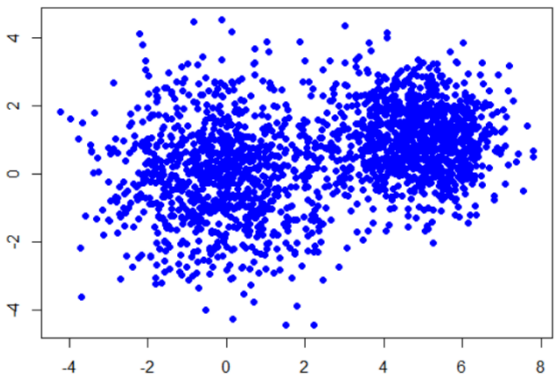
IE 529

Computational Assignment 2

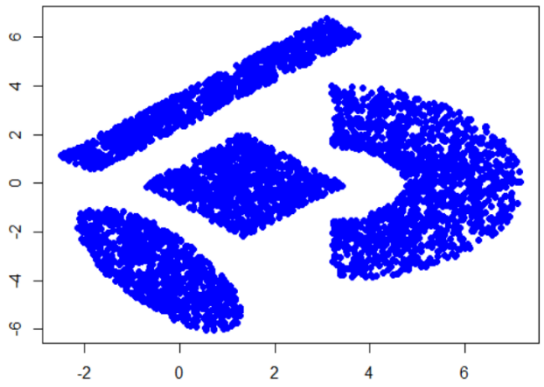
Shulu Chen(shuluc2)

2019/12/18

1. **Include a scatter plot for each set of data, as is, i.e., with no clusters identified yet.**



Dataset1: Clustering



Dataset2: ShapedData

2. **Submit your pseudocodes for the Lloyds/K-means algorithm, the Greedy K centers and Single Swap algorithms, and the Spectral Clustering algorithm, each on a separate sheet of paper. State clearly whether you are considering normalized or unnormalized Spectral Clustering.**

Lloyd's algorithm

1. Initialization – arbitrarily select k points as initialize centroids m_j
2. Computer distances (L-2 norm) between each sample x_i and each centroid m_j to form a distance matrix D.
3. Assign each x_i to its nearest m_j (partitions data into k clusters).
4. Update: recompute new "means" for each of the clusters

$$m_j = \frac{1}{|c_j|}\sum_{i=1}^{n} x_i p_{ij}$$

5. Repeat: until sum of the move-distance between "old centroids" and "new centroids" convergence.

Greedy k-centers algorithm

1. Initialization: set $C = \{c_1\}$ for arbitrary $c_1 \in X$
2. While $|c| < k$, do:
   a) Find $x_l \in X/C$ that maximize the minimum distance between x_l and C

$$\max_{x_l}\{\min_{i}(\|x_l - c_i\|_2^2)\}$$

   b) Set $C = C \cup \{x_l\}$.

   c) End while.

3. Assign each $\boldsymbol{x_i}$ to its nearest $\boldsymbol{c_i}$ (partitions data into k clusters)**.**

Single-swap heuristic algorithm

1. Initialization: select set of initial medians $C^0 = \{m_1^0, m_2^0, \dots, m_k^0\}$ and compute the clusters based on C^0.
2. Compute the cost value:

$$cost^0 = \max_{x_i \in X} (\min_{c_j \in C} \|x_i - c_j\|_2^2)$$

3. Set iteration times n.
4. While iteration time $l < n$, do:

   a) Arbitrarily select a point $x_l \in X/C$ and a centroid $m_j^l \in C$, swap them and get a temporary centroid set $C^{l\prime}$.

   b) Compute the new cost value $cost^{l\prime}$ based on $C^{l\prime}$.

   c) IF $\frac{cost^{l\prime}}{cost^l} \leq 0.95$:

      Update the centroid set: $C^l \leftarrow C^{l\prime}$

   d) End while.

Spectral Clustering algorithm(unnormalized)

1. Compute weight matrix:

$$s_{ij} = \exp\left(-\frac{\|x_i - x_j\|_2^2}{\sigma}\right)$$

$$W = \{s_{ij}\}$$

2. Compute degree matrix:

$$D = diag\{\sum_j s_{ij}\}$$

3. Compute Laplacian matrix:

$$L = D - W$$

4. Compute eigenvalue/eigenvector decomposition for L:

$$L = U^{-1}\Lambda U$$

5. Choose least k eigenvalues and take corresponding eigenvectors of U as u_k.

6. Denote rows of u_k by y_i

7. Cluster "points" $\{y_1, \dots, y_n\}$ using k-means.

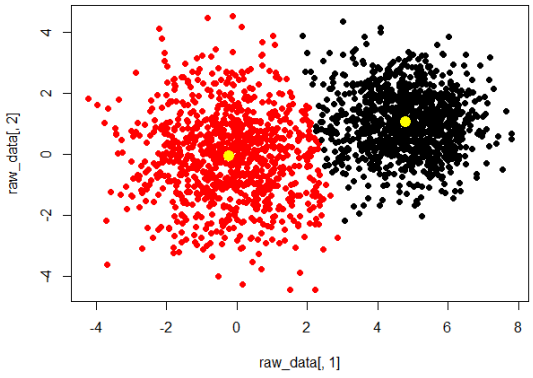
8. Define output index sets $J_l = \{j | y_j \in c_l\}$

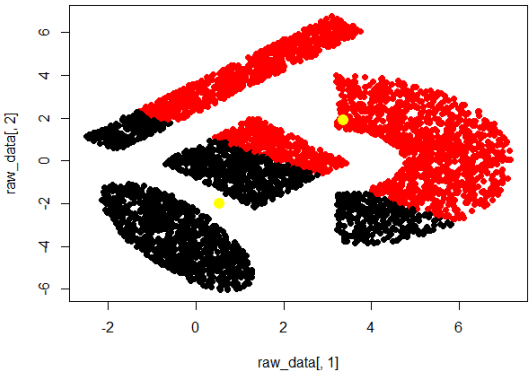
3. **For your K-means algorithm by itself, and for your Spectral Clustering algorithm, present the output(clustering) results for BOTH sets of test data given to you. (See below)**

4. **Evaluate the results for a range of K values (number of clusters), e.g., from K = 2, to K = 11 (but maybe not all of these; try at least 3 different values, then try to pick 2-3 more that will help you find the best cluster outcome). When implementing the K-means algorithm (by itself and from your Spectral Clustering algorithm), you should try multiple initializations (say 5 to 10) and then select the 'best' result, for each K value. State how many different initializations you tried and why.**

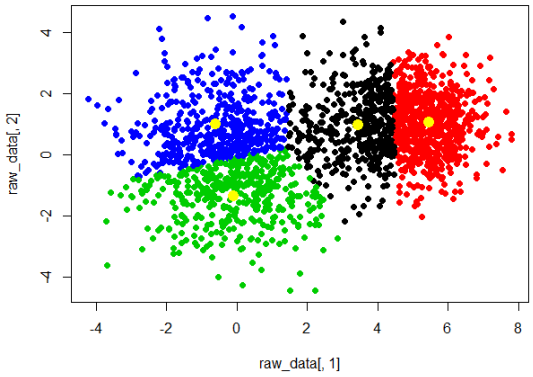
K-means on dataset1 k-means on dataset2

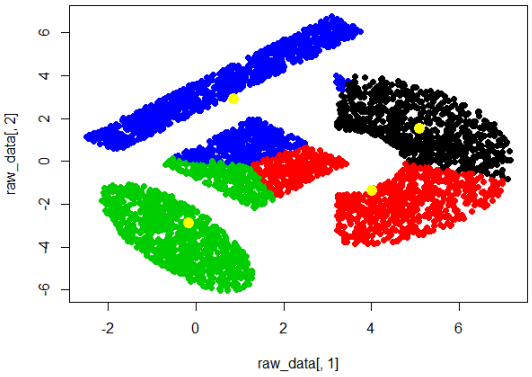
K=2



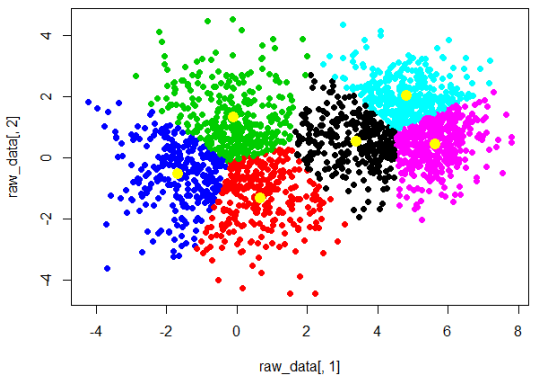


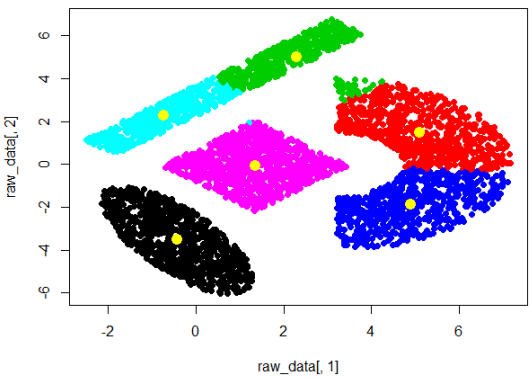
K=4





K=6

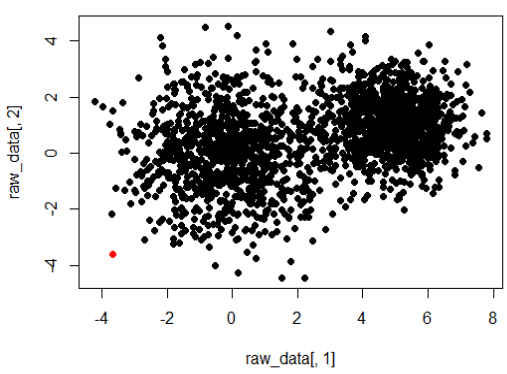


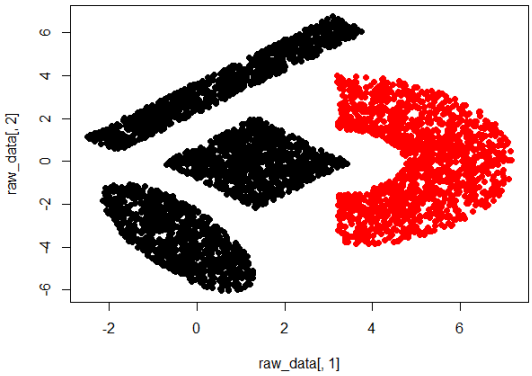


Spectral Clustering on dataset1

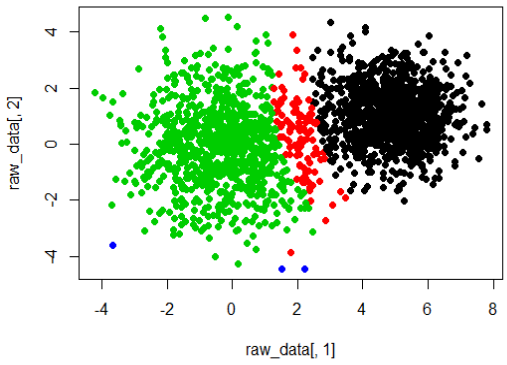
Spectral Clustering on dataset2

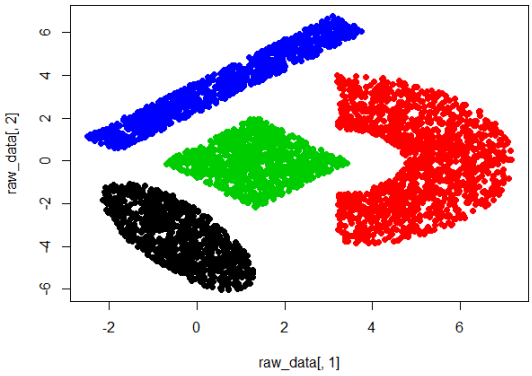
K=2



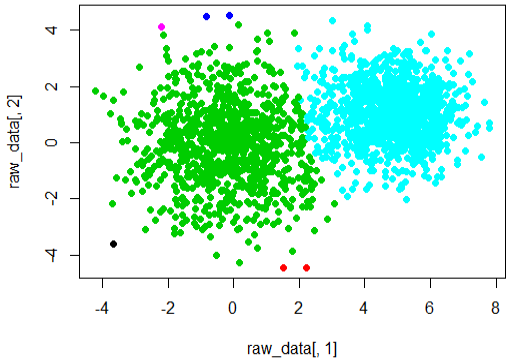


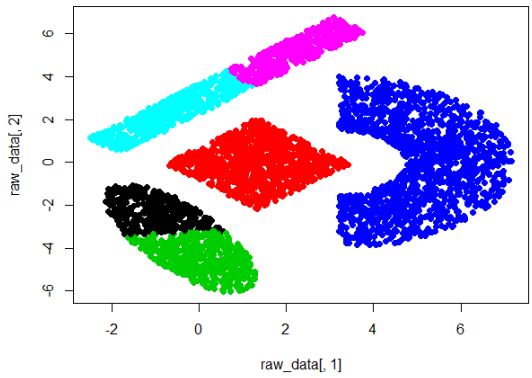
K=4





K=6

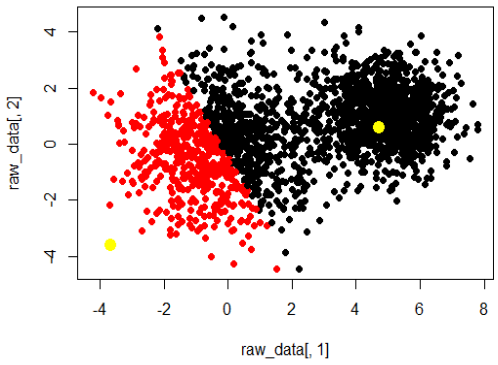


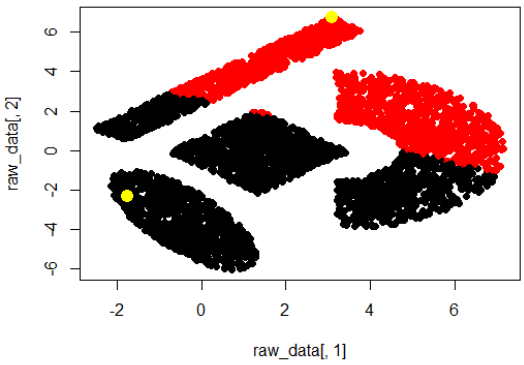


Greedy K-Centers on dataset1

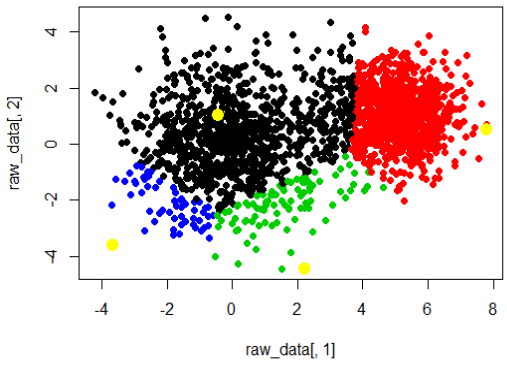
Greedy K-Centers on dataset2

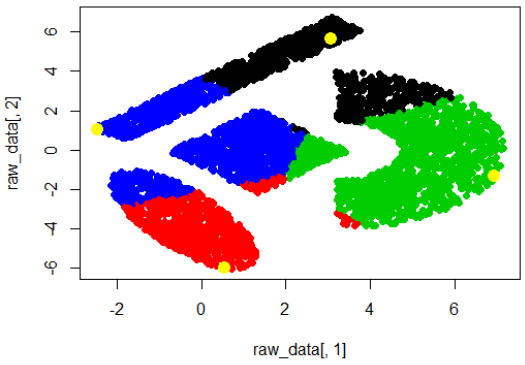
K=2



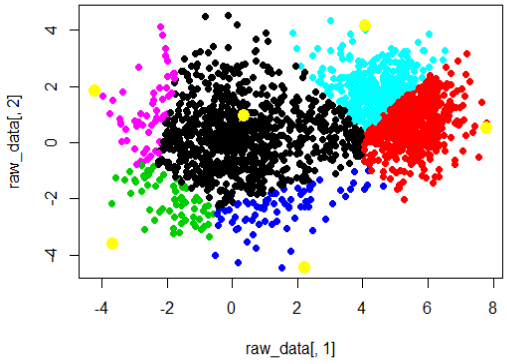


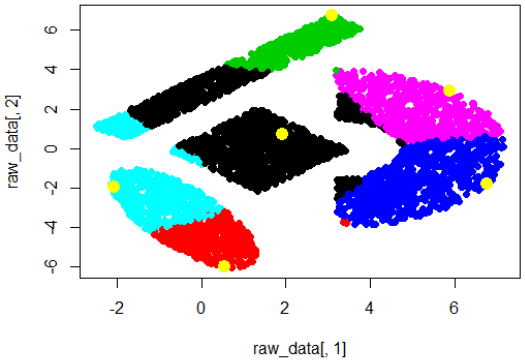
K=4





K=6

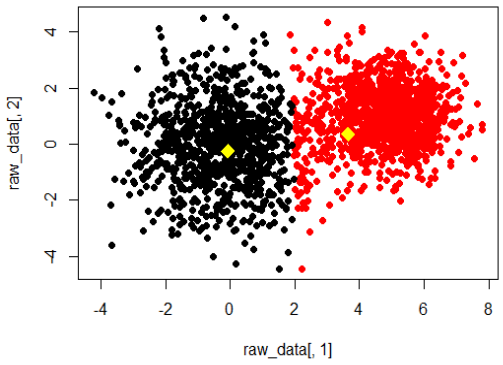


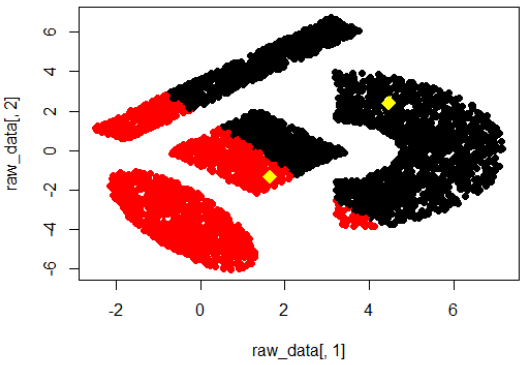


Single-Swap Heuristic on dataset1

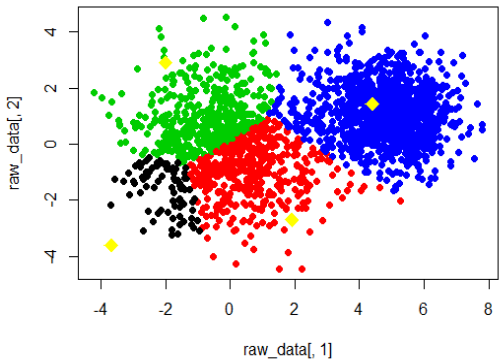
Single-Swap Heuristic on dataset2

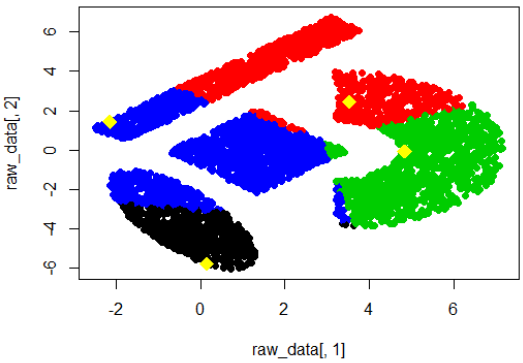
K=2



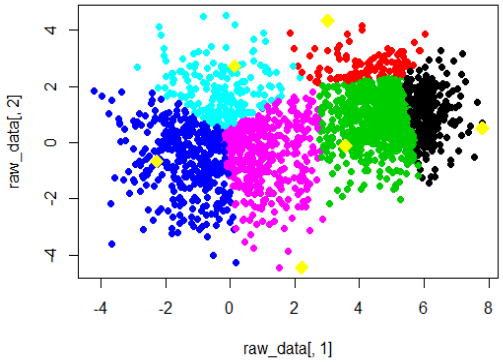


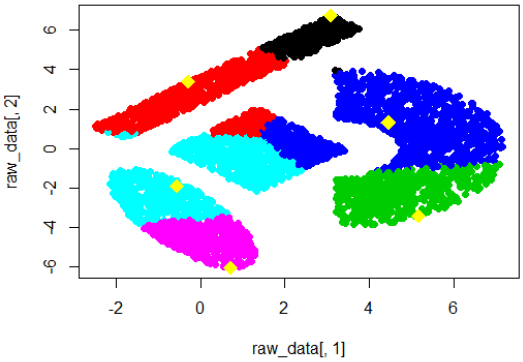
K=4





K=6





5. **For the K-means algorithm, describe how you tested for convergence and why you choose this as a test.**

   For k-means algorithm, I computer the distance between the "old centroids" and "new centroids" as the iteration gains. When the iteration gains is less than 0.0001, which shows that the centroids do not change greatly anymore, and then we can regard it as convergence.

6. **Discuss how many "natural" clusters you think each data set has, and why.**

   Dataset1: Two clusters

   Dataset2: Four clusters

   Visually, sample data in dataset1 mixed together, and the left side and the right side seems cluster at each center, so two clusters make sense. For dataset2, it is clearly to see that there are four shaped sample data, so four clusters would work better.

7. **Provide a short analysis of the computational effort you found was required for each of the algorithms. State what measure you used to evaluate computational effort (e.g., running time or flops, or ??), and how you observed this. Briefly discuss how these computational efforts compare to your expectations.**

   For Lloyd's algorithm, time complexity is $O(nkd)$ at each iteration.
   For Spectral Clustering, time complexity $O(n^3)$ because it needs to computer the similarity matrix and eigenvalue decomposition. When it comes to a large dataset, it would be time-consuming.

8. **Write a brief summary paragraph of your findings, that is, briefly summarize your KEY findings.**

   For Lloyd's algorithm, I find that the initialize points is not so important compared with Greedy K-center algorithm. The result is similar with the package "kmeans" in R language. So, we can make a conclusion that the Lloyd's algorithm is robust, and the accuracy is good.

   For Greedy K-Centers algorithm, the result would be greatly influenced by the initialized points, the results changes greatly at each iteration. Also, I do not get a good result(compared with Lloyd's algorithm).

   The Singular Swap algorithm is a good way to update the centroids after Greedy K-Centers. However, when the clusters k is small, e.g. k=2, the update policy may not work well.

   The Spectral Clustering works really well when the dataset is divided clearly with different shapes, but the result is not well when the dataset is traditional. So, Spectral Clustering may be used in some special area.

Appendix: Personal codes

Lloyd's algorithm

```r
###Lloyd's algorithm###
setwd("C:/Users/Lenovo/Desktop")
raw_data<-read.csv("clustering.csv",header = F)
k=4
n=nrow(raw_data)
plot(x=raw_data[,1],y=raw_data[,2],pch=16,col="blue")

##cluster algorithm
clusters<-function(ct,data){
  distance<-matrix(data = NA,nrow=n,ncol=k)
  cluster<-c()
  for (j in 1:k){
    distance[,j]<-(data[,1]-ct[j,1])^2+(data[,2]-ct[j,2])^2
  }
  for (i in 1:n){
    cluster[i]<-which.min(distance[i,])
  }
  cluster
}

count_cent<-function(cluster_data){
  count_table<-table(cluster_data[,3])
  x<-c()
  y<-c()
  for (i in 1:k){
    x[i]<-(1/count_table[i])*sum(data[data[,3]==i,1])
    y[i]<-(1/count_table[i])*sum(data[data[,3]==i,2])
  }
  cent_point<-cbind(x,y)
  cent_point
}
#initialization:select k points randomly
initial_point<-as.matrix(raw_data[sample(row.names(raw_data), k),])
points(initial_point,col="red",pch=16)
#points(initial_point,col="red",pch=16)
cluster_int<-clusters(initial_point,raw_data)
data<-cbind(raw_data,cluster_int)
cent_points<-count_cent(data)
iteration_distance<-(cent_points-initial_point)
iteartion_rate<-sum(iteration_distance[,1]^2+iteration_distance[,2]^2)
record_iteation<-c()
while (iteartion_rate>=0.0001) {
  cluster<-clusters(cent_points,raw_data)
  data<-cbind(raw_data,cluster)
  cent_points_new<-count_cent(data)
  iteration_distance<-(cent_points_new-cent_points)
  iteartion_rate<-sum(iteration_distance[,1]^2+iteration_distance[,2]^2)
  print(iteartion_rate)
  cent_points<-cent_points_new
  append(record_iteation, iteartion_rate)
}

plot(x=raw_data[,1],y=raw_data[,2],col=data[,3],pch=16)
points(cent_points,col="blue",pch=18,cex=1.5)

#km<-kmeans(raw_data,k)
#plot(x=raw_data[,1],y=raw_data[,2],col=km$cluster,pch=16)
#sum(data[,3]-km$cluster)
```

Greedy k-centers algorithm

```
###GreedyKCenters###
setwd("C:/Users/Lenovo/Desktop")
raw_data<-read.csv("clustering.csv",header = F)
k=6
n=nrow(raw_data)
plot(x=raw_data[,1],y=raw_data[,2],col="blue",pch=16)
#initialization:select a point randomly
initial_point<-as.matrix(raw_data[sample(row.names(raw_data), 1),])
distance_matrix<-matrix(data=NA,nrow=n,ncol=k)
point_matrix<-matrix(data=NA,nrow=k,ncol=2)
point_matrix[1,]<-initial_point
points(x=point_matrix[1,1],y=point_matrix[1,2],col="red",pch=16,cex=2)
i=1
distance_matrix[,i]<-(raw_data[,1]-point_matrix[1,1])^2+(raw_data[,2]-point_matrix[1,2])^2
y<-which.max(distance_matrix[,1])
for (i in 2:k) {
  point_new<-as.matrix(raw_data[y,])
  point_matrix[i,]<-point_new
  distance_matrix[,i]<-(raw_data[,1]-point_matrix[i,1])^2+(raw_data[,2]-point_matrix[i,2])^2
  min_dis=c()
  for (j in 1:n){
    min_dis[j]<-min(distance_matrix[j,1:i])
  }
  y<-which.max(min_dis)
  points(x=point_matrix[i,1],y=point_matrix[i,2],col="red",pch=16,cex=2)
}

clusters<-function(ct,data){
  distance<-matrix(data = NA,nrow=n,ncol=k)
  cluster<-c()
  for (j in 1:k){
    distance[,j]<-(data[,1]-ct[j,1])^2+(data[,2]-ct[j,2])^2
  }
  for (i in 1:n){
    cluster[i]<-which.min(distance[i,])
  }
  cluster
}

cluster<-clusters(point_matrix,raw_data)
data<-cbind(raw_data,cluster)
plot(x=raw_data[,1],y=raw_data[,2],col=data[,3],pch=16)
```

Single-swap heuristic algorithm

```r
###single-swap heuristic###
setwd("C:/Users/Lenovo/Desktop")
raw_data<-read.csv("clustering.csv",header = F)
k=6
n=nrow(raw_data)
#plot(x=raw_data[,1],y=raw_data[,2],col="blue",pch=16)
#initialization:select a point randomly

greedy_centers<-function(data){
  initial_point<-as.matrix(data[sample(row.names(data), 1),])
  y1<-which(data==initial_point)
  distance_matrix<-matrix(data=NA,nrow=n,ncol=k)
  point_matrix<-matrix(data=NA,nrow=k,ncol=3)
  point_matrix[1,1:2]<-initial_point
  point_matrix[1,3]<-y1
  i=1
  distance_matrix[,i]<-(data[,1]-point_matrix[1,1])^2+(data[,2]-point_matrix[1,2])^2
  y<-which.max(distance_matrix[,1])

  for (i in 2:k) {
    point_new<-as.matrix(data[y,])
    point_matrix[i,1:2]<-point_new
    distance_matrix[,i]<-(data[,1]-point_matrix[i,1])^2+(data[,2]-point_matrix[i,2])^2
    min_dis=c()
    for (j in 1:n){
      min_dis[j]<-min(distance_matrix[j,1:i])
    }
    y<-which.max(min_dis)
    #points(x=point_matrix[i,1],y=point_matrix[i,2],col="red",pch=16,cex=2)
    point_matrix[i,3]<-y
  }
  point_matrix
}

count_cluster<-function(ct,data){
  distance<-matrix(data = NA,nrow=n,ncol=k)
  min_distance<-c()
  cluster<-c()
  for (j in 1:k){
    distance[,j]<-(data[,1]-ct[j,1])^2+(data[,2]-ct[j,2])^2
  }
  for (i in 1:n){
    cluster[i]<-which.min(distance[i,])
    min_distance[i]<-min(distance[i,])
  }
  new_distance<-cbind(cluster,min_distance)
  new_distance
}

point_matrix<-as.data.frame(greedy_centers(raw_data))
colnames(point_matrix)<-c("x","y","index")
plot(x=raw_data[,1],y=raw_data[,2],col="blue",pch=16)
points(point_matrix,col="red",pch=16)
cluster_data<-count_cluster(point_matrix[,1:2],raw_data)
data<-cbind(raw_data,cluster)
data_index<-as.data.frame(rownames(data))
data<-cbind(data,data_index)
colnames(data)<-c("x","y","cluster","min_distance","index")
data_swap<-data[-c(point_matrix[,3]),]
cost<-max(data_swap[,4])
##choose and swap point
for (i in 1:1000){
  m_new<-as.data.frame(data_swap[sample(row.names(data_swap), 1),c(1,2,5)])
  c_old<-as.data.frame(point_matrix[sample(row.names(point_matrix), 1),])
  point_matrix_new<-rbind(point_matrix[-which(point_matrix[,3]==c_old[1,3]),],m_new)
  cluster_data_new<-count_cluster(point_matrix_new[,1:2],raw_data)
  data_new<-cbind(raw_data,cluster_data_new,data_index)
  data_swap_new<-data[-as.numeric(c(point_matrix_new[,3])),]
  cost_new<-max(data_new[,4])
  if (cost_new/cost<=0.95){
    point_matrix<-point_matrix_new
    cost<-cost_new
    data_swap<-data_swap_new
    print(cost)
  }
}
points(point_matrix,col="green",pch=16)
plot(x=raw_data[,1],y=raw_data[,2],col=cluster_data_new[,1],pch=16)
```

Spectral Clustering algorithm

```r
###Spectral Clustering algorithm,###

setwd("C:/Users/Lenovo/Desktop")
raw_data<-read.csv("ShapedData.csv",header = F)
k=4
#raw_data<-raw_data[1:100,]
n=nrow(raw_data)
sigma<-0.9
plot(raw_data,col="blue",pch=16)
weight_matrix<-matrix(data=NA,nrow=n,ncol=n)
#compute weight matrix#
for (i in 1:n){
    weight_matrix[,i]<-(raw_data[,1]-raw_data[i,1])^2+(raw_data[,2]-raw_data[i,2])^2
}
weight_matrix_new<-exp(-weight_matrix/sigma)
sum_w<-c()
for (i in 1:n){
  sum_w[i]<-sum(weight_matrix_new[i,])
}
D<-diag(sum_w)
I<-diag(1,nrow=n)
L<-D-weight_matrix_new
#L<-I-D^(-.5)*weight_matrix_new*D^(-.5)
L_e<-eigen(L)

uk<-L_e$vectors[,(n-k+1):n]

#km<-kmeans(uk,k)
#data<-cbind(raw_data,km$cluster)
#plot(x=data[,1],y=data[,2],col=data[,3],pch=16,colnames("x"),rownames("y"))


clusters<-function(ct,data){
  d<-ncol(ct)
  distance<-matrix(data = 0,nrow=n,ncol=k)
  cluster<-c()
  for (j in 1:k){
    for (i in 1:d){
      distance[,j]<-(data[,i]-ct[j,i])^2+distance[,j]
    }
  }
  for (i in 1:n){
    cluster[i]<-which.min(distance[i,])
  }
  cluster
}

count_cent<-function(cluster_data){
  d<-ncol(cluster_data)
  print(d)
  count_table<-table(cluster_data[,d])
  cent_point<-matrix(data=0,ncol=(d-1),nrow=k)
  for (i in 1:k){
    for (j in 1:(d-1)){
      cent_point[i,j]<-(1/count_table[i])*sum(cluster_data[cluster_data[,d]==i,j])
    }
  }
  cent_point
}

#initialization:select k points randomly
uk<-as.data.frame(uk)
initial_point<-as.matrix(uk[sample(row.names(uk), k),])
#points(initial_point,col="red",pch=16)
cluster_int<-clusters(initial_point,uk)
data<-cbind(uk,cluster_int)
cent_points<-count_cent(data)
iteration_distance<-(cent_points-initial_point)
iteartion_rate<-0
for (i in 1:k){
  iteartion_rate<-iteartion_rate+sum(iteration_distance[,i]^2)
}

```

```r
#initialization:select k points randomly
uk<-as.data.frame(uk)
initial_point<-as.matrix(uk[sample(row.names(uk), k),])
#points(initial_point,col="red",pch=16)
cluster_int<-clusters(initial_point,uk)
data<-cbind(uk,cluster_int)
cent_points<-count_cent(data)
iteration_distance<-(cent_points-initial_point)
iteartion_rate<-0
for (i in 1:k){
  iteartion_rate<-iteartion_rate+sum(iteration_distance[,i]^2)
}

record_iteation<-c()
while (iteartion_rate>=0.0001) {
  cluster<-clusters(cent_points,uk)
  data<-cbind(uk,cluster)
  cent_points_new<-count_cent(data)
  iteration_distance<-(cent_points_new-cent_points)
  iteartion_rate<-0
  for (i in 1:k){
    iteartion_rate<-iteartion_rate+sum(iteration_distance[,i]^2)
  }
  print(iteartion_rate)
  cent_points<-cent_points_new
  append(record_iteation, iteartion_rate)
}

plot(x=raw_data[,1],y=raw_data[,2],col=data[,k+1],pch=16)
#points(cent_points,col="blue",pch=18,cex=1.5)

```