STAT 341/641 Final Project

Yijing Liang

For the final exam you will fill in missing pieces of the code in the following blocks. Suppose we observe N data points, $\{\mathbf{x}_i\}_{i=1}^N$, in six dimensions so that $\mathbf{x}_i = (x_{i1}, x_{i2}, x_{i3}, x_{i4}, x_{i5}, x_{i6})$. The isolation forest separates points by repeating the steps in the following algorithm.

1. Randomly choose one of the variables in the dataset, say x_{i1};
2. Randomly choose a number c in the interval $(\min_i(x_{i1}), \max_i(x_{i1})$.
3. Divide the data into groups depending on whether $x_{i1} > c$ or $x_{i1} \leq c$.

We are going to use this part of the isolation forest algorithm to create a dimension reduction technique. For each question you will either change a question mark, ?, to code, supply your own code, or provide a short answer.

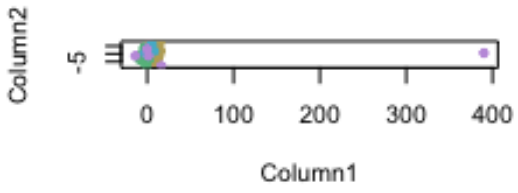
The final project is worth 20 points and due on April 25th. You may consult other students in the class to finish the final, but you must submit your own project. You may not ask for help from me or the teaching assistants.

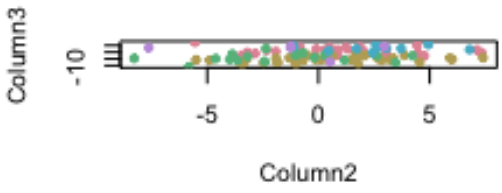
Code Block #1: This code block simulates some data we'll use throughout the project. Then it loads some functions from midterm two that we need.

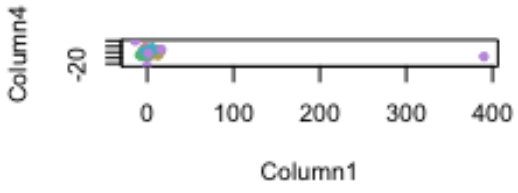
In this block, I have replaced parts of the code with ?. Make the necessary changes to the code. This block contains Q1 - Q3.

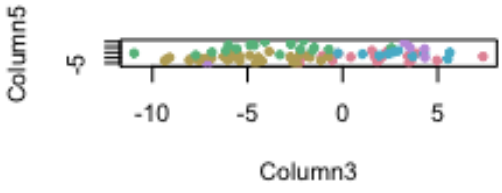
```
#### make high dimensional data with known cluster labels
library(mvtnorm)
set.seed(641)
#### Q1: Sample 25 observations from a normal distribution with mean vector
equal to one in every dimension. (1 point)

c1 <- rmvnorm(25, mean = c(rep(1,6)), sigma = 7*diag(6))

#### Q2: Sample 35 observations from a normal distribution with the given
mean vector equal and a covariance matrix with 3s on the diagonal and .1 in
the off diagonals if i and j are both even or if i and j are both odd.  Let
the other off diagonals be -.1. (1 point)
mydiag_matrix <- 3*(diag(6))
u_inds <- upper.tri(mydiag_matrix)
l_inds <- lower.tri(mydiag_matrix)
mydiag_matrix[u_inds] <- c(-0.1,0.1,-0.1,-0.1,0.1,-0.1,0.1,-0.1,0.1,-0.1,
                           -0.1,0.1,-0.1,0.1,-0.1)
```

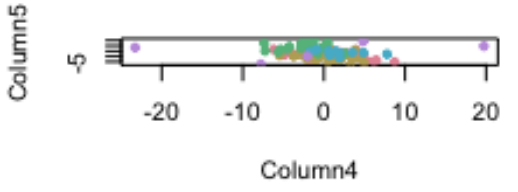
```r
mydiag_matrix[l_inds] <-  t(mydiag_matrix)[l_inds]

#### check symmetry (this should be true)
isSymmetric(mydiag_matrix)

## [1] TRUE

c2 <- rmvnorm(35, mean = c(10,1,-5,0,0.25,-.5), sigma = mydiag_matrix)
c3 <- rmvnorm(18, mean = c(-2,-2,-4,-1,8,-6), sigma = 6*diag(6))
c4 <- matrix(runif(60,1,5),ncol =6)

#### Q3:  Sample 5 points in six-dimensional space from the cauchy
distribution with location parameter zero and scale equal to 4. (1 point)

c5 <- matrix(rcauchy(n = 30, location = 0 , scale = 4),ncol =6)

mypts <- rbind(c1,c2,c3,c4,c5)
mypts <- mypts + matrix(rnorm(n = (nrow(mypts)*6), sd = 2 ),ncol = 6)

out_dat <- cbind(cluster =
c(rep(1,nrow(c1)),rep(2,nrow(c2)),rep(3,nrow(c3)),rep(4,nrow(c4)),rep(5,nrow(
c5))),
                 mypts)

library(colorspace)
## plot dimensions (1,2), (2,3), (1,4), (3,5), (4,5), (5,6)
xaxis <- c(1,2,1,3,4,5) + 1
yaxis <- c(2,3,4,5,5,6) +1
par(mfrow=c(3,2))

for (i in c(1:6)){
  plot(out_dat[,c(xaxis[i],yaxis[i])],typ = "n",xlab =
paste("Column",xaxis[i] -1,sep = ""),
       ylab = paste("Column",yaxis[i] -1,sep = ""))
  points(out_dat[,c(xaxis[i],yaxis[i])],col = rainbow_hcl(5)[out_dat[,1]],pch
= 20)
}
```

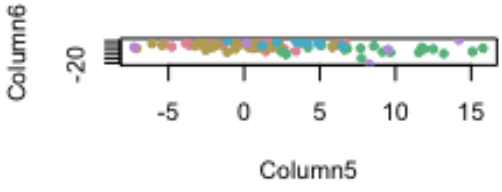













```
## our functions from midterm 2

getClusters <- function(mydata, J,N,K){
  N <- nrow(mydata)
  K <- ncol(mydata)
  cluster_matrix <- matrix(1,N,N)
  for (j in c(1:J)){
    mydim <- sample(1:K,1)
    c <- runif(1,min(mydata[,mydim]),max(mydata[,mydim]))
    tmp <- (mydata[,mydim] < c) %*% t(mydata[,mydim] < c) + (mydata[,mydim]
>= c) %*% t(mydata[,mydim] >= c)
    cluster_matrix <- cluster_matrix * tmp
    }
  return(cluster_matrix)
  }

getAvgMatrix <- function(X,R,ncuts){
  n <- nrow(X)
  k <- ncol(X)
  avg_cluster_matrix <- matrix(0,n,n)
  for (r in c(1:R)){
    obj <- getClusters(X,J = ncuts,N=n,K=k)
    avg_cluster_matrix <- obj/R + avg_cluster_matrix
```

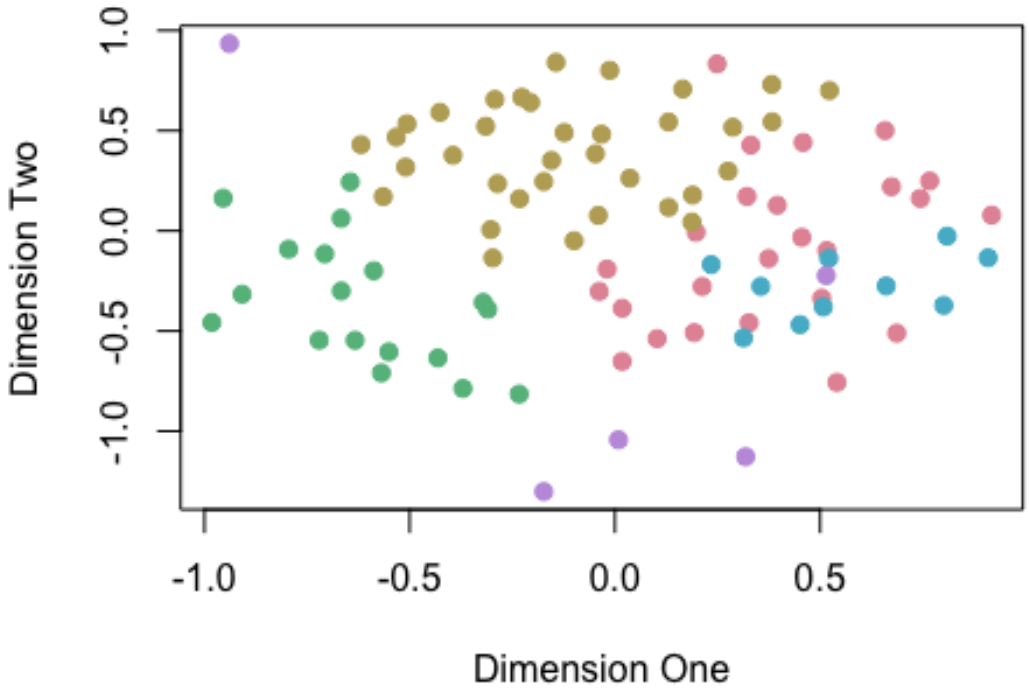
```
    }
  return(avg_cluster_matrix)
}
```

Code Block #2:

In this code chunk, we will get the pairwise neighbor probabilities and run multi-dimensional scaling.

In this block, I have replaced parts of the code with ?. Make the necessary changes to the code. This block contains Q4 and Q5.

```
set.seed(641)
#### Q4: Run the getAvgMatrix function with R = 2500 and ncuts = 5 (1 point). Recall that cell i,j of this symmetric matrix contains the probability that i and j are in the same neighborhood.

avgcluster <- getAvgMatrix(X = mypts, R = 2500, ncuts =5)


#### Q5: Run MDS using the SMACOF package on the dissimilarity matrix containing the probability that points i and j are not in the same neighborhood. (1 point)

library(smacof)

## Loading required package: plotrix

## Loading required package: e1071

##
## Attaching package: 'smacof'

## The following object is masked from 'package:base':
##
##     transform

res_mds <- smacofSym(sim2diss(avgcluster,method=1),ndim =2)

### plot the results
plot(res_mds$conf,typ="n",xlab="Dimension One",ylab="Dimension Two")
points(res_mds$conf, col = rainbow_hcl(5)[out_dat[,1]],pch = 19)
```



Code Block #3:

In the t-SNE method for visualizing high-dimensional data, probabilities p_{ij} are computed using the observed data. Then we choose a low-dimensional embedding $\mathbf{z}_1, \ldots, \mathbf{z}_N$ to minimize

$$\sum_{i=1}^{N} \sum_{j \neq i} p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

where

$$q_{ij} = \frac{\left(1 + d(\mathbf{z}_i, \mathbf{z}_j)^2\right)^{-1}}{\sum_{k \neq i} (1 + d(\mathbf{z}_i, \mathbf{z}_k)^2)^{-1}}.$$

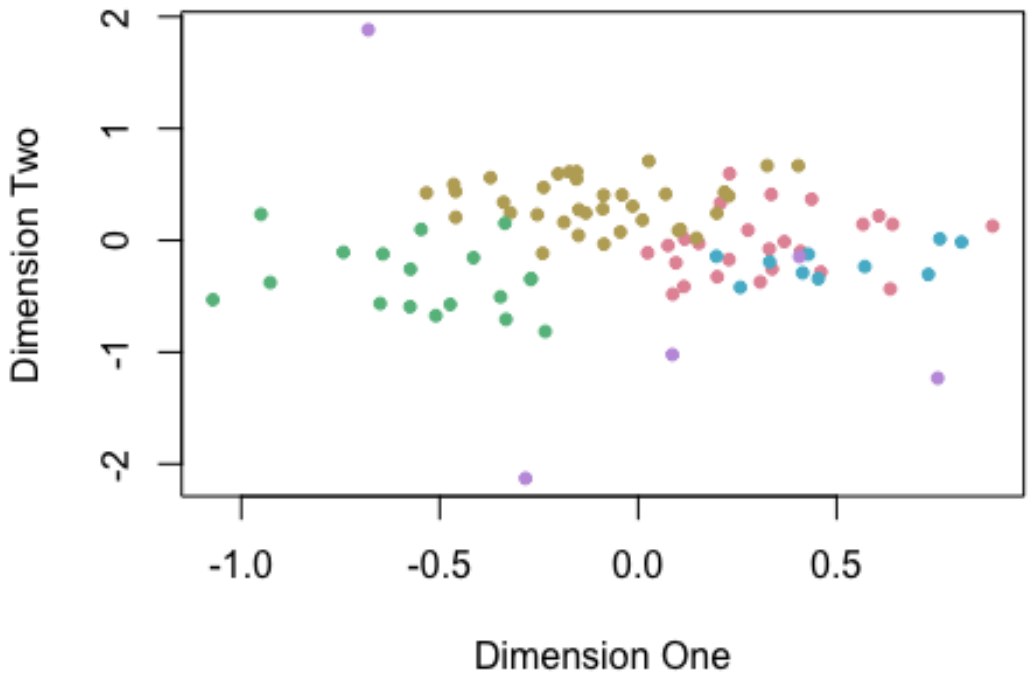
In our version of the algorithm, we will choose a low-dimensional embedding $\mathbf{z}_1, \ldots, \mathbf{z}_N$ to minimize the Hellinger distance between the p_{ij} and q_{ij} where p_{ij} are derived from the pairwise probabilities from the isolation forest cuts. The squared Hellinger distance (times a factor of two) is given by

$$2 * H^2(P, Q) = \sum_{i=1}^{N} \sum_{j \neq i} \left(\sqrt{p_{ij}} - \sqrt{q_{ij}}\right)^2.$$

In this block, I have written code to peform optimization using R's optim function. Warning: the call to optim requies a few minutes to finish. I have replaced parts of the code with a question mark. Make the necessary changes to the code. This block contains Q6 - Q8.

```
#### Adjust the probability matrix so it is a probability distribution over
all neighborhood assignments

diag(avgcluster) <- 0
myp <- avgcluster/rowSums(avgcluster)

myp <- (myp + t(myp))/(2*nrow(out_dat))

#### Write a function to compute the hellinger distance.
#### Q6:  Set the p argument default to be the myp object. (1 point)

hellinger <- function(mydat,p = myp){
  mydat <- matrix(mydat,ncol=2)
  return(sum((sqrt(getQ(mydat)) - sqrt(myp))^2))
}

#### Write a function to compute the Q matrix given a two-dimensional
representation of the data
getQ <- function(Z){
  zz <- as.matrix(dist(Z))
  num <- 1/(1 + zz^2)
  diag(num) <- 0
  mat <- num/rowSums(num)
  return((mat + t(mat))/(2*nrow(Z)))
}

#### Q7:  Use the Hellinger distance function to compute the hellinger
distance between the two-dimensional representation found by MDS and myp. (1
point)

hellinger(res_mds$conf)

## [1] 0.008926253

#### Use the optim function to choose coordinates to minimize the difference
between Q and P in the hellinger distance.  Here we use the MDS dimensions as
a starting value

myres <- optim(c(res_mds$conf),fn = hellinger, method = "BFGS",control =
list(maxit = 450))


#### Compute the hellinger distance and plot the optimized dimensions.
hellinger(myres$par)

## [1] 0.006482216
```

```
plot(matrix(myres$par,ncol = 2),typ="n",xlab="Dimension One",ylab="Dimension
Two")
points(matrix(myres$par,ncol = 2), col = rainbow_hcl(5)[out_dat[,1]],pch =
20)
```




```
#### Q8:  Compute the Mahalanobois distances of the points in cluster five in
the MDS dimensions and the optimized dimensions.  By which method is the
average Mahalanobis distance of cluster five larger? (2 points)

in_c5 <- which(out_dat[,1] == 5)
mds_data <- res_mds$conf
mds_mdist <- mahalanobis(mds_data, colMeans(mds_data), cov(mds_data))[in_c5]
avg_mds_mdist <- mean(mds_mdist)

optimized_data <- matrix(myres$par,ncol = 2)
optimized_mdist <- mahalanobis(optimized_data, colMeans(optimized_data),
cov(optimized_data))[in_c5]
avg_optimized_mdist <- mean(optimized_mdist)

print(avg_optimized_mdist >  avg_mds_mdist)

## [1] TRUE
```

Code Block #4:

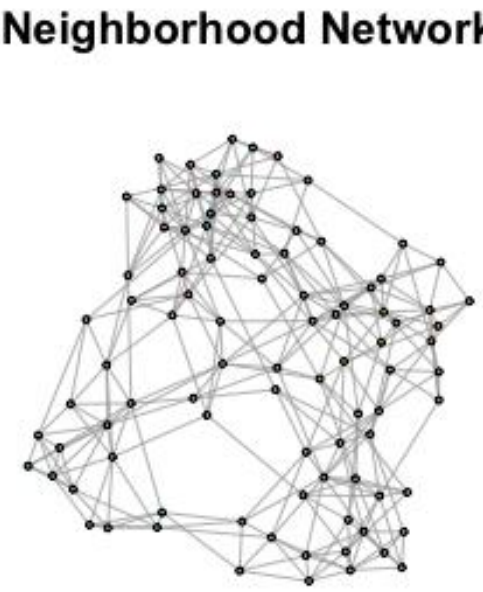
Recall that the Isomap algorithm proceeds by finding the k-nearest neighbors network, computing geodesic distances between all points, and running MDS on the resulting distance matrix. In this block, we will find the nearest neighbors network for our data and run some similar to Isomap. Rather than computing the geodesic distance between points i and j, we will compute the probability that there exists a link from i to j. For example, suppose the shortest path between points i and m goes through points j, k, and l. Then the probability of a link between i and m is

$$p(i \leftrightarrow m)$$
$$= p(i \text{ is a neighbor of } j)p(j \text{ is a neighbor of } k)p(k \text{ is a neighbor of } l)p(l \text{ is a neighbor of } m).$$

In this block, I have replaced parts of the code with ?. Make the necessary changes to the code. For Q11 and Q12, you will need to supply your own code. This block contains Q9 and Q12.

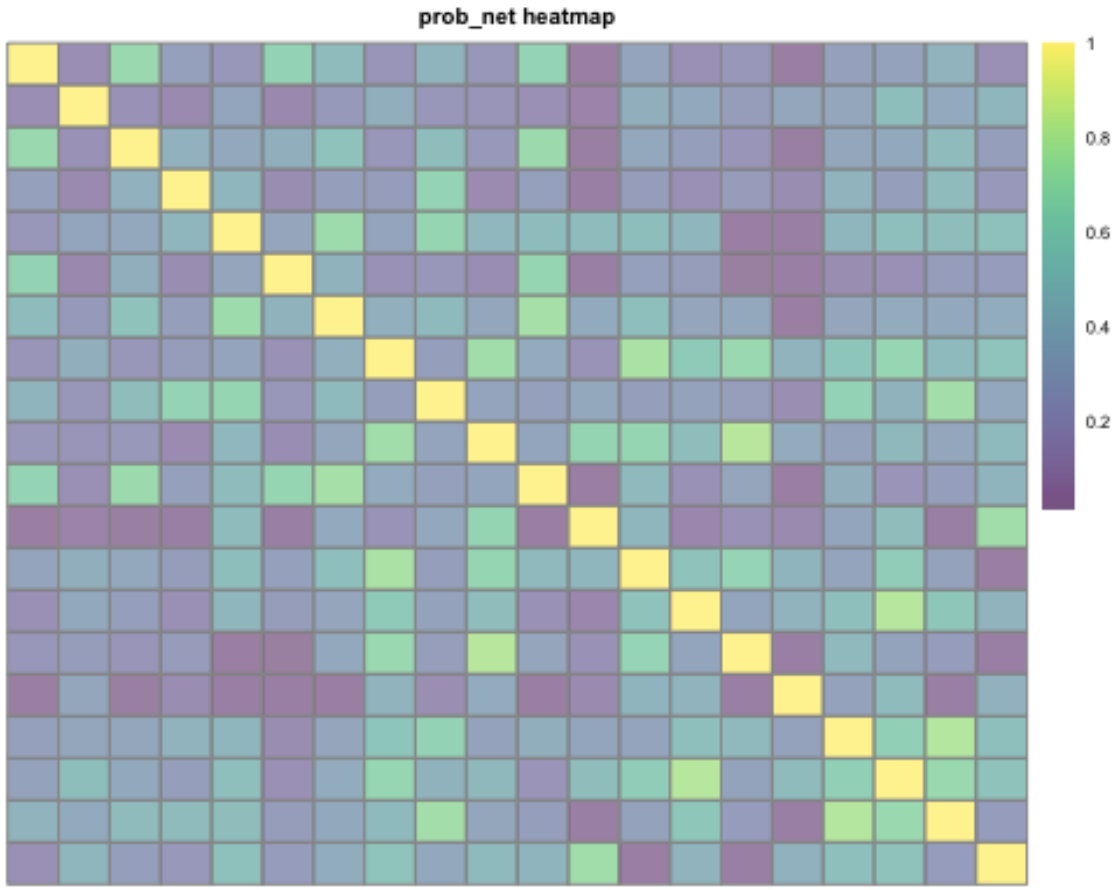
```
#### Here we try an alternative way to get the a lower dimensional
representation.
#### Let's find the k nearest neighbors from the probability distribution in
avgcluster

getNNNetwork <- function(k,pmat = avgcluster){
  snet <- matrix(0,nrow = nrow(pmat),ncol = nrow(pmat))
  order_res <- apply(pmat,1,function(x){
    tmp <- order(x,decreasing = T)[1:k]
  })
  for (i in c(1:nrow(pmat))){
    snet[i,order_res[,i]] <- 1
  }
  return(snet)
}


#### Q9:  Run getNNNetwork with k = 5 (1 point).

mynet <- getNNNetwork(5)


#### Make network plot
library(igraph)

##
## Attaching package: 'igraph'

## The following objects are masked from 'package:stats':
##
##     decompose, spectrum
```

```
## The following object is masked from 'package:base':
##
##     union

#### Make an undirected graph
gnet_undirected <- graph_from_adjacency_matrix(mynet*avgcluster,mode =
"undirected",weighted = T)

#### Q10:  Add a title to the following plot saying "Neighborhood Network".
(1 point)

plot(gnet_undirected,vertex.size = 3, edge.width = .75,edge.arrow.size = .25,
edge.arrow.width = .5, vertex.label = NA, vertex.color =
rainbow_hcl(5)[out_dat[,1]], main="Neighborhood Network")
```

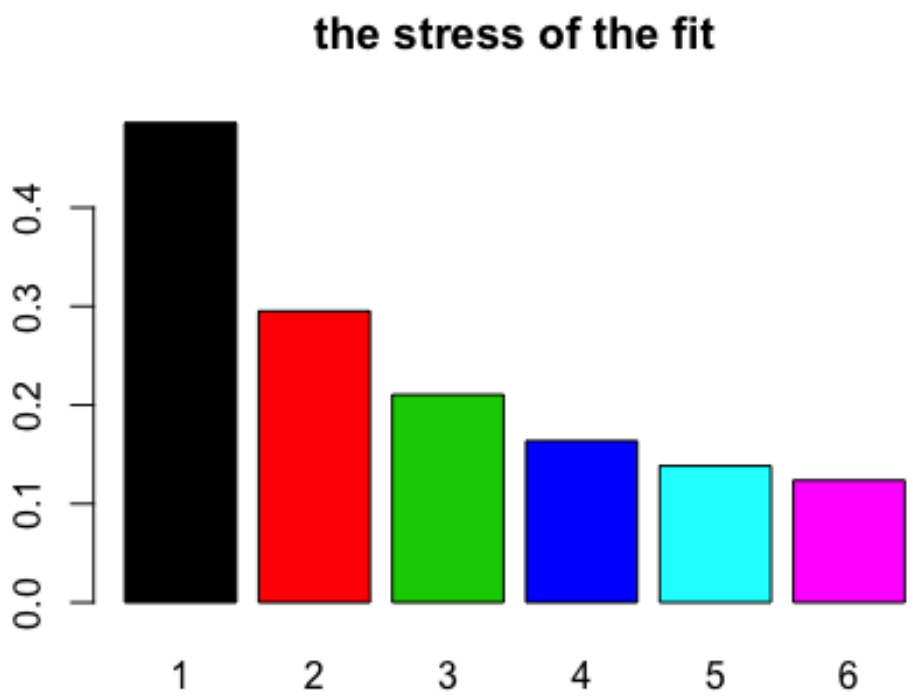



```
#### get shortest paths
nr <- nrow(avgcluster)
prob_net <- matrix(0,nr,nr)
for (i in c(1:nr)){
  all_net <- shortest_paths(gnet_undirected,from = i,to = 1:93)
  for (j in c(1:nr)){
    if( i == j){
      prob_net[i,j] <- 1
```

```
    }
    else{
      ll <- all_net$vpath[[j]]
      myprod <- 1
      for (k in c(1:(length(ll)-1))){
        myprod <- myprod * avgcluster[ll[k],ll[(k+1)]]
        }
      prob_net[i,j] <-myprod
    }
  }
}


library(pheatmap)
#### Q11:  Use the pretty heap mat package to plot the value of the first 20
rows and columns of prob_net (1 point).  See lecture notes.
library(viridisLite)
mypal <- viridis(1000,alpha = .5)
mypal<-viridis(1000,alpha=.5)
pheatmap(prob_net[c(1:20),c(1:20)],display_numbers =
F,color=mypal,cluster_rows = F,cluster_cols = F,fontsize_number = 15,
         cex=.75,cex.main=1,main="prob_net heatmap")
```



```r
#dev.off()



library(smacof)
#### Q12:  Use the SMACOF package to peform MDS on dissimilarity matrix
derived from prob_net (this is like Q5) setting ndim = 1,2,3,4,5,6.  Plot the
stress of the fit associated with the different choices of ndim. (2 points)

mds1<-smacofSym(sim2diss(prob_net,method=1),ndim=1)
mds2<-smacofSym(sim2diss(prob_net,method=1),ndim=2)
mds3<-smacofSym(sim2diss(prob_net,method=1),ndim=3)
mds4<-smacofSym(sim2diss(prob_net,method=1),ndim=4)
mds5<-smacofSym(sim2diss(prob_net,method=1),ndim=5)
mds6<-smacofSym(sim2diss(prob_net,method=1),ndim=6)
mds12<-
rbind(mds1$stress,mds2$stress,mds3$stress,mds4$stress,mds5$stress,mds6$stress
)

### plot the results
barplot(mds12[c(1:6),], main = "the stress of the fit", col = c(1:6),
names.arg = c(1:6))
```



Code Block #5: Challenge: For the three methods considered in the project (MDS, Optimized MDS, and Network MDS), compute the sum of squared residuals by group. Your result should be a table with three columns and five rows. You can use the formula

$$\text{SSR}_g = \sum_{i:\mathbf{x}_i \in C_g} d\left(\mathbf{z}_i, \bar{\mathbf{c}}_g\right)^2$$

where $g = \in \{1,2,3,4,5\}$ is the group number, C_g is the set of points in group g, and $d(\mathbf{z}_i, \bar{\mathbf{c}}_g)$ is the Euclidean distance between observation i's lower dimensional embedding $\mathbf{z}_i$ and $\bar{\mathbf{c}}_g$, the vector of means for group g.

Next, let's define a new way to compute the local outlier factor. Let $N_k(\mathbf{x}_i)$ be the indices of the k nearest neighbors for point $\mathbf{x}_i$. Define our measure to be

$$\text{LOF}(\mathbf{x}_i) = \frac{1}{k} * \frac{\sum_{j:j \in N_k(\mathbf{x}_i)} d\left(\mathbf{x}_i, \mathbf{x}_j\right)}{\sum_{j:j \in \mathcal{N}_k(\mathbf{x}_i)} \sum_{l:l \in N_k(\mathbf{x}_j)} d\left(\mathbf{x}_j, \mathbf{x}_l\right)}$$

where $d(\mathbf{x}_i, \mathbf{x}_j)$ is the Euclidean distance between $\mathbf{x}_i$ and $\mathbf{x}_j$.

Find the outlier factor of the lower dimensional embeddings for points in cluster 5 using $k = 5$ for each of the three methods. Compare these to the outlier factors for the points in cluster using the original data in six dimensions.

For Q13 and Q14, you will need to supply your own code. Q15 requires 2 - 3 sentences to explain your answer. This block contains Q13 - Q15.

```
#### Q13: Compute the SSR for the five clusters under the three different
methods.  Print out a 5 x 3 table with the results. (2 points)
#k=5
res_ults=matrix(0,nrow=5,ncol=3,dimnames=list(c("Cluster
1","2","3","4","5"),c("SSR for MDS","Optimized MDS","Network for MDS")))
network_mds<-smacofSym(sim2diss(prob_net,method=1),ndim=2)
network_data=network_mds$conf
G=5
for(g in 1:G){
  temp_cluster = mds_data[which(out_dat[,1] == g),] #get the clusters
  temp_mean=colMeans(temp_cluster) #get the colmeans of each cluster

  for (k in 1:nrow(temp_cluster)){
    res_ults[g,1]=res_ults[g,1]+dist(rbind(temp_mean,temp_cluster[k,]))^2
  }

  temp_cluster = optimized_data[which(out_dat[,1] == g),] #get the clusters
  temp_mean = colMeans(temp_cluster) #get the colmeans of each cluster

  for (k in 1:nrow(temp_cluster)){
    res_ults[g,2] = res_ults[g,2]+dist(rbind(temp_mean,temp_cluster[k,]))^2
  }
```

```
  temp_cluster = mds2$conf[which(out_dat[,1] == g),] #get the clusters
  temp_mean = colMeans(temp_cluster) #get the colmeans of each cluster

  for (k in 1:nrow(temp_cluster)){
    res_ults[g,3] = res_ults[g,3] + dist(rbind(temp_mean,temp_cluster[k,]))^2
  }
}
res_ults
```

```
##           SSR for MDS Optimized MDS Network for MDS
## Cluster 1   5.5639522      3.066409        5.218160
## 2           5.1654038      3.401897        8.596309
## 3           2.5255279      2.659259        2.878477
## 4           0.7138587      0.596796        1.076537
## 5           4.7168793     10.525603        1.834751
```

```
#### Q14: Write a function (or functions) to compute the local outlier factor
using the definition given in the above text.  Compute the outlier factors
for each of the points in cluster five for each of the three dimension
reduction methods.  Then do the same for the original data.  Print out a 5 x
4 table with the results. (3 points)

library(FNN)
```

```
##
## Attaching package: 'FNN'
```

```
## The following object is masked from 'package:igraph':
##
##     knn
```

```
#mds
mds_k5<-get.knn(mds_data, k=5)
#sum(mds_k5$nn.dist[89,])
#mds_k5$nn.index[89,]
#sum(mds_k5$nn.dist[c(90,91,16,77,10),])
lof_mds1<-
(sum(mds_k5$nn.dist[89,])/sum(mds_k5$nn.dist[c(91,90,77,16,66),]))/5
#lof_mds1
#sum(mds_k5$nn.dist[90,])
#mds_k5$nn.index[90,]
#sum(mds_k5$nn.dist[c(89,2,88,16,15),])
lof_mds2<-(sum(mds_k5$nn.dist[90,])/sum(mds_k5$nn.dist[c(89,2,91,16,88),]))/5
#lof_mds2
#sum(mds_k5$nn.dist[91,])
#mds_k5$nn.index[91,]
#sum(mds_k5$nn.dist[c(89,77,66,16,90),])
lof_mds3<-
(sum(mds_k5$nn.dist[91,])/sum(mds_k5$nn.dist[c(89,77,66,16,90),]))/5
#lof_mds3
#sum(mds_k5$nn.dist[92,])
```

```
#mds_k5$nn.index[92,]
#sum(mds_k5$nn.dist[c(87,20,81,14,22),])
lof_mds4<-
(sum(mds_k5$nn.dist[92,])/sum(mds_k5$nn.dist[c(87,20,81,14,83),]))/5
#lof_mds4
#sum(mds_k5$nn.dist[93,])
#mds_k5$nn.index[93,]
#sum(mds_k5$nn.dist[c(28,46,56,71,43),])
lof_mds5<-
(sum(mds_k5$nn.dist[93,])/sum(mds_k5$nn.dist[c(28,46,56,31,42),]))/5
#lof_mds5
lof_mds<-as.data.frame(rbind(lof_mds1,lof_mds2,lof_mds3,lof_mds4,lof_mds5))
#lof_mds

#optimized

optimized_k5<-get.knn(optimized_data, k=5)
#sum(optimized_k5$nn.dist[89,])
#optimized_k5$nn.index[89,]
#sum(optimized_k5$nn.dist[c(77,66,16,10,88),])
lof_opt1<-
(sum(optimized_k5$nn.dist[89,])/sum(optimized_k5$nn.dist[c(77,66,16,10,88),])
)/5
#lof_opt1
#sum(optimized_k5$nn.dist[90,])
#optimized_k5$nn.index[90,]
#sum(optimized_k5$nn.dist[c(89,2,86,82,88),])
lof_opt2<-
(sum(optimized_k5$nn.dist[90,])/sum(optimized_k5$nn.dist[c(89,2,86,82,88),]))
/5
#lof_opt2
#sum(optimized_k5$nn.dist[91,])
#optimized_k5$nn.index[91,]
#sum(optimized_k5$nn.dist[c(89,77,90,66,72),])
lof_opt3<-
(sum(optimized_k5$nn.dist[91,])/sum(optimized_k5$nn.dist[c(89,77,90,66,72),])
)/5
#lof_opt3
#sum(optimized_k5$nn.dist[92,])
#optimized_k5$nn.index[92,]
#sum(optimized_k5$nn.dist[c(87,14,80,22,6),])
lof_opt4<-
(sum(optimized_k5$nn.dist[92,])/sum(optimized_k5$nn.dist[c(87,14,80,22,6),]))
/5
#lof_opt4
#sum(optimized_k5$nn.dist[93,])
#optimized_k5$nn.index[93,]
#sum(optimized_k5$nn.dist[c(31,57,32,48,42),])
lof_opt5<-
(sum(optimized_k5$nn.dist[93,])/sum(optimized_k5$nn.dist[c(31,57,32,48,42),])
```

```
)/5
#lof_opt5
lof_opt<-as.data.frame(rbind(lof_opt1,lof_opt2,lof_opt3,lof_opt4,lof_opt5))
#lof_opt


#network mds
network_k5<-get.knn(mds2$conf, k=5)
#sum(network_k5$nn.dist[89,])
#network_k5$nn.index[89,]
#sum(network_k5$nn.dist[c(16,91,77,90,4),])
lof_net1<-
(sum(network_k5$nn.dist[89,])/sum(network_k5$nn.dist[c(16,91,77,90,4),]))/5
#lof_net1
#sum(network_k5$nn.dist[90,])
#network_k5$nn.index[90,]
#sum(network_k5$nn.dist[c(16,15,89,21,26),])
lof_net2<-
(sum(network_k5$nn.dist[90,])/sum(network_k5$nn.dist[c(16,15,89,21,26),]))/5
#lof_net2
#sum(network_k5$nn.dist[91,])
#network_k5$nn.index[91,]
#sum(network_k5$nn.dist[c(89,77,16,66,90),])
lof_net3<-
(sum(network_k5$nn.dist[91,])/sum(network_k5$nn.dist[c(89,77,16,66,90),]))/5
#lof_net3
#sum(network_k5$nn.dist[92,])
#network_k5$nn.index[92,]
#sum(network_k5$nn.dist[c(82,25,18,8,14),])
lof_net4<-
(sum(network_k5$nn.dist[92,])/sum(network_k5$nn.dist[c(82,25,18,8,14),]))/5
#lof_net4
#sum(network_k5$nn.dist[93,])
#network_k5$nn.index[93,]
#sum(network_k5$nn.dist[c(49,63,60,46,43),])
lof_net5<-
(sum(network_k5$nn.dist[93,])/sum(network_k5$nn.dist[c(49,63,60,46,43),]))/5
#lof_net5
lof_net<-as.data.frame(rbind(lof_net1,lof_net2,lof_net3,lof_net4,lof_net5))
#lof_net

#originaldata
#mypts
origin_k5<-get.knn(mypts, k=5)
#sum(origin_k5$nn.dist[89,])
#origin_k5$nn.index[89,]
#sum(origin_k5$nn.dist[c(12,17,10,21,62),])
lof_ori1<-
(sum(origin_k5$nn.dist[89,])/sum(origin_k5$nn.dist[c(12,17,10,21,62),]))/5
#lof_ori1
```

```r
#sum(origin_k5$nn.dist[90,])
#origin_k5$nn.index[90,]
#sum(origin_k5$nn.dist[c(81,34,41,54,2),])
lof_ori2<-
(sum(origin_k5$nn.dist[90,])/sum(origin_k5$nn.dist[c(81,34,41,54,2),]))/5
#lof_ori2
#sum(origin_k5$nn.dist[91,])
#origin_k5$nn.index[91,]
#sum(origin_k5$nn.dist[c(77,66,72,75,73),])
lof_ori3<-
(sum(origin_k5$nn.dist[91,])/sum(origin_k5$nn.dist[c(77,66,72,75,73),]))/5
#lof_ori3
#sum(origin_k5$nn.dist[92,])
#origin_k5$nn.index[92,]
#sum(origin_k5$nn.dist[c(18,22,14,87,20),])
lof_ori4<-
(sum(origin_k5$nn.dist[92,])/sum(origin_k5$nn.dist[c(18,22,14,87,20),]))/5
#lof_ori4
#sum(origin_k5$nn.dist[93,])
#origin_k5$nn.index[93,]
#sum(origin_k5$nn.dist[c(32,90,58,57,33),])
lof_ori5<-
(sum(origin_k5$nn.dist[93,])/sum(origin_k5$nn.dist[c(32,90,58,57,33),]))/5
#lof_ori5
lof_ori<-as.data.frame(rbind(lof_ori1,lof_ori2,lof_ori3,lof_ori4,lof_ori5))
#lof_ori


lof<-data.frame(lof_mds,lof_net,lof_opt,lof_ori)
rownames(lof)[rownames(lof)=="lof_mds1"]<-"X 1"
rownames(lof)[rownames(lof)=="lof_mds2"]<-"X 2"
rownames(lof)[rownames(lof)=="lof_mds3"]<-"X 3"
rownames(lof)[rownames(lof)=="lof_mds4"]<-"X 4"
rownames(lof)[rownames(lof)=="lof_mds5"]<-"X 5"

#colnames(lof)
names(lof)[names(lof) == "V1"] <- "lof for MDS"
names(lof)[names(lof) == "V1.1"] <- "lof for Network MDS"
names(lof)[names(lof) == "V1.2"] <- "lof for Optimized data"
names(lof)[names(lof) == "V1.3"] <- "lof for original data"
lof
```

```
##     lof for MDS lof for Network MDS lof for Optimized data
## X 1  0.04151339          0.03728165             0.10944683
## X 2  0.06012088          0.05677150             0.13930934
## X 3  0.06373922          0.04764356             0.13092911
## X 4  0.03742951          0.03639955             0.03474537
## X 5  0.17348502          0.04759841             0.40387956
##     lof for original data
## X 1            0.12470711
```

```
## X 2            0.10332089
## X 3            0.13670081
## X 4            0.04191104
## X 5            1.62755005
```

Q15: Which dimension reduction method would you choose to study outliers in the lower dimensional embedding? (1 point)
#Since the lof of the optimized MDS is more relative to the original data and the outlier is still an outlier based on lof for optimized data. so we should choose the optimized mds to study outliers.