STAT 341/641 Midterm Two Project

Yijing Liang (student id: 301301244)
March 30, 2020

For the second midterm you will fill in missing pieces of the code in the following blocks. Suppose we observe data points, , in three dimensions so that . Recall that the isolation forest separates points by repeating the steps in the following algorithm.

Randomly choose one of the variables in the dataset, say ;
Randomly choose a number in the interval .
Divide the data into groups depending on whether or .

The midterm project is worth 16 points and due on April 2nd. You may consult other students in the class to finish the midterm, 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 divides the data into groups. The results are recorded in an matrix where entry indicates whether observation and belong to the same cluster. In this block, I have replaced parts of the code with a question mark. Make the necessary changes to the code.

```
## you will need to change the path to load the data
out_dat <- read.csv("/Users/yijingliang/Downloads/cluster.csv")

getClusters <- function(mydata, J){
 N <- nrow(mydata)
 K <- ncol(mydata)
 cluster_matrix <- matrix(1,N,N)
 for (j in c(1:J)){
 ## 1. sample a dimension (1 point)

 mydim <- sample(1:K,1)

 ## 2. sample a number c greater than the minimum and less than the maximum of that dimension (1 point)

 c <- runif(1,min(mydata[,mydim]),max(mydata[,mydim]))

 ## 3. compute a matrix that determines whether the each pair of points satifies the condition (1 point)

 tmp <- (mydata[,mydim] < c) %*% t(mydata[,mydim] < c) | (mydata[,mydim] >= c) %*% t(mydata[,mydim] >= c)

 ## 4. Update the cluster matrix (1 point)

 cluster_matrix <- cluster_matrix * tmp
 }
 return(cluster_matrix)

}

set.seed(341)
mycluster_matrix <- getClusters(out_dat, J = 4)
```

Code Block #2: This code block divides the data into groups. The results are recorded in an matrix where entry indicates whether observation and belong to the same cluster. In this block, I have replaced parts of the code with an asterik. Make the necessary changes to the code.

```
## 5. determine the number of groups (1 point)

unique_row <- mycluster_matrix[which(!duplicated(mycluster_matrix)),]
number_groups <- nrow(unique_row)
print(number_groups)
## [1] 7
```

[1] 7
6. get the group assignments (1 point)

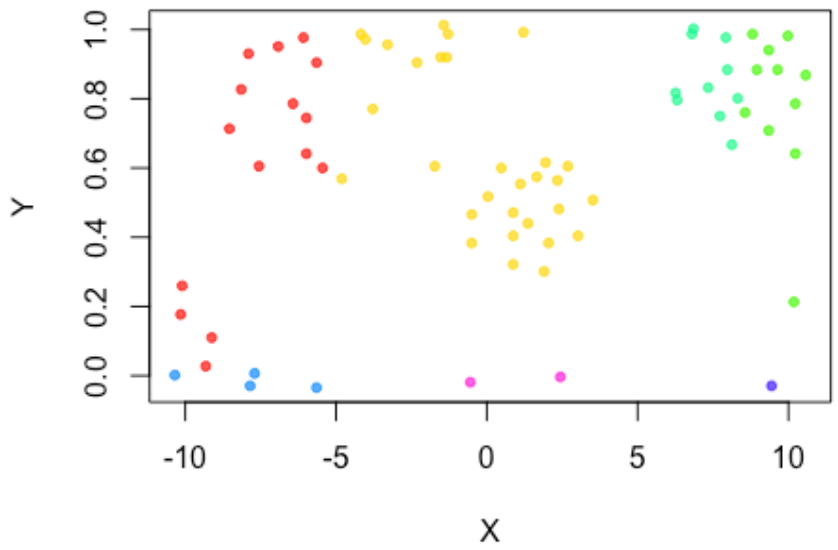
```
mygroups <- apply(unique_row,1,function(x){which(x==1)})
cluster_assignments <- numeric(nrow(out_dat))
for (ii in c(1:number_groups)){
 cluster_assignments[mygroups[[ii]]] <- ii
}
```

7. assign a colour to each group (1 point)

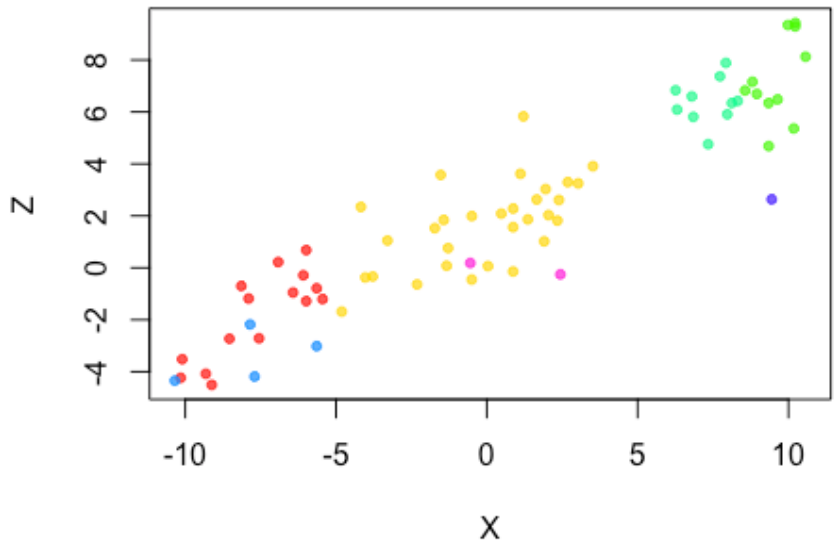
```
mycols <- rainbow(n = number_groups, alpha = .75)
group_cols <- mycols[cluster_assignments]
```

8. Make a plot the data with the points coloured by cluster number (1 point)

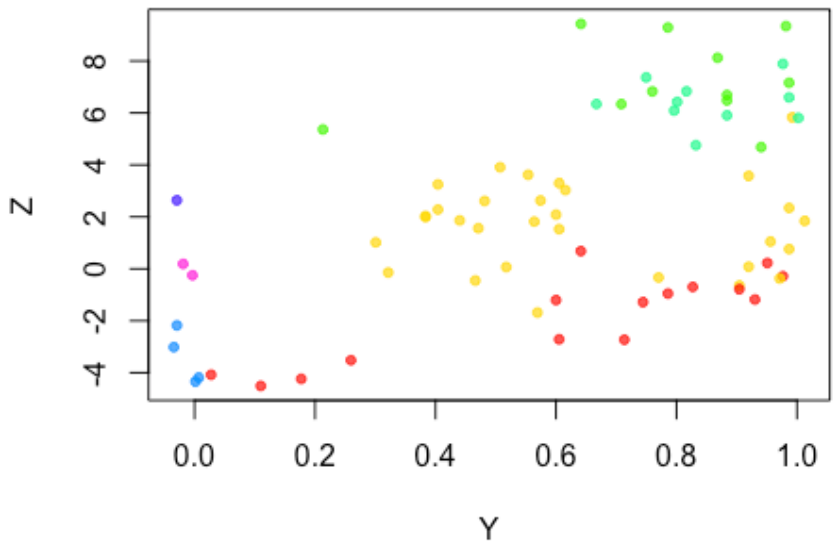
```
plot(out_dat[,c(1,2)],typ="p",col=group_cols,pch = 20)
```




```
plot(out_dat[,c(1,3)],typ="p",col=group_cols,pch = 20)
```




```
plot(out_dat[,c(2,3)],typ="p",col=group_cols,pch = 20)
```



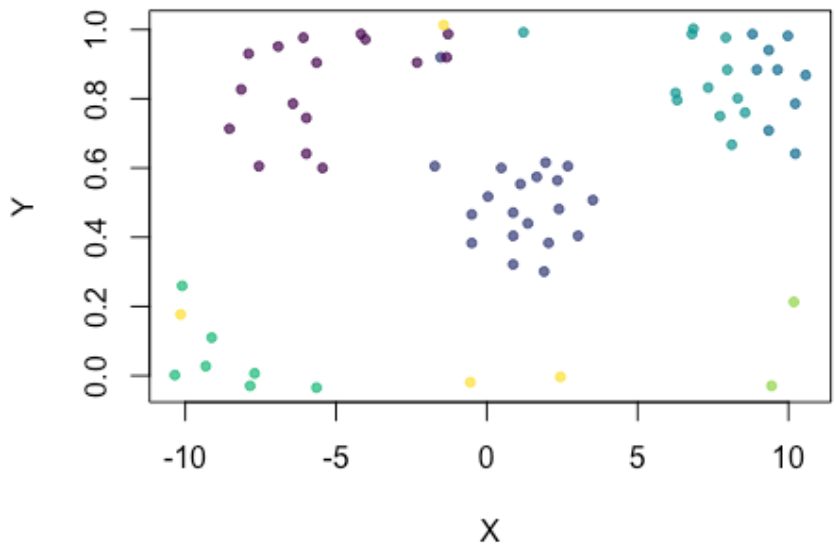
Code Block #3: Now, let's repeat this times and average the results of the cluster matrices. Entry in the resulting matrix is the probability that two points belong to the same cluster, . We will set the number of clusters, , to be equal to the number of clusters in the single run. Then we will sample from the set of cluster assignments and plot the results.

In the following code block, I have written a function that computes the logged probability of a cluster

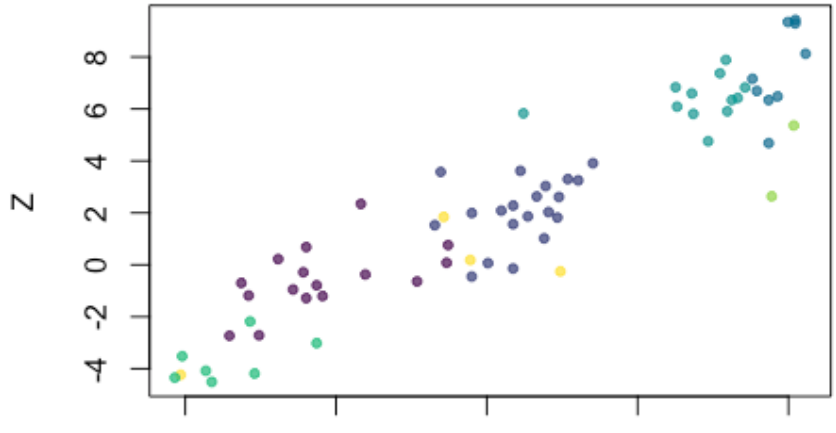
You only have to run this. You needn't worry about how I derived this.

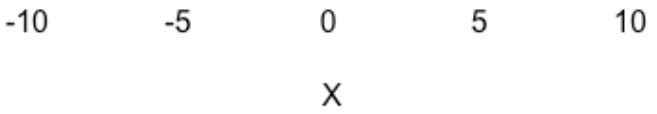
```
set.seed(641)
R <- 2500
avg_cluster_matrix <- matrix(0,nrow(out_dat),nrow(out_dat))
for (r in c(1:R)){
 ## 9. call the function (1 point)
 obj <- getClusters(out_dat,J = 4)

 ## 10. average the results (1 point)
 avg_cluster_matrix <- obj/R + avg_cluster_matrix
}

## print some entries

avg_cluster_matrix[ 1:10,1:10]
##        [,1]   [,2]   [,3]   [,4]   [,5]   [,6]   [,7]   [,8]   [,9]  [,10]
##  [1,] 1.0000 0.8736 0.7628 0.6720 0.4336 0.7036 0.5524 0.3888 0.4316 0.6316
##  [2,] 0.8736 1.0000 0.7828 0.6904 0.4448 0.6744 0.5756 0.3984 0.3928 0.5796
##  [3,] 0.7628 0.7828 1.0000 0.8072 0.5916 0.7216 0.4616 0.5100 0.4484 0.5396
##  [4,] 0.6720 0.6904 0.8072 1.0000 0.6684 0.8224 0.5676 0.5572 0.4944 0.6604
##  [5,] 0.4336 0.4448 0.5916 0.6684 1.0000 0.6464 0.5240 0.8024 0.5708 0.5144
##  [6,] 0.7036 0.6744 0.7216 0.8224 0.6464 1.0000 0.6660 0.5888 0.6188 0.7668
##  [7,] 0.5524 0.5756 0.4616 0.5676 0.5240 0.6660 1.0000 0.5944 0.6580 0.6420
##  [8,] 0.3888 0.3984 0.5100 0.5572 0.8024 0.5888 0.5944 1.0000 0.7248 0.4604
##  [9,] 0.4316 0.3928 0.4484 0.4944 0.5708 0.6188 0.6580 0.7248 1.0000 0.6012
## [10,] 0.6316 0.5796 0.5396 0.6604 0.5144 0.7668 0.6420 0.4604 0.6012 1.0000
## 11. set the number of clusters equal to that for the single run (1 point)
ngroups <- nrow(obj[which(!duplicated(obj)),])
ngroups
## [1] 9
## Sampling from the set of all clusters by changing one point at a time
## You don't need to understand this part!!!
S <- 250000
set.seed(341)
mycluster <- cluster_assignments
s <- 0
myclusterlist<-matrix(NA,nrow=S,ncol=73)
while(s < S){
 cluster_proposal <- mycluster
 ind <- sample(c(1:nrow(avg_cluster_matrix)),1)
```

```
 newc <- sample(c(1:ngroups),1)
 newvec <- as.numeric(newc == cluster_proposal[-ind])
 oldvec <- as.numeric(cluster_proposal[ind] == cluster_proposal[-ind])

 p1 <- sum(log(newvec*avg_cluster_matrix[ind,-ind] + (1-newvec)*(1-avg_cluster_matrix[ind,-
ind])))
 p2 <- sum(log(oldvec*avg_cluster_matrix[ind,-ind] + (1-oldvec)*(1-avg_cluster_matrix[ind,-
ind])))

 if(p1 >p2){
   mycluster[ind] <- newc
 }
 s <- s + 1
 myclusterlist[s,]<-mycluster
}

## 12. plot the results (1 point - you can use the other code block to do this)
uniquerow2<-myclusterlist[which(!duplicated(myclusterlist)),]
clusters<-nrow(uniquerow2)
cluster_assignments2<-uniquerow2[sample(clusters,1),]

mycols2<-hcl.colors(n=number_groups,palette = "viridis", alpha = .75)
group_cols2<-mycols2[cluster_assignments2]


plot(out_dat[,c(1,2)],typ="p",col=group_cols2 ,pch = 20)
```

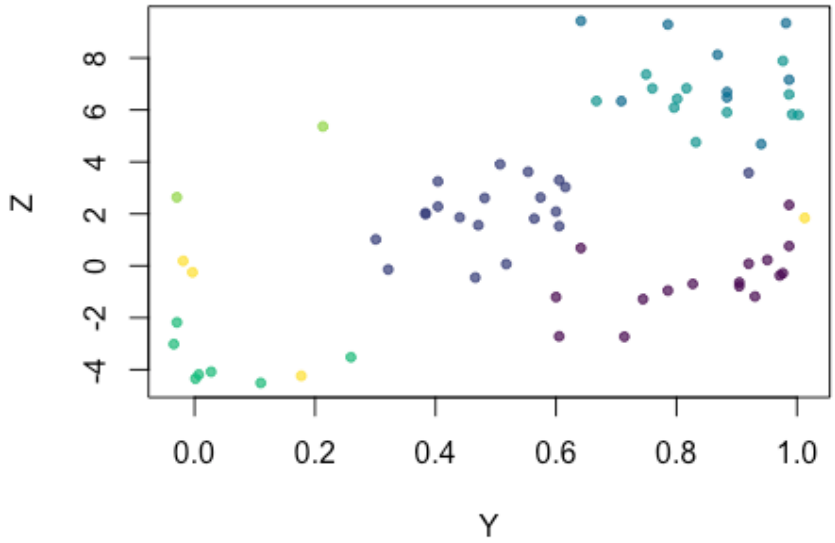



```
plot(out_dat[,c(1,3)],typ="p",col=group_cols2 ,pch = 20)
```






```
plot(out_dat[,c(2,3)],typ="p",col=group_cols2 ,pch = 20)
```



Code Block #4: Challenge: Now compare the results of the single run to the 2,500 runs. For the two results, compute the sum of squared residuals given by

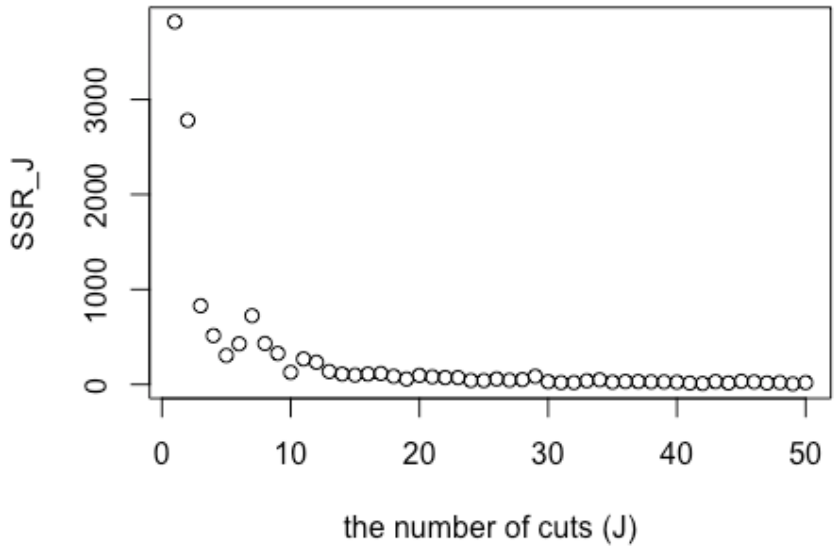
where is the number of groups, is the set of points in cluster , and is the Euclidean distance between and the mean of its cluster .

```
## 13. Compute the SSR for the two set of results (2 points)
library(philentropy)

#the cluster centers
#for single run
centers1=matrix(NA,ngroups,ncol(out_dat))
#cluster assignment for single run
uniquerow3<-obj[which(!duplicated(obj)),]
clusters2<-nrow(uniquerow3)
cluster_assignments1<-uniquerow3[sample(clusters2,1),]


#calculate the euclidean distance ():
Euclidean_distance<-function(x,x0){
 return(sum((x-x0)^2))
}

SSR=0
for(i in 1:ngroups){
 centers1[i,]<-apply(out_dat[cluster_assignments1==i,],2,mean)
 SSR=SSR+sum(apply(out_dat[cluster_assignments1==i,],1,Euclidean_distance,centers1[i,]))
}


cluster_assignments2
##  [1] 1 1 1 1 1 1 1 1 8 9 2 1 1 1 1 3 4 4 4 4 4 4 4 3 3 3 2 2 2 2 2 2 2 5 7 5 6 6 1
## [39] 9 1 2 1 7 4 1 7 2 2 2 4 4 4 3 3 3 4 4 3 3 5 5 5 5 5 7 2 2 2 2 2 2 2 2
#the cluster centers
#for 2500
centers=matrix(NA,number_groups,ncol(out_dat))


SSR2=0
for(i in 1:number_groups){
 centers[i,]<-apply(out_dat[cluster_assignments2==i,],2,mean)
```

```
 SSR2=SSR2+sum(apply(out_dat[cluster_assignments2==i,],1,Euclidean_distance,centers[i,]))
}

print(c(SSR,SSR2))
## [1] 186.3945 380.3080
# the second results is better
## 14. In the space below, answer the following question.  What happens to the SSR for a single run
of our clustering method as the number of cuts (J) grows large?  Why? (2 points)

SSR_J<-numeric(50)
for(j in 1:50){
 obj<-getClusters(out_dat,J=j)
 unique_row3<-obj[which(!duplicated(obj)),]
 ngroups<-nrow(unique_row3)
 print(ngroups)
 mygroups<-apply(unique_row3,1,function(x){which(x==1)})
 cluster_assignments1<-numeric(nrow(out_dat))
 for (ii in c(1:ngroups)){
   cluster_assignments1[mygroups[[ii]]] <- ii
 }

 centers1=matrix(NA,ngroups,ncol(out_dat))

 for(i in 1:ngroups){
   centers1[i,]<-apply(out_dat[cluster_assignments1==i,],2,mean)

 SSR_J[j]=SSR_J[j]+sum(apply(out_dat[cluster_assignments1==i,],1,Euclidean_distance,centers1
[i,]))
 }
}
## [1] 2
## [1] 3
## [1] 5
## [1] 7
## [1] 9
## [1] 9
## [1] 13
## [1] 8
## [1] 19
## [1] 17
## [1] 16
## [1] 21
## [1] 27
## [1] 32
## [1] 27
## [1] 29
## [1] 35
## [1] 35
## [1] 30
## [1] 32
## [1] 32
## [1] 33
## [1] 37
## [1] 45
## [1] 49
## [1] 42
## [1] 46
## [1] 41
## [1] 36
## [1] 47
## [1] 53
## [1] 52
## [1] 52
## [1] 46
## [1] 54
## [1] 59
## [1] 49
## [1] 53
## [1] 53
## [1] 54
## [1] 56
## [1] 59
## [1] 54
## [1] 56
## [1] 53
```

```
## [1] 56
## [1] 57
## [1] 55
## [1] 66
## [1] 53
```

```
#create a plot and see the relationship between the number of cuts(J) and SSR
plot(1:50,SSR_J,xlab="the number of cuts (J)")
```




```
# as the number of cuts in crease, the SSR for a single run of our clustering become smaller. and it has the tendency to 0.
```