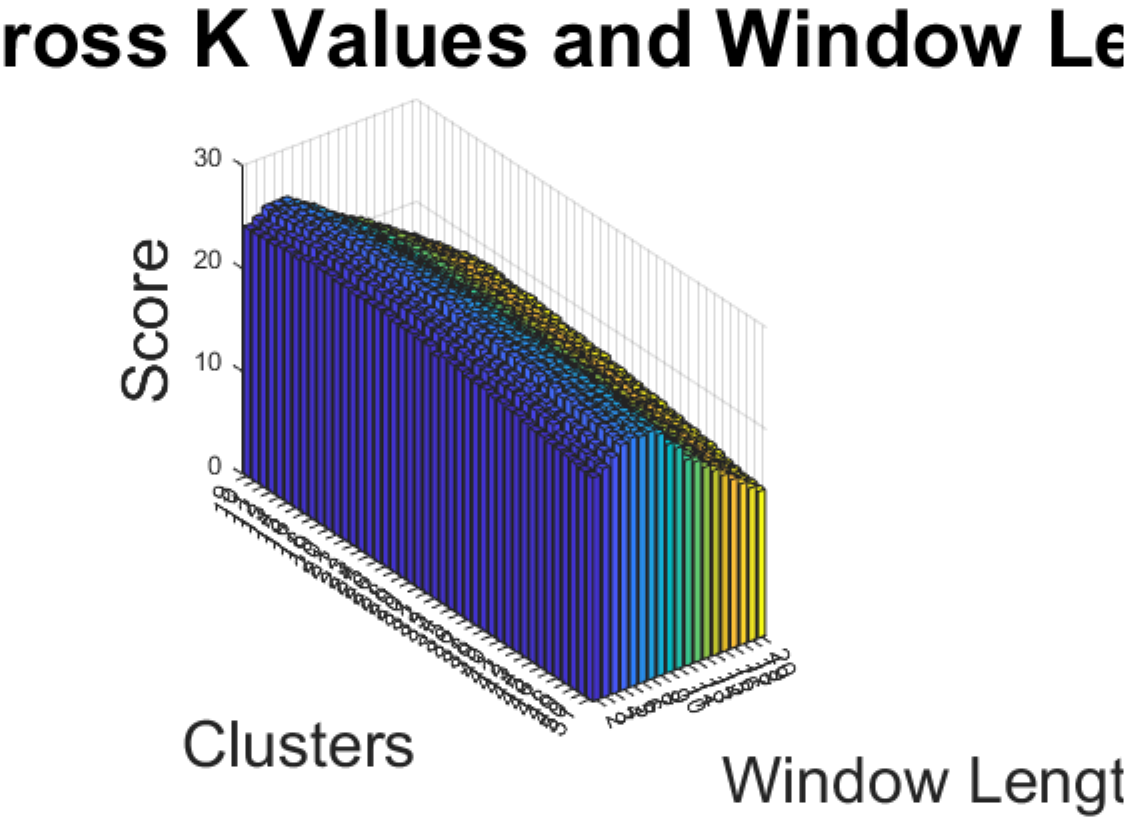
```
%eparates the data into "training" and “testing” groups, uses kmeans
 clustering
% on the “training” group, and results in the construction of k centroids

%----------Load Data----------

close all
load('COVIDbyCounty.mat');

%----------Seperate data into testing and training----------

minDivisionCount = intmax; %Number of counties in smallest division
trainingCensus = sortrows(CNTY_CENSUS, 3, "ascend"); % Sort by division

% Count counties number in each division
for i = 1 : 9
    quantities = 0;
    for j = 1 : 225
        if trainingCensus.DIVISION(j) == i
            quantities = quantities + 1;
        end
    end
    minDivisionCount = min(minDivisionCount, quantities);
    % disp("Division " + i + " has " + quantities + " counties");
end
disp("Every division has at least " + minDivisionCount + " counties");

test = 9; % How any test samples from each division (removed from training)
for div = 1 : 9
    for i = 1 : test
        % find a random number from 1 to 25 - (i - 1)
        rand = int32(randi([1,25 - (i - 1)]));
        % Find the random row and assign it to testing data
        testingCensus((div - 1) * test + i, :) = trainingCensus((div - 1) *
 (minDivisionCount-test) + rand,:);
        % Remove the testing data from training data table
        trainingCensus((div - 1) * (minDivisionCount-test) + rand, : ) = [];
    end
end

%Sort rows based on fips code, to match intersect function output order
trainingCensus = sortrows(trainingCensus, 1, "ascend");
testingCensus = sortrows(testingCensus, 1, "ascend");

%Find the rows of the dataset in order to extract from CNTY_COVID
[~,trainingIdx] = intersect(CNTY_CENSUS.fips, trainingCensus.fips);
[~,testingIdx] = intersect(CNTY_CENSUS.fips, testingCensus.fips);

%Extract data
trainingCases = CNTY_COVID(trainingIdx,:);
testingCases = CNTY_COVID(testingIdx,:);
```

```
testing_labels = testingCensus.DIVISION;

%----------Determine optimal clustering parameters----------

minK = 9;  %K-means k value (number of clusters)
maxK = 51;
minW = 2;  % Block average window length
maxW = 20;
iterations = 1000; % Average across (1000 takes 16 minutes on my machine,
 reduce for faster runtime)

K = minK:maxK;
W = minW:maxW;

scores = zeros(maxK, maxW, iterations);
trainingDivisions = trainingCensus.DIVISION;

parpool(24);
for w = W
    tic
    parfor k = K % Parallel for loop here significantly decreases runtime
        rng(w * k);
        % Transform training and test data based on window length
        A = generateBlockAverageMatrix(length(dates), w);
        transformedTrainingCases = (A * trainingCases')';
        transformedTestingCases = (A * testingCases')';

        % Average to remove k-means randomness
        for i = 1:iterations
            % Run k-means with current k value
            [centroidIdx, centroids] = kmeans(transformedTrainingCases, k);

            % Label centroids based on most common division within their
 cluster
            centroid_labels = zeros(k,1);
            for centroid = 1:k
                cluster = centroidIdx == centroid;
                centroid_labels(centroid) = mode(trainingDivisions(cluster));
            end

            % Save score
            score = checkTestResult(centroids, centroid_labels,
 testing_labels, transformedTestingCases);
            % scores(k-minK+1,w) = scores(k-minK+1,w) + score;
            scores(k, w, i) = score;
        end
    end

    %Log progreess to console
    toc
    disp(w + "/" + maxW);
end
scores = mean(scores, 3);
```

```
%----------Plot Results----------

fontSize = 24;

bar3(scores)

title("Score Across K Values and Window Lengths", 'FontSize', fontSize + 6)
xlabel("Window Length", 'FontSize', fontSize)
ylabel("Clusters", 'FontSize', fontSize)
zlabel("Score", "FontSize", fontSize)

xlim([W(1)-0.5, W(end)+0.5])
set(gca, 'XTick', W)
ylim([K(1)-0.5, K(end)+0.5])
set(gca, 'YTick', K)

%----------Save Results----------

% Find maximum score and save indices that lead to it
[bestK, bestW] = find(ismember(scores, max(scores(:))));
disp("Using a window length of " + bestW + " and " + bestK + " clusters.")

A = generateBlockAverageMatrix(length(dates), bestW(1));
transformedTrainingCases = (A * trainingCases')';

% Perform Clustering
[centroidIdx, centroids, sumd] = kmeans(transformedTrainingCases, bestK(1));

% Create the centroid labels based on the most common division in each
% cluster
centroid_labels = zeros(bestK,1);
for centroid = 1:bestK
    cluster = centroidIdx == centroid;
    centroid_labels(centroid) = mode(trainingCensus.DIVISION(cluster));
    disp("Centroid " + centroid + " assigned to division " +
 centroid_labels(centroid))
end

save("competition.mat", "centroids", "centroid_labels", "A");

%---------Utility Functions---------

function A = generateBlockAverageMatrix(n, window)
    block = ones(1,window)/window;
    A=zeros(n-window,n);
    for i = 1:n-window+1
        row = [zeros(1,i-1), block, zeros(1,n-i-window+1)];
        A(i,:) = row;
    end
end

function score = checkTestResult(centroids, centroid_labels, testing_labels,
 testingCases)
    nCorrect = 0;
    for i = 1:height(testingCases)
        testCase = testingCases(i,:);

        [~, assignedCentroid] = min(pdist2(centroids,testCase, 'euclidean'));
        centroid_label = centroid_labels(assignedCentroid);
        trueDivision = testing_labels(i);

        if(trueDivision == centroid_label)
            nCorrect = nCorrect + 1;
        end
    end

    score = nCorrect - 0.5*height(centroids);
end
```

```
Every division has at least 25 counties
Starting parallel pool (parpool) using the 'Processes' profile ...
Connected to parallel pool with 24 workers.
Elapsed time is 30.225649 seconds.
2/20
Elapsed time is 20.193887 seconds.
3/20
Elapsed time is 19.787015 seconds.
4/20
Elapsed time is 20.348116 seconds.
5/20
Elapsed time is 19.925892 seconds.
6/20
Elapsed time is 20.178259 seconds.
7/20
Elapsed time is 19.932932 seconds.
8/20
Elapsed time is 20.032502 seconds.
9/20
Elapsed time is 20.103533 seconds.
10/20
Elapsed time is 19.816013 seconds.
11/20
Elapsed time is 19.795591 seconds.
12/20
Elapsed time is 19.678293 seconds.
13/20
Elapsed time is 19.632877 seconds.
14/20
Elapsed time is 19.463206 seconds.
15/20
Elapsed time is 19.638776 seconds.
16/20
Elapsed time is 19.855146 seconds.
17/20
Elapsed time is 19.698224 seconds.
18/20
Elapsed time is 19.711657 seconds.
```

```
19/20
Elapsed time is 19.604370 seconds.
20/20
Using a window length of 5 and 25 clusters.
Centroid 1 assigned to division 1
Centroid 2 assigned to division 1
Centroid 3 assigned to division 9
Centroid 4 assigned to division 1
Centroid 5 assigned to division 4
Centroid 6 assigned to division 7
Centroid 7 assigned to division 3
Centroid 8 assigned to division 4
Centroid 9 assigned to division 8
Centroid 10 assigned to division 4
Centroid 11 assigned to division 3
Centroid 12 assigned to division 7
Centroid 13 assigned to division 7
Centroid 14 assigned to division 8
Centroid 15 assigned to division 2
Centroid 16 assigned to division 8
Centroid 17 assigned to division 5
Centroid 18 assigned to division 3
Centroid 19 assigned to division 5
Centroid 20 assigned to division 6
Centroid 21 assigned to division 5
Centroid 22 assigned to division 9
Centroid 23 assigned to division 2
Centroid 24 assigned to division 1
Centroid 25 assigned to division 4
```



Published with MATLAB® R2023a