## Contents


- **HW #2: Least Square and Spectral Clustering Algorithms**
- 1.a. Represent weighted adjacency matrix as Laplacian
- 1.b. Choosing k constant
- 1.c. Elbow method
- 1.d. Plot eigenvalues
- 1.e. Unnormalized spectral clustering
- 1.f. normalized spectral clustering (Shi and Malik 2000)
- 1.g. normalized spectral clustering (Ng, Jordan, Weiss 2002)
- 1.h. Plot the results of this clustering
- 2.i. Results
- 2.j. Alternative Clustering: Agglomerative Hierarchical Clustering Tree


## HW #2: Least Square and Spectral Clustering Algorithms

Implementing spectral clustering algorithm on chromosome HI-C data for TAD identification

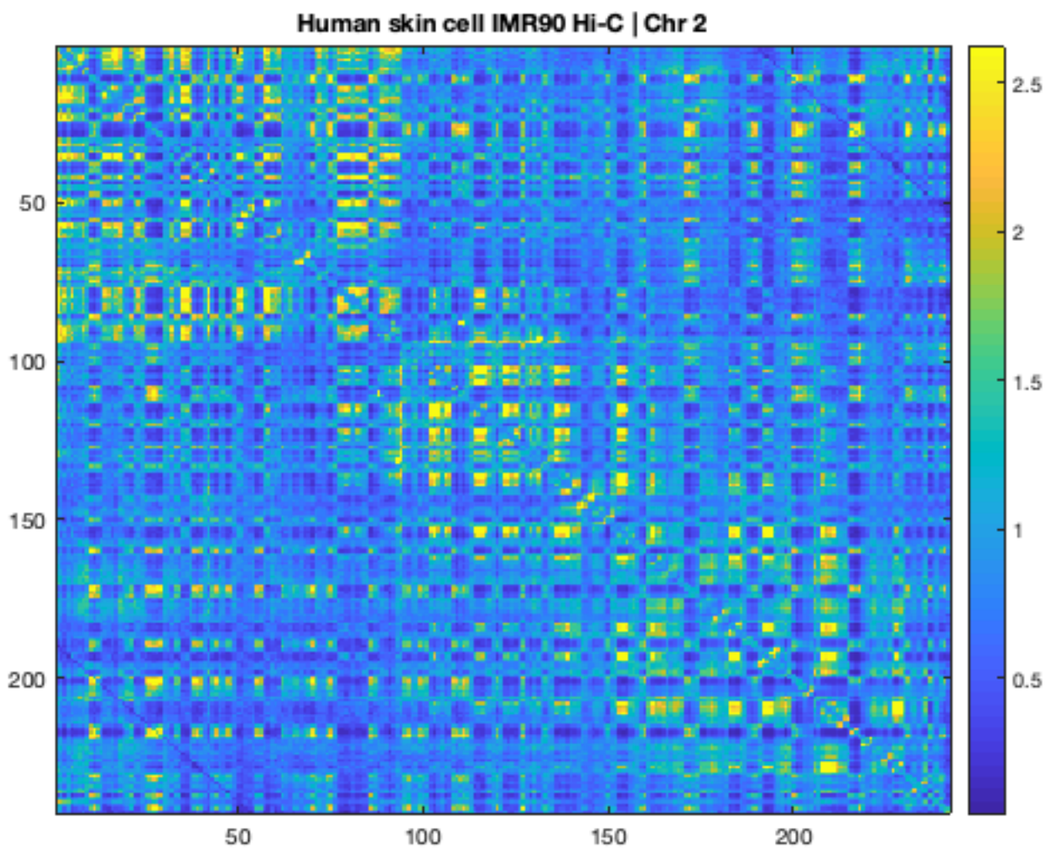

Notebook Author: Koki Sasagawa


Date: 3-12-2019

```
% setup
clear
close all
```

Perform 3 spectral clustering algorithms outlined in "A tutorial on spectral clustering" by Ulrike Von Luxburg, 2007

```
% load weighted adjacency matrix
load('imr90chr2.mat')

figure(2);
imagesc(imr90chr2)
title('Human skin cell IMR90 Hi-C | Chr 2')
colorbar
```




## 1.a. Represent weighted adjacency matrix as Laplacian

$L = D - W$

Where

- D - degree matrix
- W - edge matrix

The degree of a vertex $v_i \in V$ is defined as

$d_i = \sum_{j=1}^{n} w_{ij}$

Calculate unnormalized graph Laplacian

```
degrees = sum(imr90chr2, 2);
D_matrix = diag(degrees);

% Unnormalized Laplacian
L = D_matrix - imr90chr2;
```

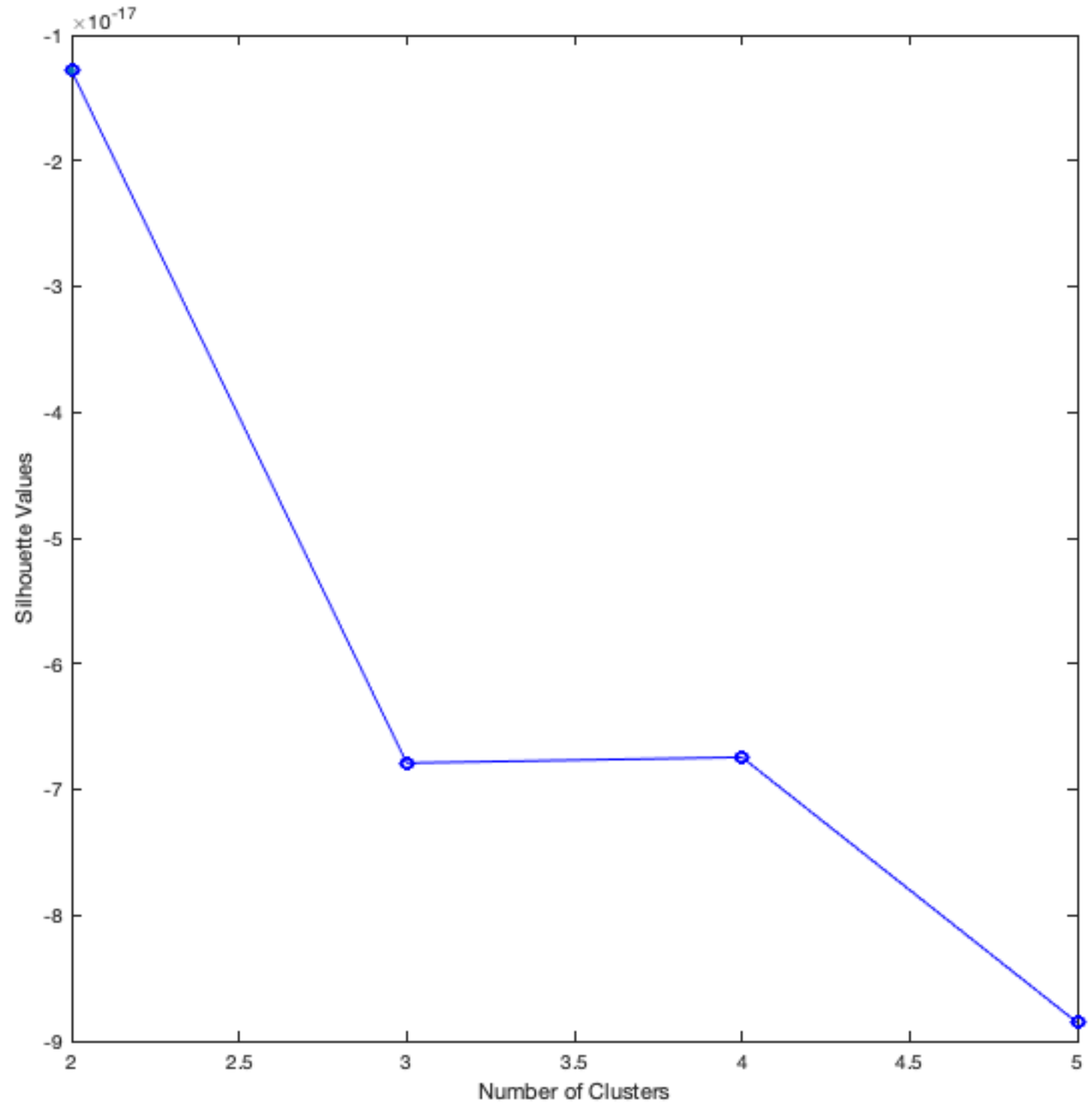
### 1.b. Choosing k constant

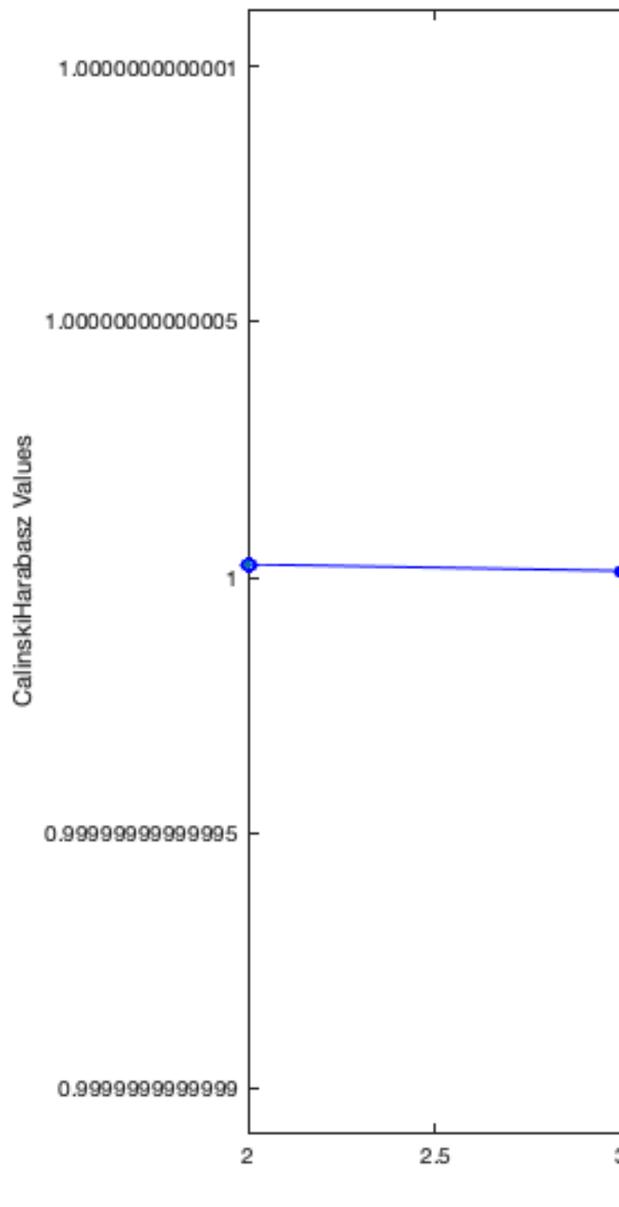
Evaluate k using different metrics

- Silhouette
- Calinski Harabasz
- Davies Bouldin

```
[V,~] = eig(L);

% clust = zeros(size(V,1), 10);
%
% for i=1:10
%     clust(:,i) = kmeans(V, i, 'EmptyAction', 'singleton', 'replicate', 20);
% end
%
% eva1 = evalclusters(V, clust, 'Silhouette')
% eva2 = evalclusters(V, clust, 'CalinskiHarabasz')
% eva3 = evalclusters(V, clust, 'DaviesBouldin')

eva1 = evalclusters(V, 'kmeans', 'Silhouette', 'klist', [1:5])
eva2 = evalclusters(V, 'kmeans', 'CalinskiHarabasz', 'klist', [1:5])
eva3 = evalclusters(V, 'kmeans', 'DaviesBouldin', 'klist', [1:5])

% Plot out results
fig(3) = figure(3);
set(fig(3), 'unit', 'normalized', 'Position', [.15 .15 .7 .4])

for k = 1:3
    ax(k) = subplot(1,3,k);
end

subplot(ax(1))
plot(eva1)
axis('square')

subplot(ax(2))
plot(eva2)
axis('square')

subplot(ax(3))
plot(eva3)
axis('square')
```

```
eva1 =

  SilhouetteEvaluation with properties:

    NumObservations: 242
         InspectedK: [1 2 3 4 5]
    CriterionValues: [NaN -1.2846e-17 -6.7898e-17 -6.7439e-17 -8.8543e-17]
           OptimalK: 2


eva2 =

  CalinskiHarabaszEvaluation with properties:

    NumObservations: 242
         InspectedK: [1 2 3 4 5]
    CriterionValues: [NaN 1.0000 1.0000 1.0000 1.0000]
           OptimalK: 2


eva3 =

  DaviesBouldinEvaluation with properties:

    NumObservations: 242
         InspectedK: [1 2 3 4 5]
    CriterionValues: [NaN 15.2315 12.2734 11.4555 7.5185]
           OptimalK: 5
```



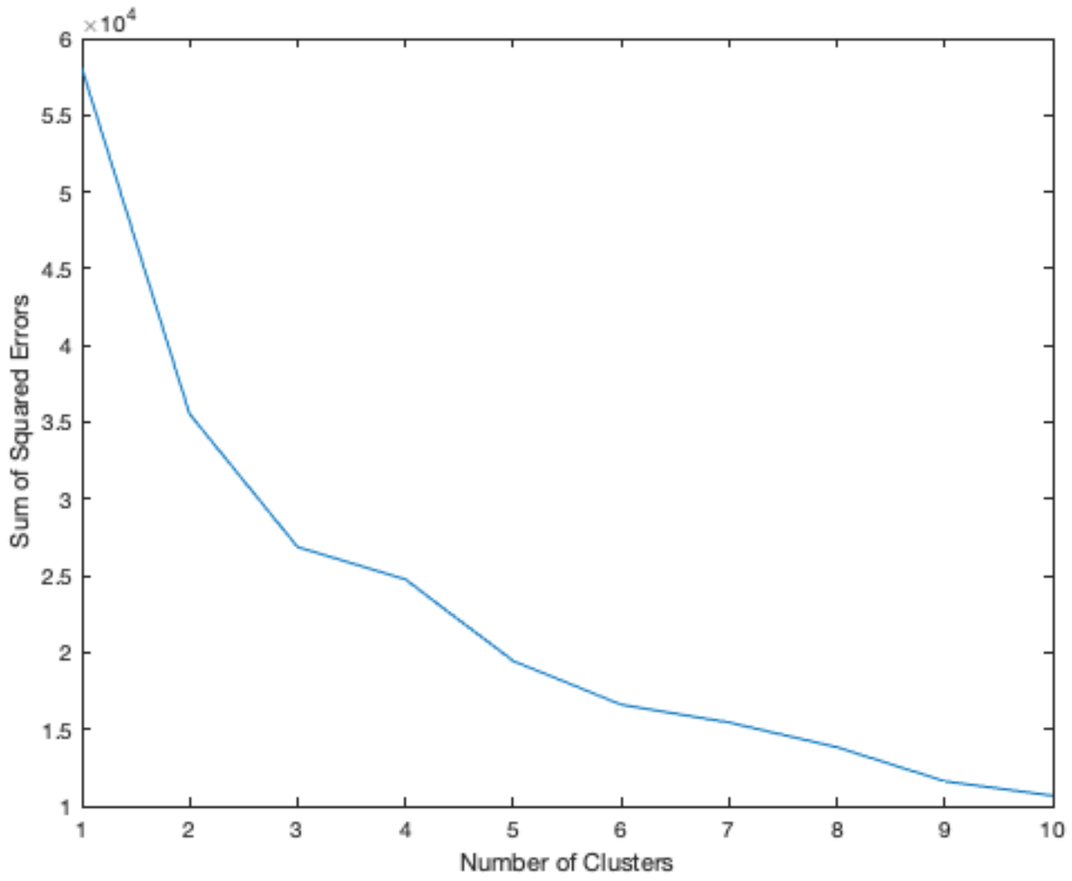




### 1.c. Elbow method

sumd - within-cluster sums of point-to-centroid distances in the k-by-1 vector sumd

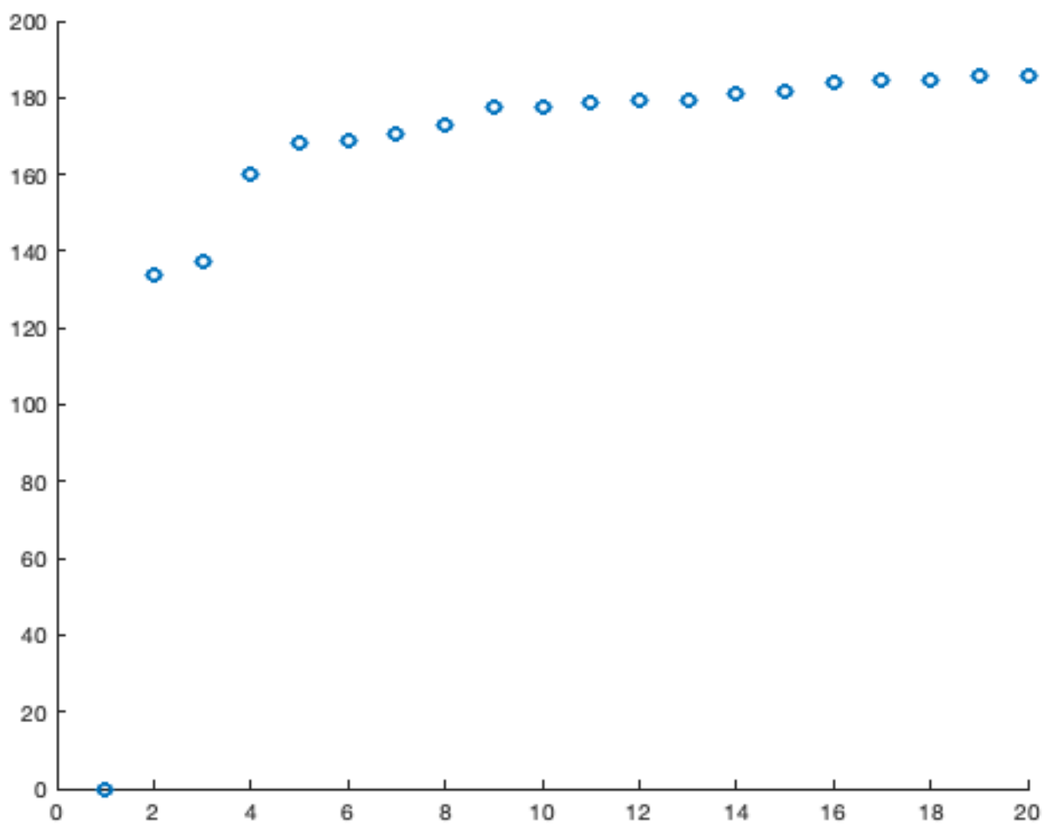
Plot the sum of the squared error values

```
elbow = zeros(size(10, 1), 1);

for i=1:10
    rng(1);
    [~,~,sumd] = kmeans(V, i, 'EmptyAction', 'singleton');
    elbow(i) = sum(sumd.^2);
end

figure(4)
plot(elbow)
ylabel('Sum of Squared Errors')
xlabel('Number of Clusters')
```




### 1.d. Plot eigenvalues

```
figure(5);
[~,S] = eig(L);
S = diag(S);
scatter(1:20, S(1:20))
```




### 1.e. Unnormalized spectral clustering

- S = diagonal matrix S of eigenvalues
- U = matrix V whose columns are right eigenvectors such that $A * V = V * D$

```
% Initialize k constant: Average of all results
k = 3;

% eigen decomposition
[U, S] = eig(L);

% choose first k minimum eigenvalues
min_vals = abs(diag(S));
[val, idx] = sort(min_vals, 'ascend');

% select k corresponding eigenvectors
Uk = U(:, idx(1:k));

tic; % Start stopwatch timer
% Kmeans parameters
% * Display - level of output to display
% * Distance - distance metric
% * EmptyAction - action to take if cluster loses all member observations
% * MaxIter - max number of iterations
% * Replicates - number of times to repeat clustering using new insitial
%   cluster centroid positions
C = kmeans(Uk, k, 'Display', 'final', 'Distance','sqeuclidean', ...
```

```
    'EmptyAction', 'singleton', 'MaxIter', 100, 'Replicates', 5);
toc % Stop stopwatch timer
```

```
Replicate 1, 4 iterations, total sum of distances = 0.289561.
Replicate 2, 6 iterations, total sum of distances = 0.289561.
Replicate 3, 7 iterations, total sum of distances = 1.15485.
Replicate 4, 1 iterations, total sum of distances = 0.289561.
Replicate 5, 6 iterations, total sum of distances = 0.289561.
Best total sum of distances = 0.289561
Elapsed time is 0.007204 seconds.
```

### 1.f. normalized spectral clustering (Shi and Malik 2000)

Normlalized Laplacian $L_{rw}$

Two ways:

- $L_{rw} = D^{-1}L$
- Generalized eigen decomposition

```
% Multiply L by inv of degree matrix
% calculate inverse of Degree matrix (D_matrix)
D_inv = inv(D_matrix);
L_rw = D_inv * L;
[U, S] = eig(L_rw); % Where U is now generalized eigenvectors

% 2. Generalized eigen decomposition
% [U, S] = eig(L, D_matrix);

% choose first k minimum eigenvalues
min_vals = abs(diag(S));
[val, idx] = sort(min_vals, 'ascend');

% select k corresponding eigenvectors
Uk = U(:, idx(1:k));

tic; % Start stopwatch timer
% Kmeans parameters
% * Display - level of output to display
% * Distance - distance metric
% * EmptyAction - action to take if cluster loses all member observations
% * MaxIter - max number of iterations
% * Replicates - number of times to repeat clustering using new insitial
%   cluster centroid positions
C2 = kmeans(Uk, k, 'Display', 'final', 'Distance','sqeuclidean', ...
    'EmptyAction', 'singleton', 'MaxIter', 100, 'Replicates', 5);
toc % Stop stopwatch timer
```

```
Replicate 1, 10 iterations, total sum of distances = 0.498637.
Replicate 2, 10 iterations, total sum of distances = 0.498637.
Replicate 3, 3 iterations, total sum of distances = 0.498637.
Replicate 4, 6 iterations, total sum of distances = 0.498637.
Replicate 5, 3 iterations, total sum of distances = 0.498637.
Best total sum of distances = 0.498637
Elapsed time is 0.008038 seconds.
```

### 1.g. normalized spectral clustering (Ng, Jordan, Weiss 2002)

compute normalized Laplacian $L_{sym}$ Form the matrix T from U by normalizing the rows to norm 1

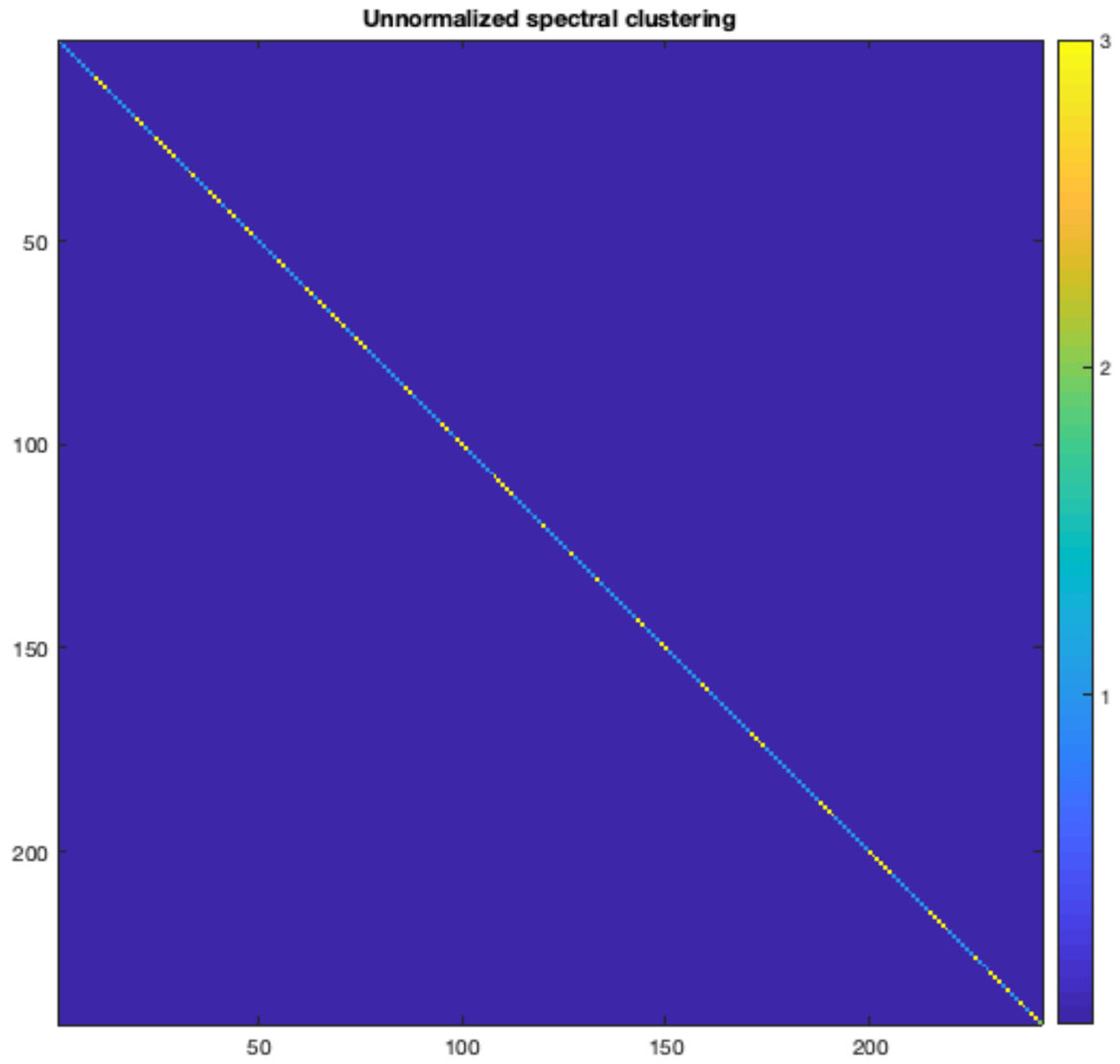
```
% calculate D^(-1/2)
D_inv = spdiags(1./(degrees.^0.5), 0, size(D_matrix, 1), size(D_matrix, 2));

% calculate normalized Laplacian
L_sym = D_inv * L * D_inv;

% eigen decomposition
[U, S] = eig(L_sym);

% Normalize eigenvectors row-wise
U = bsxfun(@rdivide, U, sqrt(sum(U.^2, 2)));

% choose first k minimum eigenvalues
min_vals = abs(diag(S));
[val, idx] = sort(min_vals, 'ascend');

% select k corresponding eigenvectors
Uk = U(:, idx(1:k));

tic; % Start stopwatch timer
% Kmeans parameters
% * Display - level of output to display
% * Distance - distance metric
% * EmptyAction - action to take if cluster loses all member observations
```

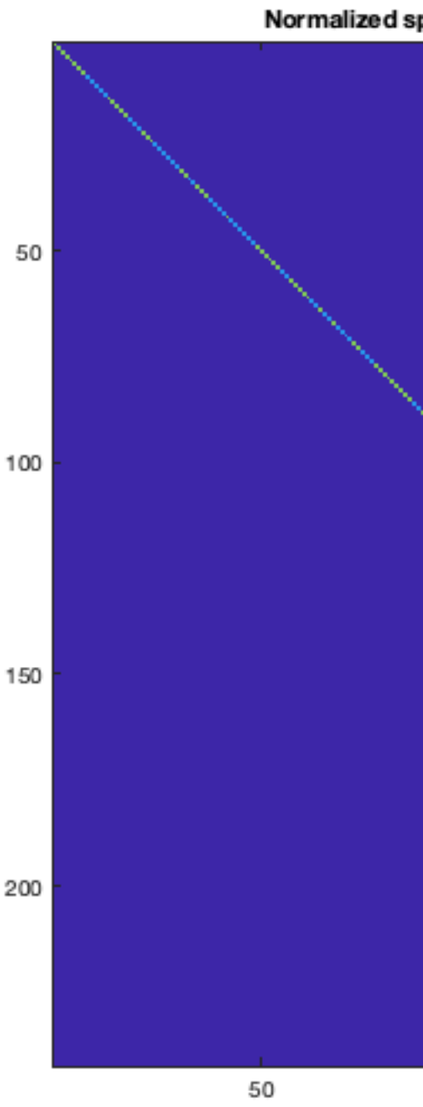
```
% * MaxIter - max number of iterations
% * Replicates - number of times to repeat clustering using new insitial
%   cluster centroid positions
C3 = kmeans(Uk, k, 'Display', 'final', 'Distance','sqeuclidean', ...
    'EmptyAction', 'singleton', 'MaxIter', 100, 'Replicates', 5);
toc % Stop stopwatch timer
```

```
Replicate 1, 6 iterations, total sum of distances = 0.490924.
Replicate 2, 11 iterations, total sum of distances = 1.09984.
Replicate 3, 7 iterations, total sum of distances = 0.490924.
Replicate 4, 5 iterations, total sum of distances = 0.490924.
Replicate 5, 4 iterations, total sum of distances = 0.490974.
Best total sum of distances = 0.490924
Elapsed time is 0.008617 seconds.
```

### 1.h. Plot the results of this clustering

```
fig(6) = figure(6);
set(fig(6), 'unit', 'normalized', 'Position', [.15 .15 .7 .4])

for k = 1:3
    ax(k) = subplot(1,3,k);
end

subplot(ax(1))
imagesc(diag(C))
cbr = colorbar;
set(cbr, 'YTick', 1:1:4);
title('Unnormalized spectral clustering')
axis('square')

subplot(ax(2))
imagesc(diag(C2))
cbr = colorbar;
set(cbr, 'YTick', 1:1:4);
title('Normalized spectral clustering (Shi and Malik 2000)')
axis('square')

subplot(ax(3))
imagesc(diag(C3))
cbr = colorbar;
set(cbr, 'YTick', 1:1:4);
title('Normalized spectral clustering (Ng, Jordan, Weiss 2002)')
axis('square')
```




## 2.i. Results

For k, I choose the value 3 as an average of various methods on reporting optimal k value.

Time for variations of spectral clustering:

- Method 1: 0.014705 seconds
- Method 2: 0.011395 seconds
- Method 3: 0.012070 seconds

There was no significant difference between the 3 methods. All have comparable computation times

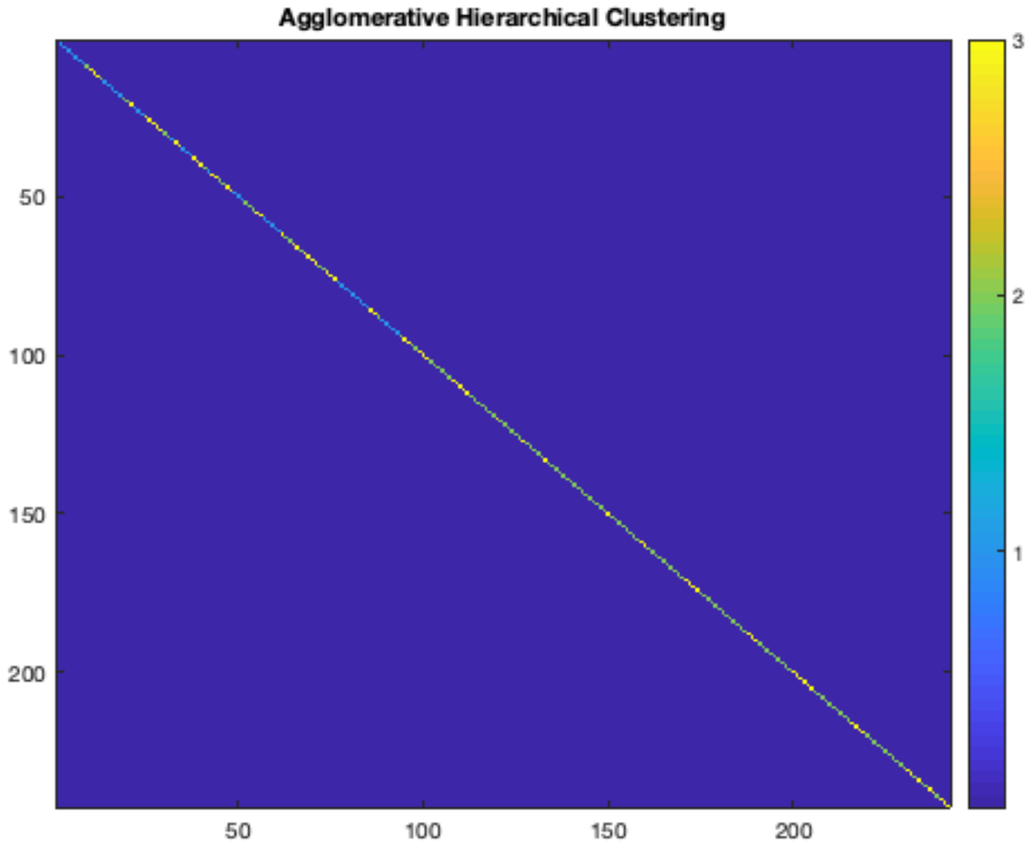
In terms of clustering, as reflected in the images, it seems that the normalized approaches are better able to assign more categories than the unnormalized approach

Both normalized apporaches seem to be identifying various groups well.

## 2.j. Alternative Clustering: Agglomerative Hierarchical Clustering Tree

Create a hierarchical cluster tree using Ward's linkage

```
figure(7)
% Returns matrix Z that encodes a tree containing hierarchical clusters of
% the rows and the input data
Z = linkage(imr90chr2, 'ward');
c = cluster(Z,'Maxclust',3);

imagesc(diag(c))
cbr = colorbar;
set(cbr, 'YTick', 1:1:4);
title('Agglomerative Hierarchical Clustering')
```




*Published with MATLAB® R2019a*