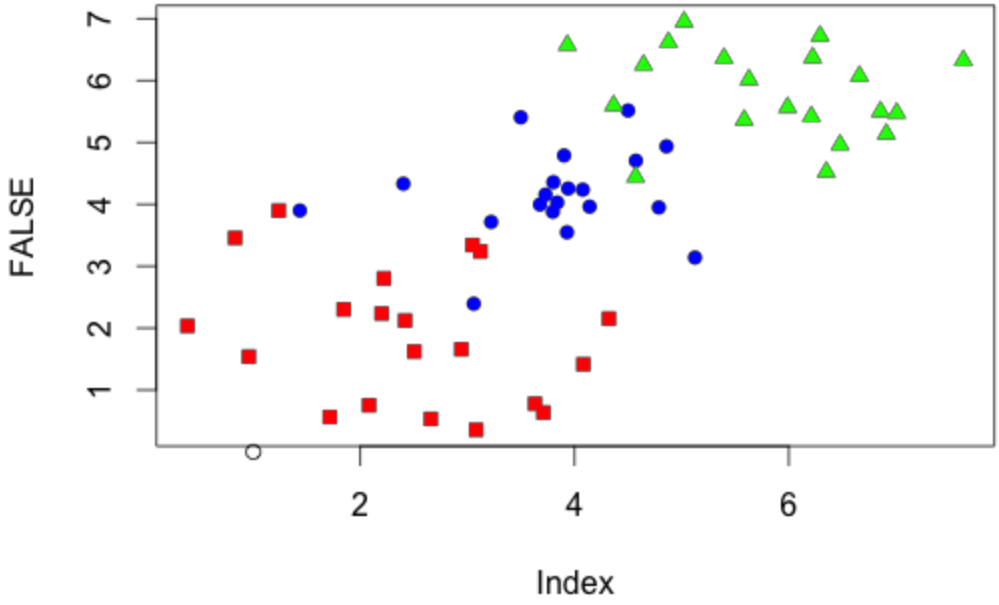
Group : Abdulrahman Alosaimy(amalosai), Siddharth Sharma(ssharm10)
Lecture Assignment: **Community Detection using Spectral Graph Theory**
Instructions:
For each Exercise, submit the script, the output of running this script, and the answers to the questions (if applicable).

Exercise 1: ***Generating the data set***. Write a script (in R, Matlab, or SAS) that generates a total of 60 points whose (*x*,*y*)-coordinates are drawn from a mixture of three Gaussians in a 2-dimentional real space. Each mixture has a mean of 2, 4, and 6, respectively, a standard deviation of one, and about 20 points.

```
set.seed(146)
num = 20
points = list(x1=rnorm(num,2,1),x2=rnorm(num,4,1),x3=rnorm(num,6,1),y1=rnorm(num,2,1),y2=rnorm(num,4,1),
y3=rnorm(num,6,1))
allpoints = list(x=c(points$y1,points$y2,points$y3),y=c(points$x1,points$x2,points$x3))
mat = matrix(c(points$y1,points$y2,points$y3,points$x1,points$x2,points$x3),ncol=2)
```

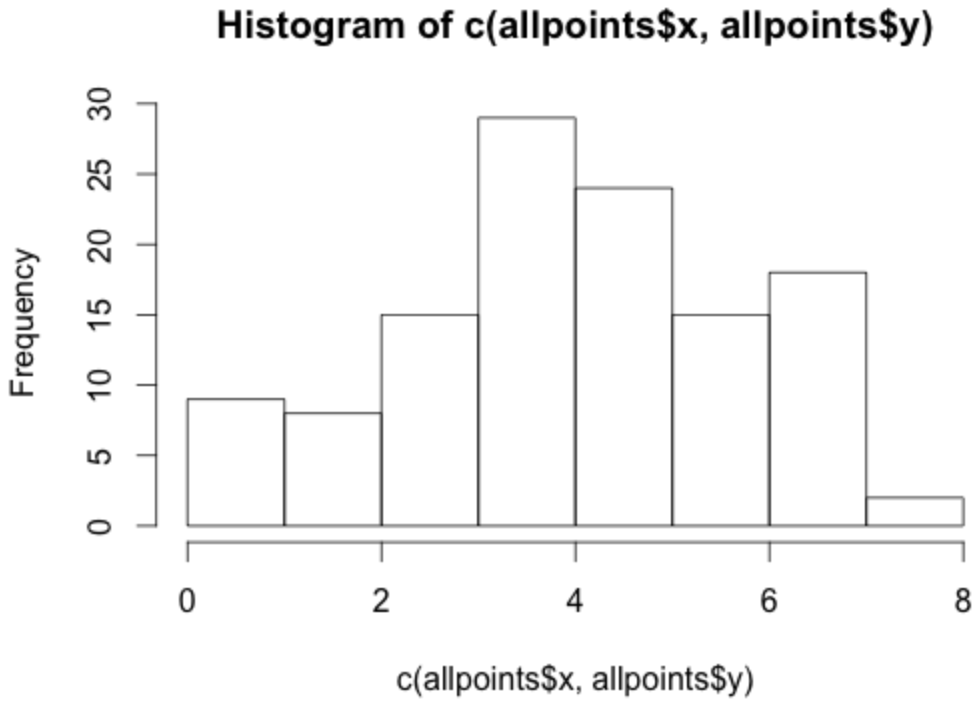
(a) Plot all the points in a single 2-dimensional space by using different shapes for each mixture.

```
plot(-10000,xlim=c(min(allpoints$x),max(allpoints$x)),ylim=c(min(allpoints$y),max(allpoints$y)),
    ylab="y",xlab="x")
points(x=points$y1, y=points$x1, pch=15,col="red")
points(x=points$y2, y=points$x2, pch=16,col="blue")
points(x=points$y3, y=points$x3, pch=17,col="green")
```



(b) Plot a histogram of all the points.

```
hist(c(allpoints$x,allpoints$y))
```

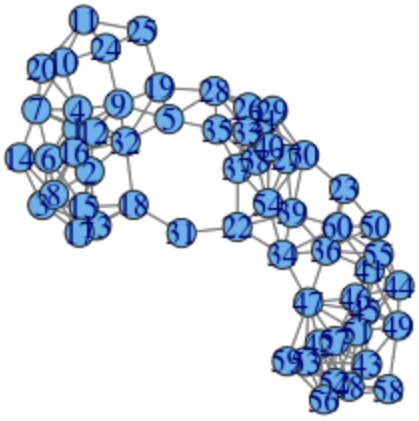


Exercise 2: ***Generating the similarity graphs***. Write a script that generates the following *similarity graphs* for the data set in Exercise 1 (see Lecture Notes):

(a) **KNN**: The *K-nearest neighbor graph* using the value of K=10. Plot the graph.

```
library(igraph)
k = 10
ds = as.matrix(dist(mat))
tmp = matrix(FALSE,nrow=num*3,ncol=num*3)
for (i in 1:(num*3)){
   o1 = head(order(ds[i,]),n=k+1)
   tmp[i,o1] = TRUE
}
#mutual knn only
adj1 = t(tmp) & tmp
for(i in 1:60)
  adj1[i,i] = FALSE

plot(graph.adjacency(adj1,mode="undirected"))
```

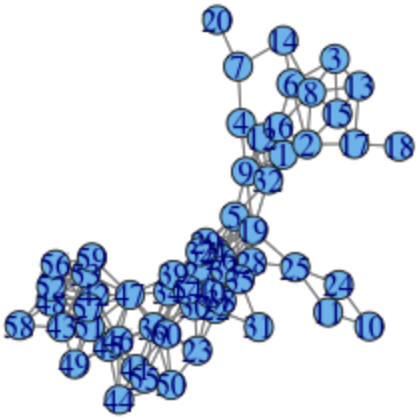


(b) **GK**: The complete similarity graph using the Gaussian kernel with sigma=1 as similarity function.

```
gauss = function(dis,sigma){
```

```
  return (exp(-(dis^2)/(2*sigma^2)))
}

adj2 = gauss(ds,1) > 0.5
for(i in 1:60)
   adj2[i,i] = FALSE
plot(graph.adjacency(adj2,mode="undirected"))
```



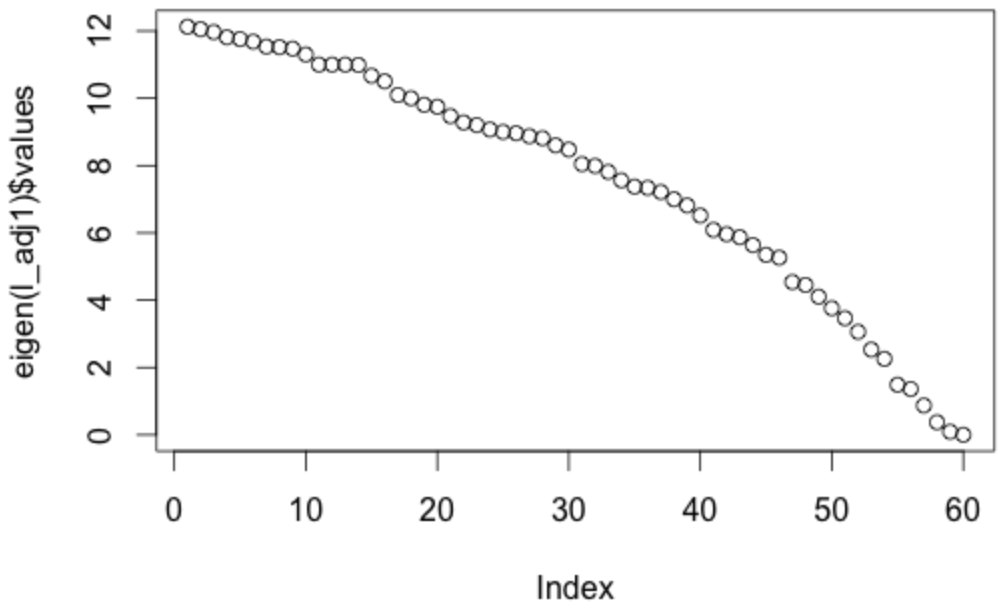
Exercise 3: ***Characterizing the graph spectra***. Write a script that generates the *graph Laplacian matrix L = D - A* and the *normalized graph Laplacian matrix L_hat = I - A_hat* and calculates the graph spectra for each of the graphs in Exercise 2.

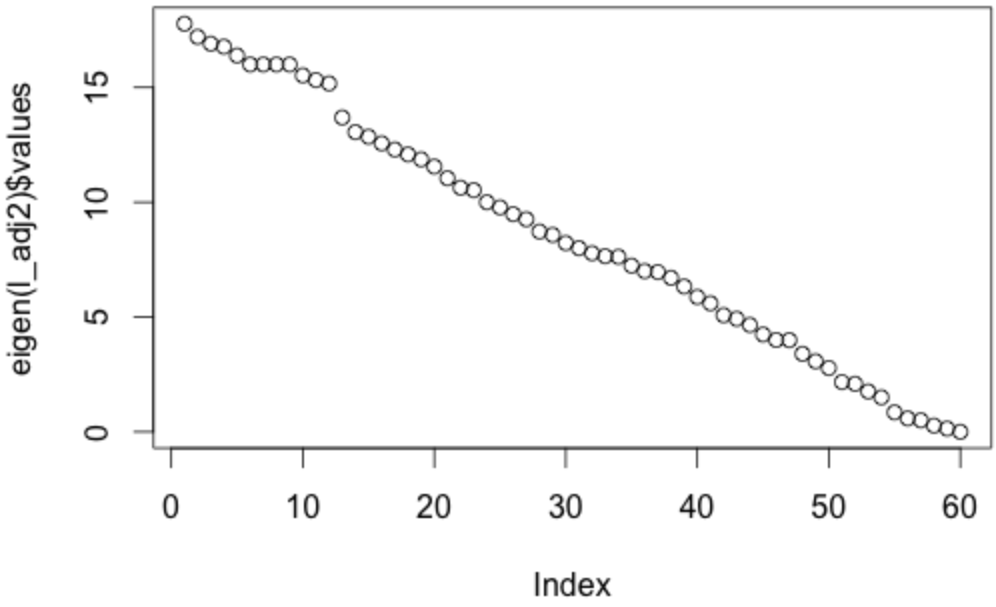
```
deg <- function (mat){
  return (diag(rowSums(mat)))
}
laplacian <- function(mat){
  return (deg(mat) - mat)
}
A_hat <- function(A){
  D = deg_minus_half(A)
  return ( D %*% A %*% D)
}
deg_minus_half <- function(A){
  return( diag(1/sqrt(rowSums(A))) )
}
L_hat <- function(A){
```

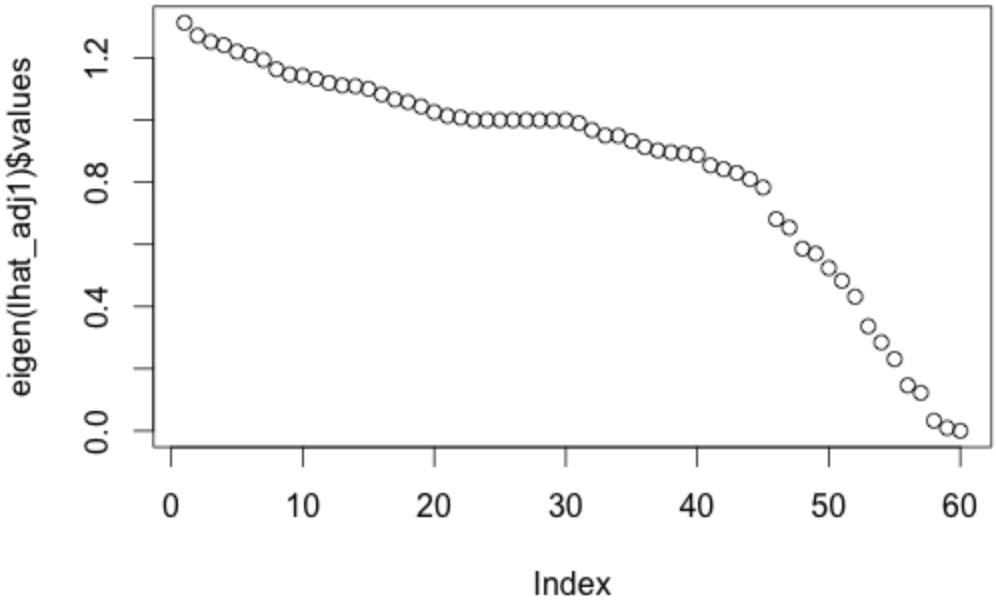
```
  return (diag(nrow(A)) - A_hat(A))
}
l_adj1 = laplacian(adj1)
l_adj2 = laplacian(adj2)
lhat_adj1 = L_hat(adj1)
lhat_adj2 = L_hat(adj2)
```

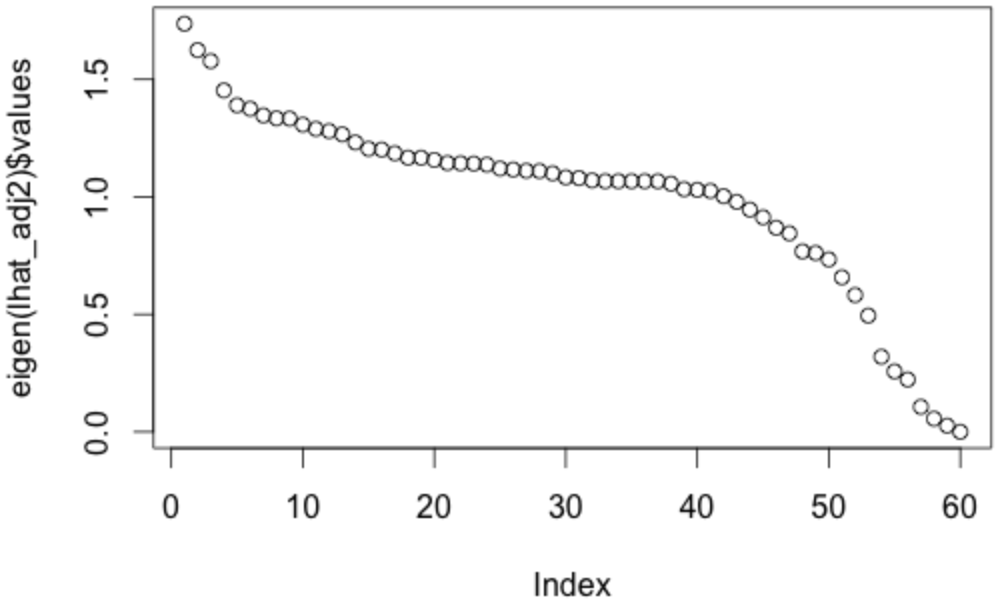
(a) Plot each graph's eigenspectra as a separate figure with *i* as *x*-axis and \lambda_i as *y*-axis (a total of four plots)?

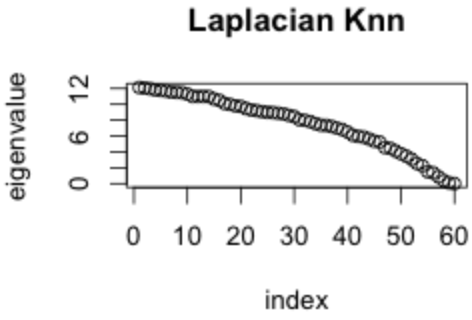
```
plot(eigen(l_adj1)$values, xlab="index",ylab="eigenvalue", main="Laplacian Knn")
plot(eigen(l_adj2)$values, xlab="index",ylab="eigenvalue", main="Laplacian Gaussian")
plot(eigen(lhat_adj1)$values, xlab="index",ylab="eigenvalue", main="Normlized Laplacian Knn")
plot(eigen(lhat_adj2)$values, xlab="index",ylab="eigenvalue", main="Normlized Laplacian Gaussian")
```

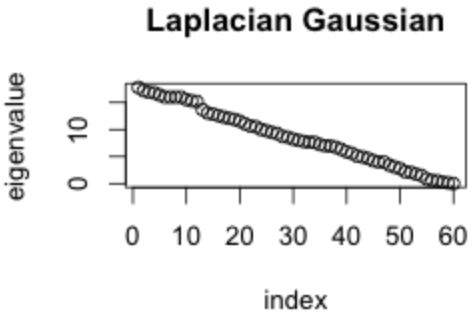


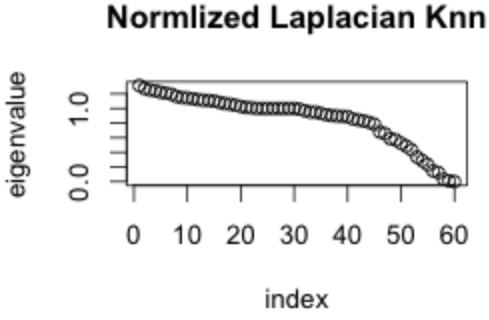

eigen(l_adj2)$values
0
5
10
15
0
10
20
30
40
50
60
Index

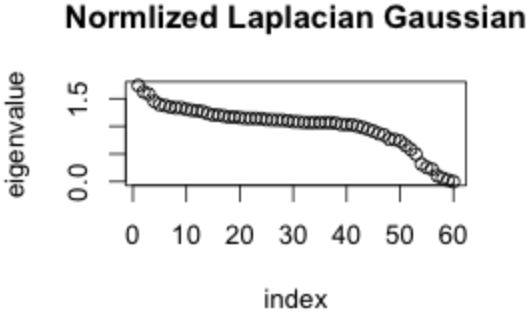

eigen(lhat_adj1)$values
0.0
0.4
0.8
1.2
0
10
20
30
40
50
60
Index











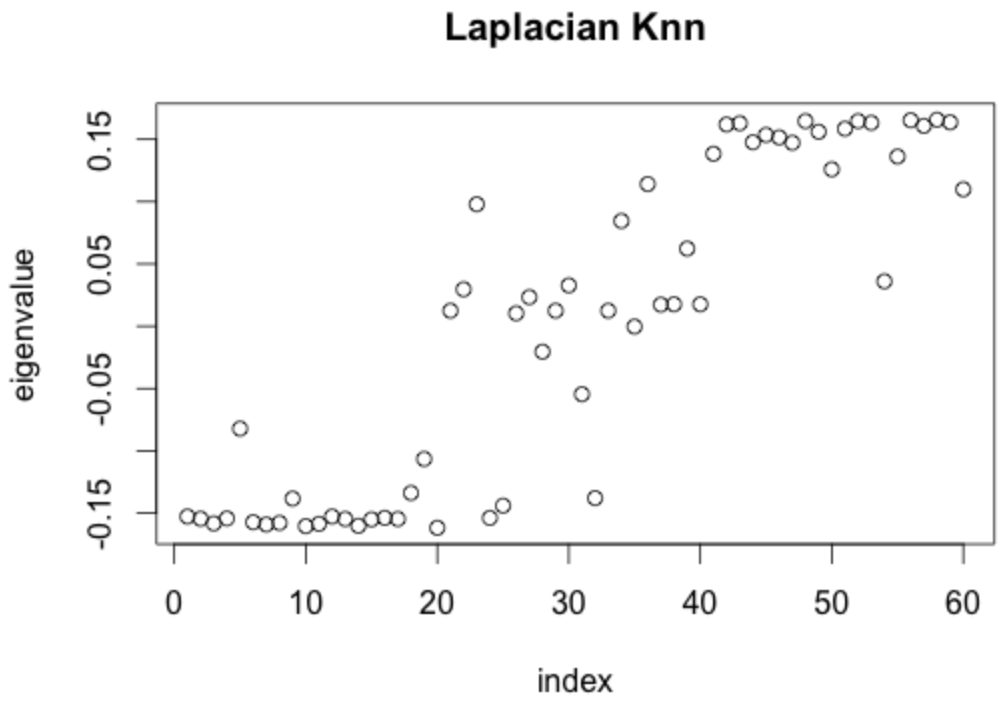
(b) What do you observe about the multiplicity of the "close to" zero eigenvalues? Are your observations consistent with the Properties described in lecture notes?

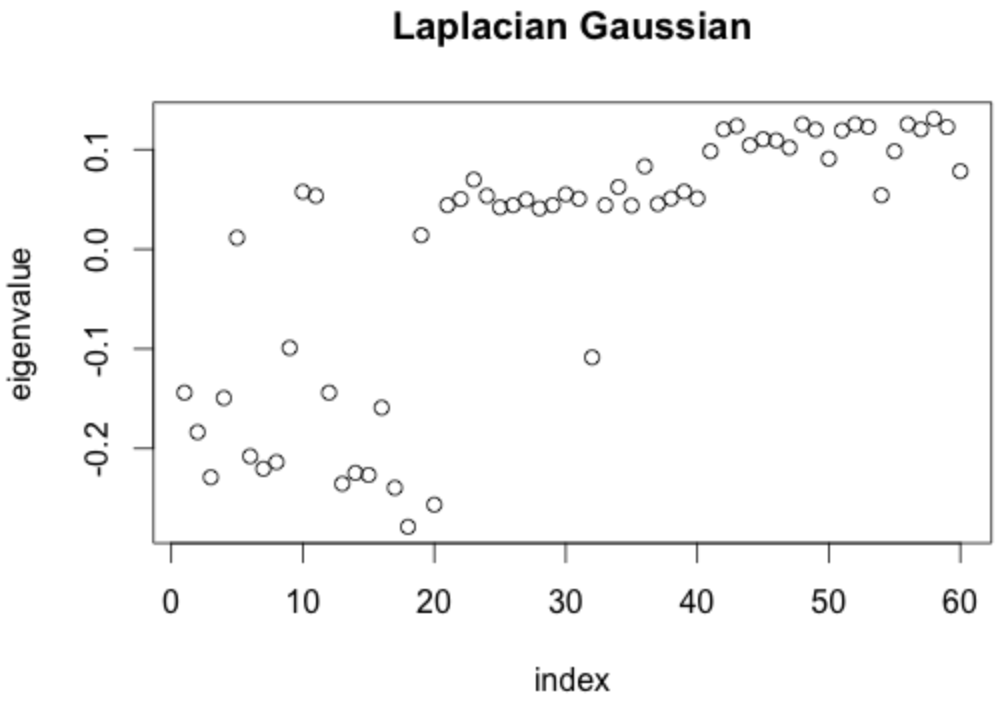
Three eigenvalues are close to zero for laplacian
The eigenvalues for laplcaian for Knn graph
2.852072e-01 7.013601e-02 0
The eigenvalues for laplcaian for gaussian graph
2.686290e-01 1.404686e-01 0
The eigenvalues for normlized laplcaian for Knn graph
3.246284e-02 9.254913e-03 1.332268e-15
The eigenvalues for normalized laplcaian for Knn graph
5.583158e-02 2.609727e-02 1.776357e-15

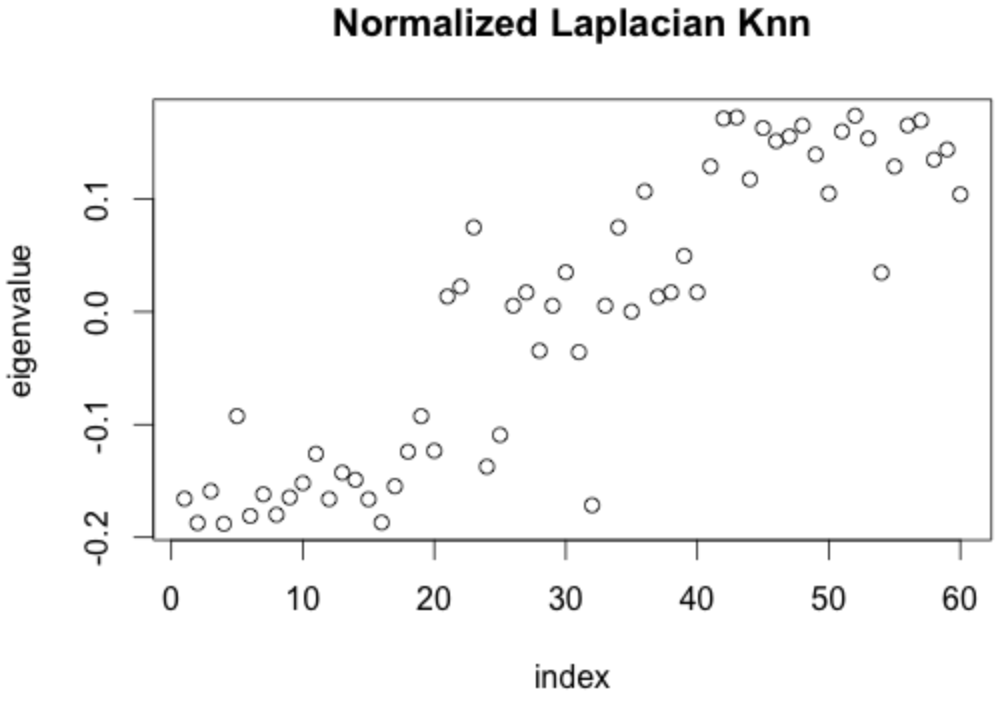
Yes, since the data is gaussian mixture model of three different gaussian distribution, we expected three eigenvalues close to zero.

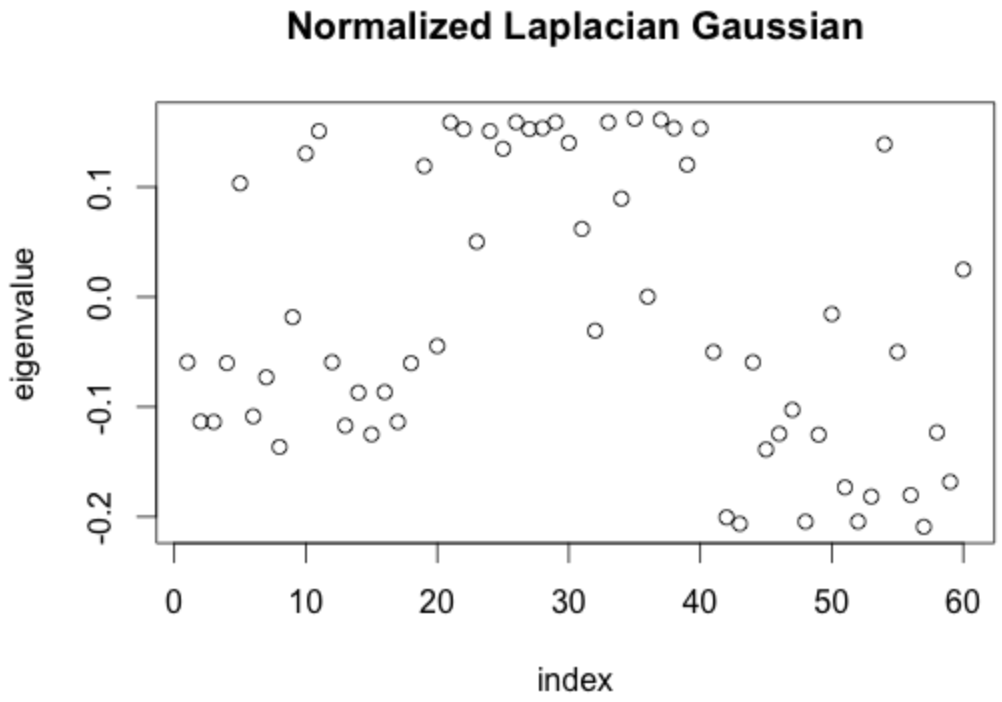
(c) Plot each graph's eigenvector plot for the eigenvector *u* corresponding to the second smallest eigenvalue, with *i* as *x*-axis and *u_i* vector component as *y*-axis.

```
plot(eigen(l_adj1)$vectors[,num*3-1], xlab="index",ylab="eigenvalue", main="Laplacian Knn")
plot(eigen(l_adj2)$vectors[,num*3-1], xlab="index",ylab="eigenvalue", main="Laplacian Gaussian")
plot(eigen(lhat_adj1)$vectors[,num*3-1], xlab="index",ylab="eigenvalue", main="Normalized Laplacian Knn")
plot(eigen(lhat_adj2)$vectors[,num*3-1], xlab="index",ylab="eigenvalue", main="Normalized Laplacian Gaussian")
```




Laplacian Gaussian
eigenvalue
-0.2
-0.1
0.0
0.1
0
10
20
30
40
50
60
index


Normalized Laplacian Knn
eigenvalue
-0.2
-0.1
0.0
0.1
0
10
20
30
40
50
60
index



(d) If you were using this plot for 2-way graph partitioning into S and V-S, the points from which mixtures will end up in which partition?

For KNN Laplacian, Gaussian Laplacian and Normalized Laplacian KNN , i we choose a threshold of zero, we will get points from first mixture (mean = 2, and sd =1) in the partition S, and the points from second and third mixture will end up in partition V-S.

For Gaussian Normalized Laplacian, it is different. If we choose a threshold of zero, we will get points from first mixture (mean = 4, and sd =1) in the partition S, and the points from second and third mixture will end up in partition V-S.

(e) Calculate the conductance (write the script) for each of the identified partitions, S and V-S for the KNN graph using both the normalized and unnormalized Laplacian.

```
conductance = function(mat,sub){
  m_s = sum(mat[sub,sub])
  n_s = length(sub)
  c_s = sum(mat[sub,-sub]) #+ sum(mat[-sub,sub])
  return(c_s/(2*m_s+c_s))
}

conductance(adj1,which(eigen(l_adj1)$vectors[,num*3-1]<0))
conductance(adj1,which(eigen(lhat_adj1)$vectors[,num*3-1]>0.05))
```

Laplacian KNN: 0.03614458
Normalized Laplacian KNN: 0.01775148

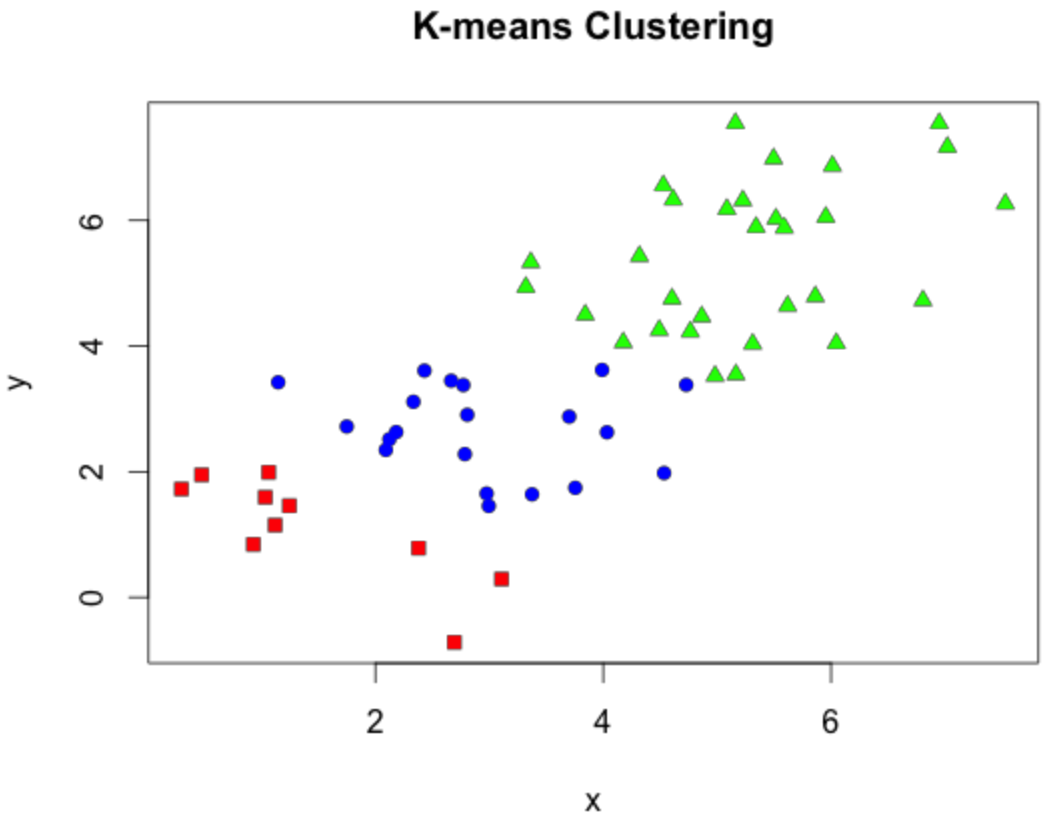
(f) Calculate the lower and upper bounds for the graph conductance using the inequalities provided in the lecture notes. How does this value compare with the conductance obtained for S or V-S in 3.e?

$\lambda_2/2$ < conductance < sqrt(2*λ_2)
lower bound is 0.004627456.
upper bound is 0.1360508
0.004627456 < 01775148 <0.1360508

Exercise 4: ***Spectral graph clustering***. Write a script that performs spectral graph clustering using the normalized graph Laplacian of each of the graph in Exercise 2. The pseudo-code of the clustering method is described in the lecture notes. For the k-means clustering method use the value of k=3.

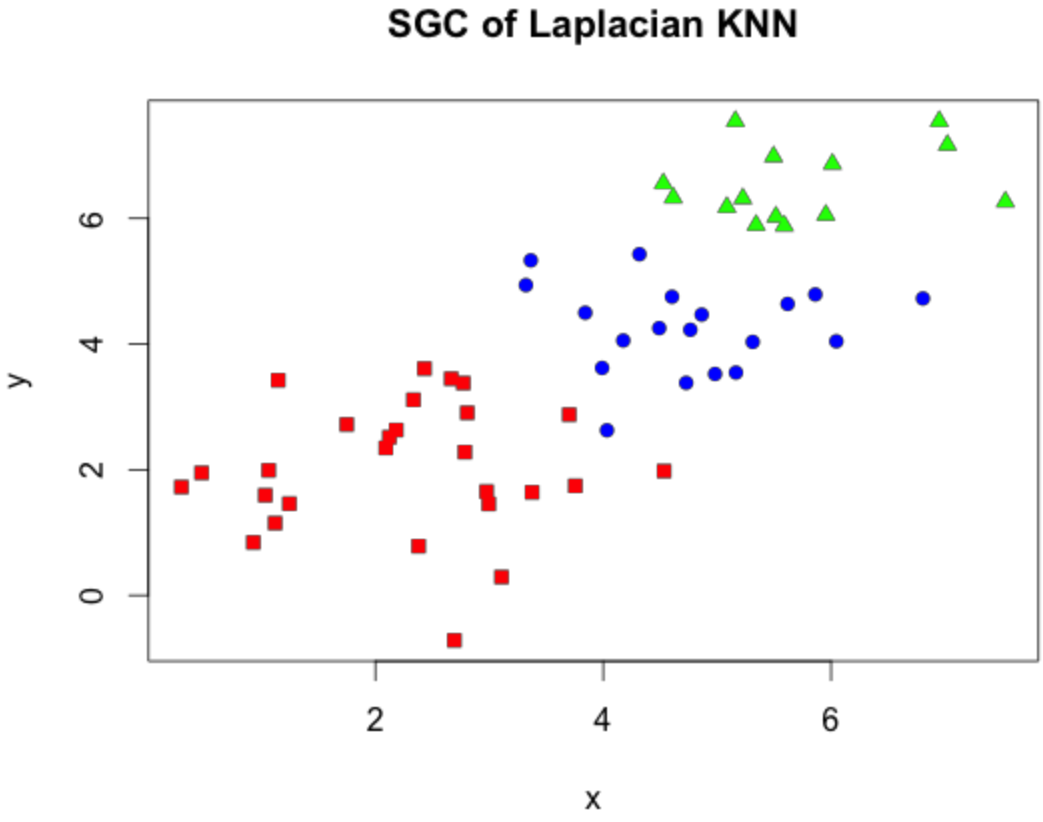
(a) Run the k-means clustering algorithm provided by R/Matlab/SAS on the data set in Exercise 1, using the Euclidean distance as the dissimilarity metric, and the value of k=3. Plot the points in 2-dimensional space but use different shape for each of the identified cluster.

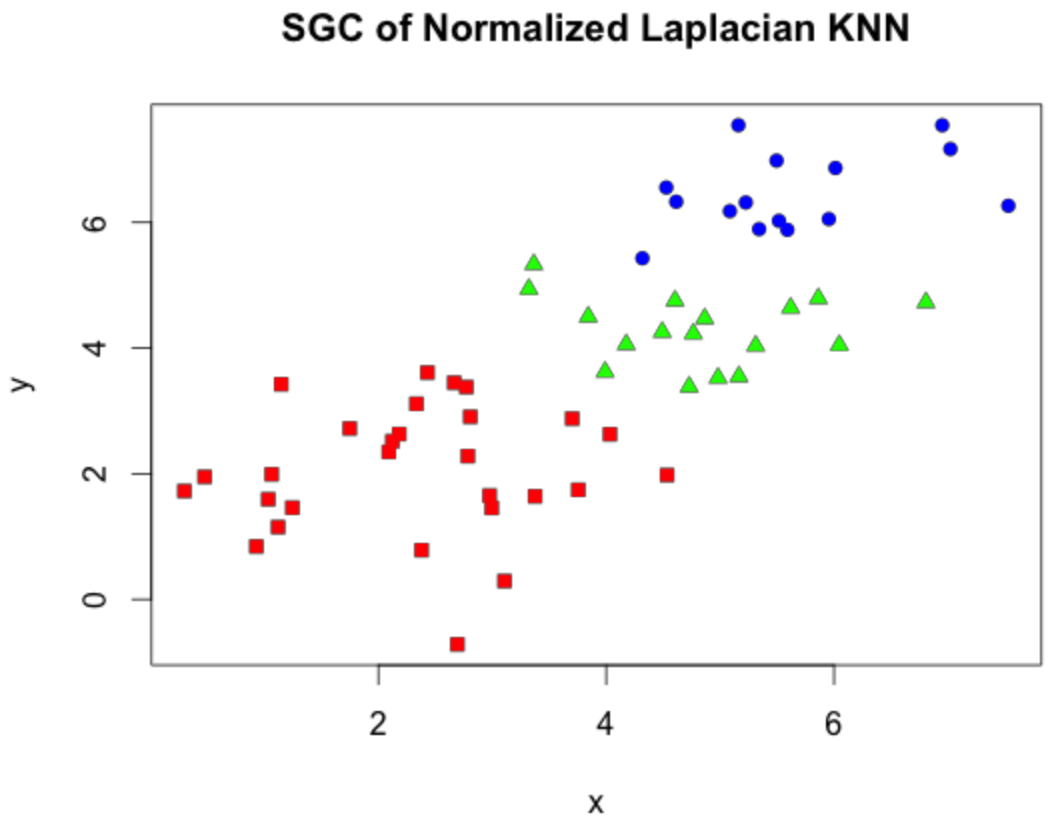
```
k =3
kmeans_clustering = kmeans(mat,centers=k)
plot(-10000,xlim=c(min(allpoints$x),max(allpoints$x)),ylim=c(min(allpoints$y),max(allpoints$y))
    ,xlab="x",ylab="y")
points(mat[which(kmeans_clustering$cluster==1),], pch=15,col="red")
points(mat[which(kmeans_clustering$cluster==2),], pch=16,col="blue")
points(mat[which(kmeans_clustering$cluster==3),], pch=17,col="green")
title("K-means Clustering")
```

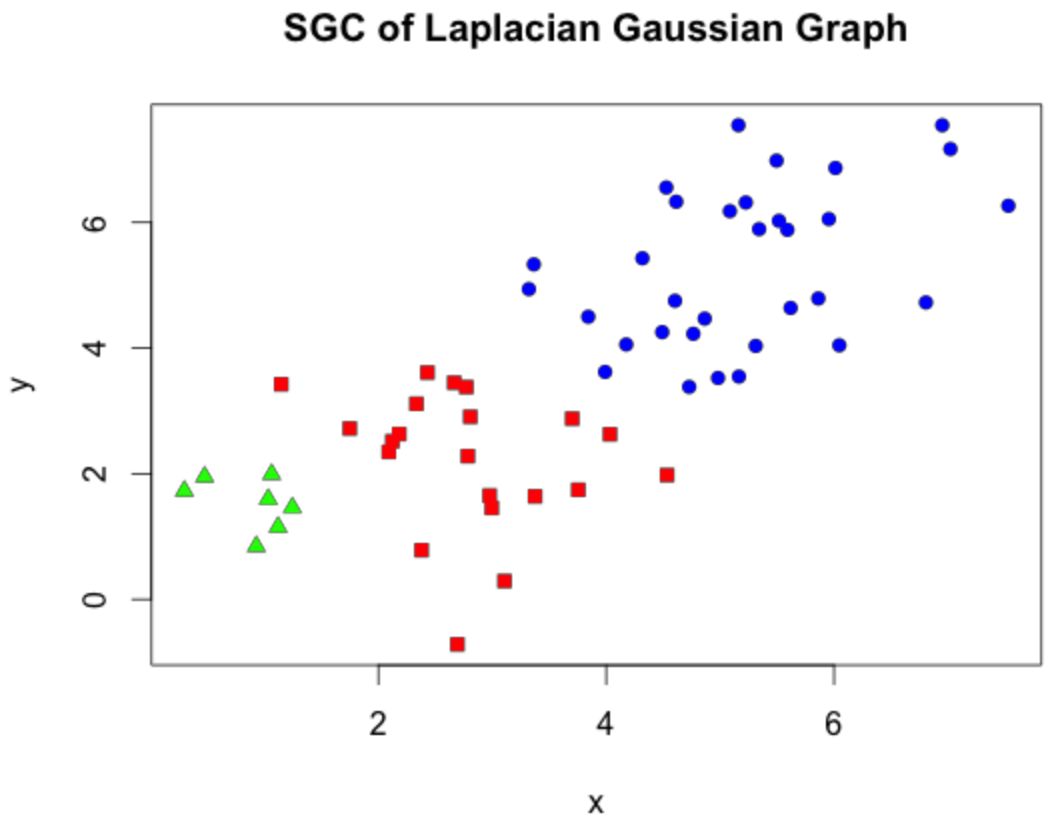


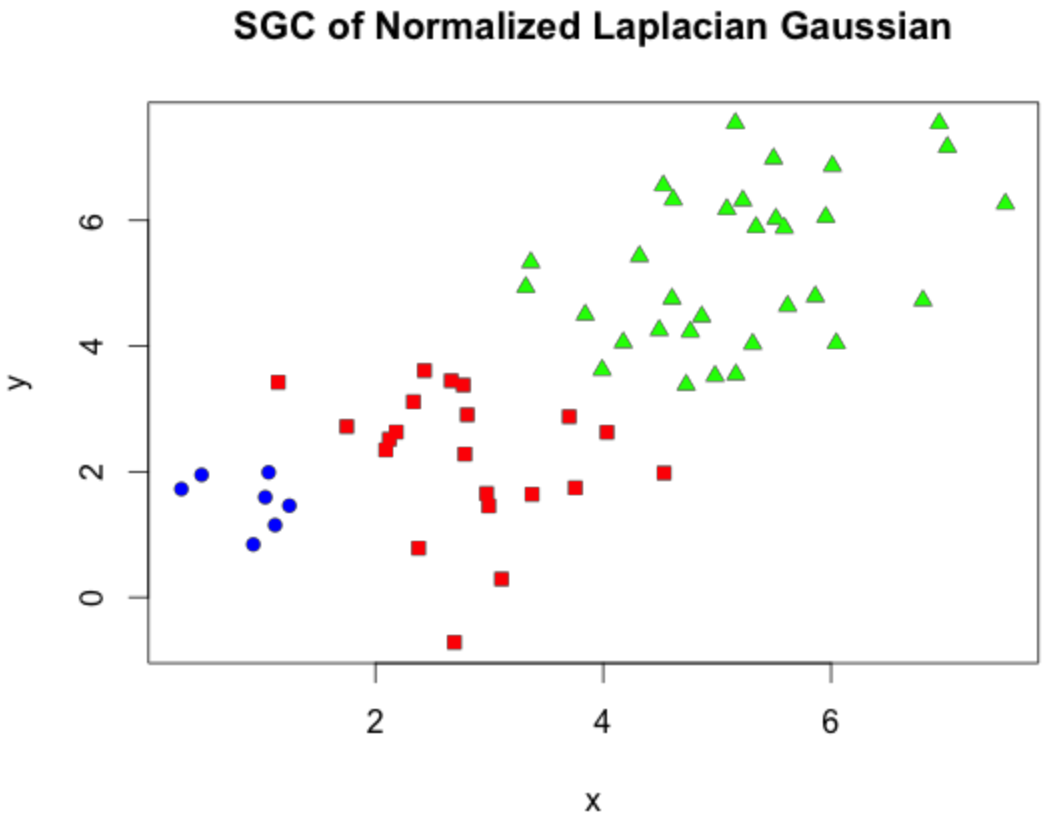
(b) Run the spectral graph clustering and plot the corresponding points in Ex.1 with the shapes based on the identified cluster (one plot for each graph). Are there mismatches between the results obtained from spectral graph clustering and the kmeans clustering in 4.a?

```
spectral_graph_clustering <- function(mat2,k){
  eig = eigen(mat2)
  y = eig$vectors[,(num*3-k+1):(num*3)]
  km = kmeans(y,centers=k)
  plot(-10000,xlim=c(min(allpoints$x),max(allpoints$x)),ylim=c(min(allpoints$y),max(allpoints$y)),
       xlab="x",ylab="y")
  points(mat[which(km$cluster==1),], pch=15,col="red")
  points(mat[which(km$cluster==2),], pch=16,col="blue")
  points(mat[which(km$cluster==3),], pch=17,col="green")
  title("Spectral Graph Clustering")
  return(km)
}
sgc_l1 = spectral_graph_clustering(l_adj1,3)
sgc_lhat1 = spectral_graph_clustering(lhat_adj1,3)
sgc_l2 = spectral_graph_clustering(l_adj2,3)
sgc_lhat2 = spectral_graph_clustering(lhat_adj2,3)
```


SGC of Laplacian KNN
y
x
0
2
4
6
2
4
6


SGC of Normalized Laplacian KNN
y
x
0
2
4
6
2
4
6


SGC of Laplacian Gaussian Graph
y
x
0
2
4
6
2
4
6



<u>Exercise 5</u>: ***Performance evaluation***. Compute the performance metrics for the community/cluster detection methods assuming the ground-truth clustering corresponds to the Gaussian mixture the point has been generated from.

(a) Write the script that calculates the following metric for each community: Separability, Density, Cohesiveness, and Clustering Coefficient (see lecture notes for definitions and the required paper reading, *Defining and Evaluating Network Communities based on Ground-truth by J. Yang, J. Leskovec. IEEE International Conference On Data Mining (ICDM), 2012.*

```
density_metric = function(mat,sub){
 m_s = sum(mat[sub,sub])
 n_s = length(sub)
 return(m_s/((n_s*(n_s - 1))/2))
}
separability_metric = function(mat,sub){
 m_s = sum(mat[sub,sub])
 c_s = sum(mat[sub,-sub]) #+ sum(mat[-sub,sub])
 return(m_s/c_s)
}
cohesiveness_metric = function(mat,sub){
 #for all possible subsets:
```

```
 i = 1
 indexes = 1:ncol(mat)
 subset = indexes[-sub]
 max = 0
 for(i in 1:(length(sub))){
    m = combn(subset, i)
    for(j in 1:ncol(m)){
      c = conductance(mat,m[,j])
      if (c > max)
        max = c
    }
 }
 return(max)
}

clustering_coefficient_metric = function(mat,sub){
 cv <- function(x){
   neighbors = which(mat[x,]==TRUE)
   n_v = length(neighbors) * (length(neighbors) -1) /2
   m_v = sum(mat[neighbors,neighbors])/2
   if (n_v == 0) # the case of only one friend
     return (0)
   return(m_v / n_v)
 }
 return(sum(as.numeric(lapply(sub,cv)))/length(sub))
}
```

(c) Compute these community metrics for each of the 3 communities identified by the spectral graph clustering method and the graphs in Ex. 2. Do you observe the differences in the metrics for different types of graphs (KNN vs. GK)?

	Density	Separability	Clustering Coefficient
C1 : Lap KNN	0.5698006	33.33333	0.6767784
C2: Lap KNN	0.7953216	9.714286	0.6192565
C3: Lap KNN	1.120879	12.75	0.709127
C1: Norm Lap KNN	0.5502646	52	0.6716553
C2: Norm Lap KNN	1.047619	13.75	0.6856614
C3: Norm Lap KNN	0.9117647	10.33333	0.6397292
C1: Lap Gaussian	0.6190476	21.66667	0.618722
C2: Lap Gaussian	0.4717742	58.5	0.6838993

C3: Lap Gaussian	1.904762	20	0.8955782
C1: Norm Lap Gaussian	0.6190476	21.66667	0.618722
C2: Norm Lap Gaussian	1.904762	20	0.8955782
C3: Norm Lap Gaussian	0.4717742	58.5	0.6838993

(d) Write the script that calculates the following metrics for the performance of the community detection algorithm against the ground-truth: Precision, Recall, and F1-measure.

```
confusion_matrix = function(original,predicted){
  mat = matrix(0,nrow=2,ncol=2)
  mat[1,1] = sum(original & predicted) #both true: true positive
  mat[1,2] = sum(!original & predicted) #original is true, predicted is false: false positive
  mat[2,1] = sum(original & !predicted) #original is true, predicted is false: true negative
  mat[2,2] = sum(!original & !predicted) #original is true, predicted is false: true negative
  return(mat)
}
precision = function(original,predicted){
  mat = confusion_matrix(original,predicted)
  return (mat[1,1] / (mat[1,1]+mat[1,2]))
}
recall = function(original,predicted){
  mat = confusion_matrix(original,predicted)
  return (mat[1,1] / (mat[1,1]+mat[2,1]))
}
f1 = function(original,predicted){
  p = precision(original,predicted)
  r = recall(original,predicted)
  if (p+r == 0)
    return(0)
  return ((2*p*r) / (p+r))
}
```

(e) Compare the performance of kmeans clustering method with the spectral graph clustering method in terms of the metrics in 5.d.

	Precision	Recall	F1-Measure
Cluster 1			
Kmean1		0.5	0.6666667
L KNN	0.05263158	0.05	0.05128205
NL KNN	0.7142857	1	0.8333333
L GAUS	1	0.35	0.5185185
NL GAUS	1	0.35	0.5185185

Cluster 2

Kmean	0.3333333	0.4761905	0.3921569
L KNN	0.3333333	0.4285714	0.375
NL KNN	0.7058824	0.5714286	0.6315789
L GAUS	0.4285714	0.4285714	0.4285714
NL GAUS	0.4285714	0.4285714	0.4285714

Cluster 3

Kmean	0.05	0.04761905	0.04878049
L KNN	1	0.6666667	0.8
NL KNN	1	0.7142857	0.8333333
L GAUS	0.625	0.952381	0.754717
NL GAUS	0.625	0.952381	0.754717