Block 3: clustering

## Practical 1: Simulated data

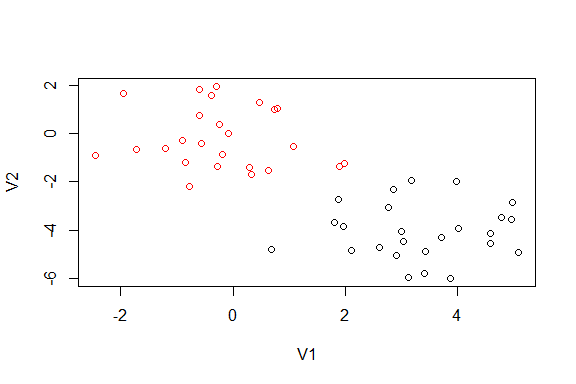
### K-Means Clustering

We begin with kmeans on a simple simulated example in which there truly are two clusters in the data. The dataset consists of 50 observations, each with two values. The first 25 observations have a mean shift relative to the next 25 observations. You can read the data from the file makesimdat.txt:

> x <- read.table('makesimdat.txt')

**Question 1.** Visualize this dataset. Confirm that the first 25 observations are separated from the second 25 observations.

> plot(x,col=c(rep(1,25),rep(2,25)))



We now perform K-means clustering with K = 2.

> km.out=kmeans (x,2, nstart =20)

The cluster assignments of the 50 observations are contained in km.out$cluster.

**Question 2.** Have a look at km.out$cluster and confirm that the two groups of 25 observations are separated in two different clusters.

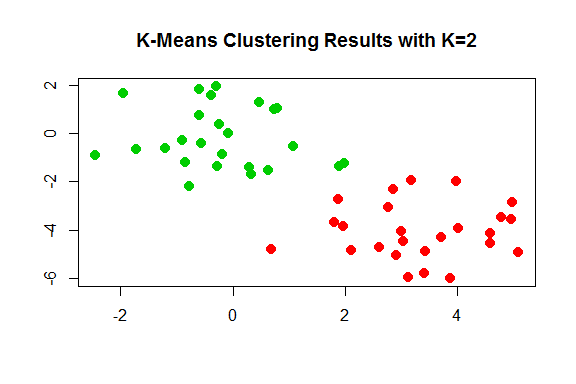
> km.out$cluster

## [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2  
## [36] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2

*The first 25 observations are in cluster 1, the last 25 observations are in cluster 2.*

The K-means clustering perfectly separated the observations into two clusters even though we did not supply any group information to kmeans. We can plot the data, with each observation colored according to its cluster assignment:

> plot(x, col=(km.out$cluster +1), main="K-Means Clustering Results with K=2", xlab="", ylab="", pch=20, cex=2)



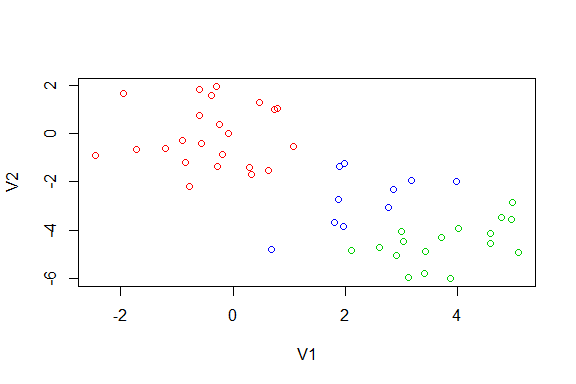
Here the observations can be easily plotted because they are two-dimensional. In this example, we knew that there really were two clusters because we generated the data. However, for real data, in general we do not know the true number of clusters. We could instead have performed K-means clustering on this example with K = 3.

**Question 3.** Perform K-means using K=3. Have a look at the resulting object, in particular the $cluster component. Also, plot the clustering result in the same way as you did above for the clustering with K=2. What do you observe?

> km.out3=kmeans (x,3, nstart =20)  
> km.out3$cluster

## [1] 2 3 2 3 2 2 2 3 2 3 2 3 2 3 2 3 2 2 2 2 2 3 2 2 2 1 1 1 1 1 1 1 1 1 1  
## [36] 1 1 1 1 1 1 1 1 3 1 3 1 1 1 1

> plot(x,col=(km.out3$cluster+1))



*One of the two original clusters is more or less kept as it was, the other one is split into two.*

We will now compare the use of nstart=1 with nstart=20.

> set.seed(3)  
> km.out1=kmeans (x,3, nstart = 1)  
> km.out20=kmeans (x,3, nstart = 20)

Set.seed is used here in order to be able to reproduce exactly the same results upon rerunning the code. You don't need to worry about its meaning.

**Question 4.** Have a look at $tot.withinss for both km.out1 and km.out20. How can you see that the clustering with nstart = 20 is better than the one with nstart = 1? Refer to the reader for an explanation about tot.withinss.

> km.out1$tot.withinss

## [1] 104.3319

> km.out20$tot.withinss

## [1] 97.97927

*The total sum of squares is lower when using nstart = 20 compared to nstart = 1*

### Hierarchical Clustering

We now use the data from above with hclust, which implements hierarchical clustering in R. Before doing so, the dist function is used to compute the Euclidean distance matrix between the items that we want to cluster.

**Question 5.** Have a look at the result of applying dist to x. How many rows and how many columns are there in the distance matrix. Why is that? And why is only about half of the values in the distance matrix shown?

> dist(x)

##

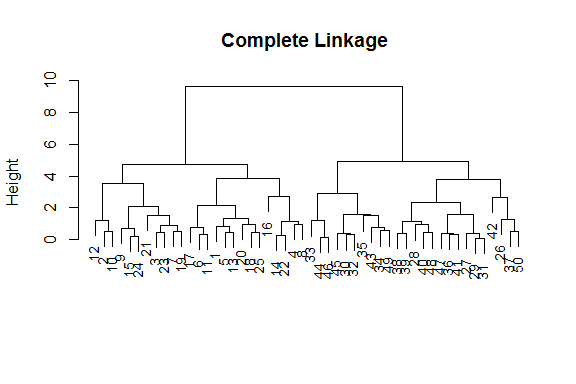
*Fifty rows and fifty columns; one for each item in dataset x. Only the part of the matrix below the diagonal is shown because the diagonal itself would contain zeros (distance of an item with itself) and the values above the diagonal are identical to the ones below the diagonal (the distance between i and j is the same as the distance between j and i).*

We now use this distance matrix to perform hierarchical clustering with two different linkage methods (if you don't remember what this means, consult the reader).

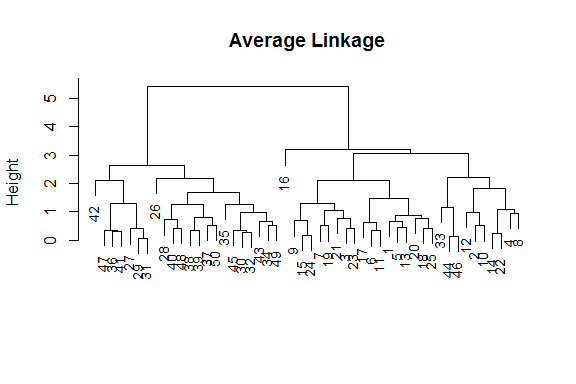
> hc.complete = hclust(dist(x), method="complete")  
> hc.average = hclust(dist(x), method ="average")

**Question 6.** Use plots of the dendrograms to get an idea about the results of these two different clustering attempts. How do these results compare to each other, and to the results of kmeans?

> plot(hc.complete ,main="Complete Linkage ", xlab="", sub="", cex=.9)



> plot(hc.average , main="Average Linkage", xlab="", sub="", cex=.9)



*The numbers at the bottom of the plot identify each observation. With complete linkage, the two main clusters separate observation 1-25 from 26-50, as was done by kmeans. With average linkage, this is not completely the case. For example, observation 44 and 46 have switched from one cluster to the other.*

To determine the cluster labels for each observation associated with a given cut of the dendrogram, we can use the cutree function.

**Question 7.** Confirm the result obtained with the plots obtained above (question 6) using cutree.

> cutree(hc.complete , 2)

## [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2  
## [36] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2

> cutree(hc.average , 2)

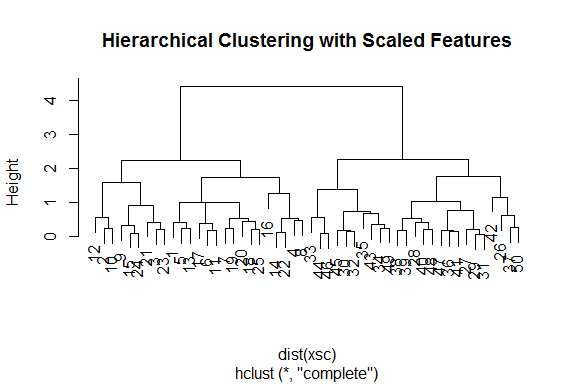
## [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 1 2 2  
## [36] 2 2 2 2 2 2 2 2 1 2 1 2 2 2 2

*For complete linkage, the cutree result shows that the first 25 observations are in cluster 1, and the last 25 observations are in cluster 2. For average linkage, there are a few exceptions.*

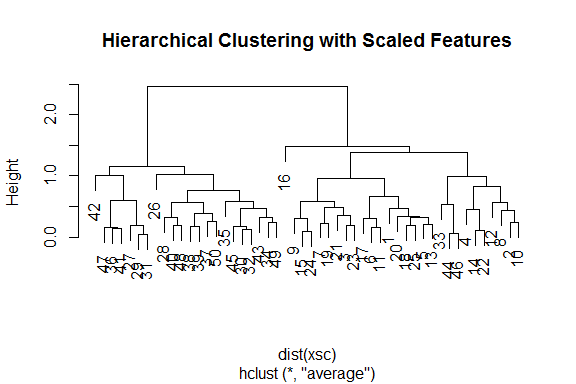
To scale the variables before performing hierarchical clustering of the observations, we can use the scale function.

**Question 8.** Apply scale to x, and then perform hierarchical clustering. Have a look at the clustering result. Is it different from the result obtained without scaling?

> xsc=scale(x)  
> plot(hclust(dist(xsc), method ="complete"), main=" Hierarchical Clustering with Scaled Features ")



> plot(hclust(dist(xsc), method ="average"), main=" Hierarchical Clustering with Scaled Features ")



> cutree(hclust(dist(xsc),method="complete"),2)

## [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2  
## [36] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2

> cutree(hclust(dist(xsc),method="average"),2)

## [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 1 2 2  
## [36] 2 2 2 2 2 2 2 2 1 2 1 2 2 2 2

*The clustering result does not seem much different from before. This was probably to be expected because the range of the two features in the original dataset is quite comparable. Note that the values of 'height' in the dendrogram are different now. This is directly due to the scaling of the variables.*

## Practical 2. NCI60 Data Example

In this second part of the practical we analyse a cancer cell line microarray dataset (NCI60), which consists of 6,830 gene expression measurements on 64 cancer cell lines. It is available via the library ISLR, which we should first install. Note that to avoid problems (as observed during previous practicals) you might better install the package using the R console only, and not within Rmarkdown.

> install.packages('ISLR');

Each cell line is labeled with a cancer type. We do not make use of the cancer types in performing clustering. However, afterwards we will check to see the extent to which these cancer types agree with the results of the clustering.

Let’s first extract labels and experimental measurements (microarray data):

> library(ISLR)  
> nci.labs=NCI60$labs  
> nci.data=NCI60$data

**Question 9.** What is the number of rows and columns of nci.data? Is this consistent with the number of genes and cell lines mentioned above?

> dim(nci.data)

## [1] 64 6830

**Question 10.** Now, examine the cancer types for the cell lines. Which cancer types are present, and what types have the largest number of samples?

> table(nci.labs)

## nci.labs  
## BREAST CNS COLON K562A-repro K562B-repro LEUKEMIA   
## 7 5 7 1 1 6   
## MCF7A-repro MCF7D-repro MELANOMA NSCLC OVARIAN PROSTATE   
## 1 1 8 9 6 2   
## RENAL UNKNOWN   
## 9 1

> nci.labs[which(table(nci.labs)==max(table(nci.labs)))]

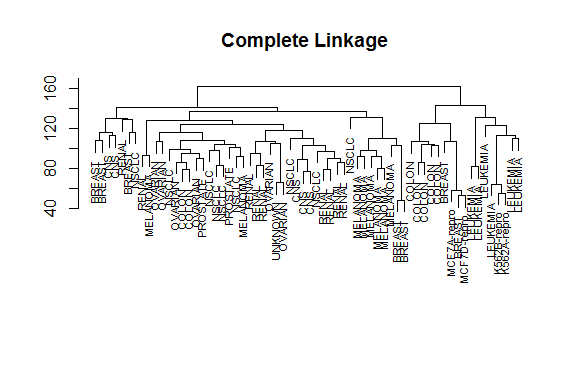
## [1] "NSCLC" "RENAL"

We now proceed to hierarchically cluster the cell lines in the NCI60 data, with the goal of finding out whether or not the observations cluster into distinct types of cancer. To begin, we standardize the variables to have mean zero and standard deviation one. As mentioned in the reader, this step is optional and should be performed only if we want each gene to be on the same scale.

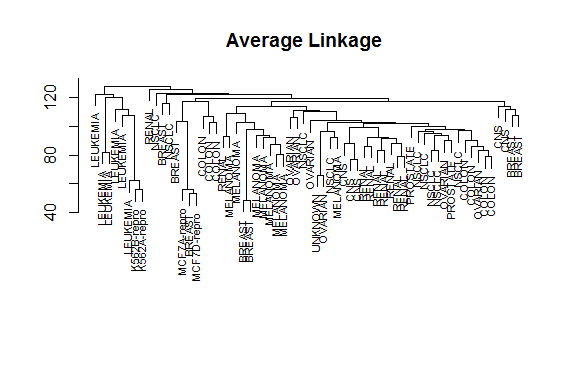
> sd.data=scale(nci.data)

**Question 11.** Now perform hierarchical clustering of the observations, based on Euclidean distance, using complete, and average linkage. Confirm in the plots that although clustering is not perfect, cell lines within a single cancer type do tend to cluster together.

> data.dist=dist(sd.data)  
> plot(hclust(data.dist), labels=nci.labs , main="Complete Linkage ", xlab="", sub="",ylab="",cex=0.7)



> plot(hclust(data.dist , method ="average"), labels=nci.labs, main="Average Linkage ", xlab="", sub="",ylab="",cex=0.7)



We will use complete linkage hierarchical clustering for the analysis that follows. We can cut the dendrogram at the height that will yield a particular number of clusters, say four:

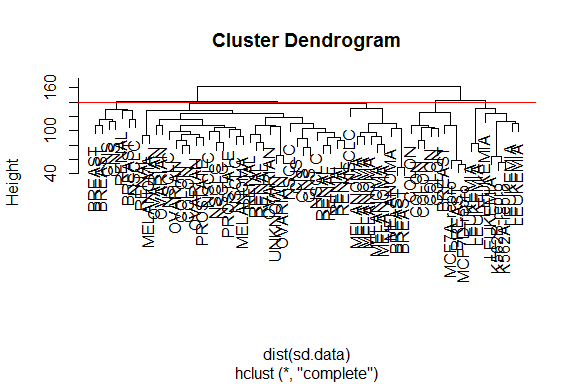
> hc.out=hclust(dist(sd.data))  
> hc.clusters =cutree (hc.out ,4)  
> table(hc.clusters ,nci.labs)

## nci.labs  
## hc.clusters BREAST CNS COLON K562A-repro K562B-repro LEUKEMIA MCF7A-repro  
## 1 2 3 2 0 0 0 0  
## 2 3 2 0 0 0 0 0  
## 3 0 0 0 1 1 6 0  
## 4 2 0 5 0 0 0 1  
## nci.labs  
## hc.clusters MCF7D-repro MELANOMA NSCLC OVARIAN PROSTATE RENAL UNKNOWN  
## 1 0 8 8 6 2 8 1  
## 2 0 0 1 0 0 1 0  
## 3 0 0 0 0 0 0 0  
## 4 1 0 0 0 0 0 0

There are some clear patterns. All the leukemia cell lines fall in cluster 3, while the breast cancer cell lines are spread out over three different clusters. This is interesting because it could indicate that there are different subtypes of this particular disease.

We can now plot the cut on the dendrogram that produces these four clusters:

> plot(hc.out , labels =nci.labs)  
> abline(h=139, col="red")



The abline function draws a straight line on top of any existing plot in R. The argument h=139 plots a horizontal line at height 139 on the dendrogram; this is the height that results in four distinct clusters.

**Question 12.** Verify in this plot that the resulting clusters are the same as the ones we obtained using cutree(hc.out,4). *Compare the labels of the items in the dendrogram with the labels listed above when using cutree.*

Remember from the reader that K-means clustering and hierarchical clustering can yield very different results. We will now find out how these NCI60 hierarchical clustering results compare to what we get if we perform K-means clustering with K = 4:

> set.seed(2)  
> km.out=kmeans(sd.data , 4, nstart =20)  
> km.clusters =km.out$cluster

**Question 13.** Use the table command to compare K-means an hierarchical clustering. What do you observe?

> table(km.clusters ,hc.clusters )

## hc.clusters  
## km.clusters 1 2 3 4  
## 1 11 0 0 9  
## 2 0 0 8 0  
## 3 9 0 0 0  
## 4 20 7 0 0

*One of the clusters (cluster 3 from hclust and cluster 2 from kmeans) is identical. However, the big cluster 1 from hclust is split into several clusters in kmeans. In kmeans, these cases are merged with clusters 2 and 4 from hclust.*

Above we clustered the samples (cell lines). In the last part of this practical, we will cluster the genes based on their expression in the various samples. In principle we could do so using all genes, but to prevent excessive computations, we will use a selected subset. We will do so by taking the 100 genes with the highest difference between their lowest and their highest expression value.

First, we use the transpose (t) function to swap rows and columns in the expression dataset. This is needed to make sure that the function dist, calculates distances between genes instead of samples; dist computes distances between the rows of a data matrix. Then, we use a cutoff on the difference between lowest and highest expression value; setting this cutoff to another value instead of 8.34 will result in a larger or a smaller set of selected genes.

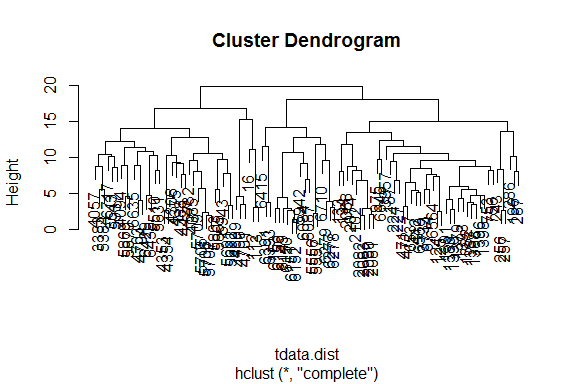
> tdata=t(nci.data)  
> dim(tdata)

## [1] 6830 64

> mymin<-apply(tdata,1,min)  
> mymax<-apply(tdata,1,max)  
> minmax=mymax-mymin  
> top100=which(minmax>8.34)  
> tdata<-tdata[top100,]

**Question 14.** Apply hierarchical clustering using complete linkage, after scaling tdata. Plot the resulting tree, and use cutree to select four clusters out of the dendrogram.

> tdata.dist=dist(scale(tdata))  
> thc.out=hclust(tdata.dist,method="complete")  
> plot(thc.out)



> cl=cutree(thc.out,4)  
> cl

## 16 112 113 124 125 133 134 187 196 224 227 243 256 257 286   
## 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2   
## 287 412 471 472 753 756 975 1379 1382 1387 1388 1389 1390 1391 1393   
## 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2   
## 1396 1398 2068 2074 2080 2081 2082 2083 2102 3383 3543 3733 3957 4057 4085   
## 2 2 2 2 2 2 2 2 2 3 3 2 2 3 3   
## 4094 4308 4353 4354 4375 4388 4426 4699 4700 4701 4703 4704 5384 5510 5556   
## 3 3 3 3 3 3 3 1 1 1 3 3 3 3 4   
## 5557 5631 5691 5692 5705 5706 5707 5709 5732 5803 5805 5869 5870 5937 5942   
## 4 3 3 3 3 3 3 3 3 3 3 3 3 3 4   
## 5943 6084 6087 6149 6150 6151 6152 6153 6156 6277 6278 6279 6391 6393 6415   
## 3 4 4 4 4 4 4 4 4 4 4 4 4 4 4   
## 6429 6430 6537 6560 6564 6635 6646 6710 6714 6717   
## 3 3 2 2 2 3 2 4 2 3

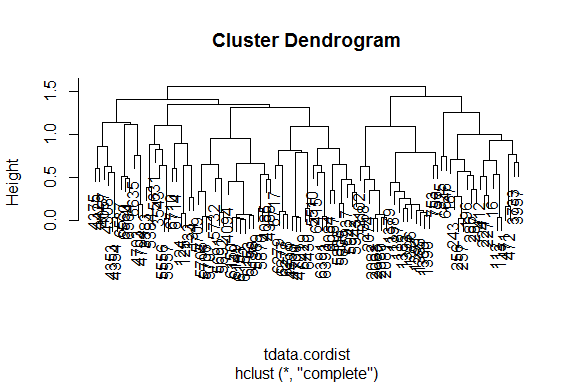
> table(cl)

## cl  
## 1 2 3 4   
## 6 43 33 18

**Question 15.** Instead of using Euclidean distance as in question 14, use a correlation based distance. Again plot the resulting tree and use cutree. Compare the resulting clusters with the ones obtained above with Euclidean distance.

Note that the function cor should not be given tdata as input because then it will calculate correlation between samples instead of genes. This is because the function cor calculates correlation between columns of the data matrix.

> tdata.cordist=as.dist(1-cor(nci.data[,top100]))  
> thc.corout=hclust(tdata.cordist,method="complete")  
> plot(thc.corout)



> clcor=cutree(thc.corout,4)  
> clcor

## 16 112 113 124 125 133 134 187 196 224 227 243 256 257 286   
## 1 1 1 2 2 2 2 1 1 1 1 1 1 1 1   
## 287 412 471 472 753 756 975 1379 1382 1387 1388 1389 1390 1391 1393   
## 1 1 1 1 3 3 1 3 3 3 3 3 3 3 3   
## 1396 1398 2068 2074 2080 2081 2082 2083 2102 3383 3543 3733 3957 4057 4085   
## 3 3 3 3 3 3 3 3 3 2 2 1 1 4 2   
## 4094 4308 4353 4354 4375 4388 4426 4699 4700 4701 4703 4704 5384 5510 5556   
## 2 4 4 4 4 2 4 2 2 2 4 4 2 2 2   
## 5557 5631 5691 5692 5705 5706 5707 5709 5732 5803 5805 5869 5870 5937 5942   
## 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2   
## 5943 6084 6087 6149 6150 6151 6152 6153 6156 6277 6278 6279 6391 6393 6415   
## 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2   
## 6429 6430 6537 6560 6564 6635 6646 6710 6714 6717   
## 2 2 4 4 4 4 1 2 2 2

> table(clcor)

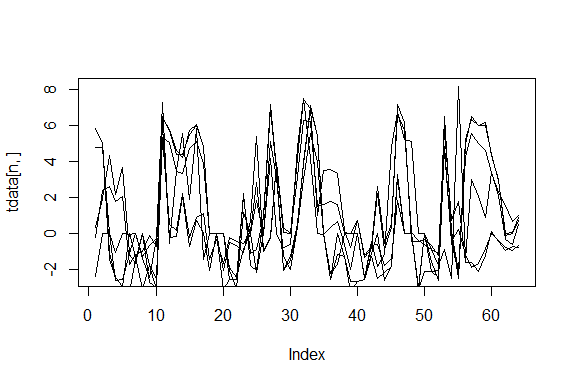
## clcor  
## 1 2 3 4   
## 19 50 19 12

> table(clcor,cl)

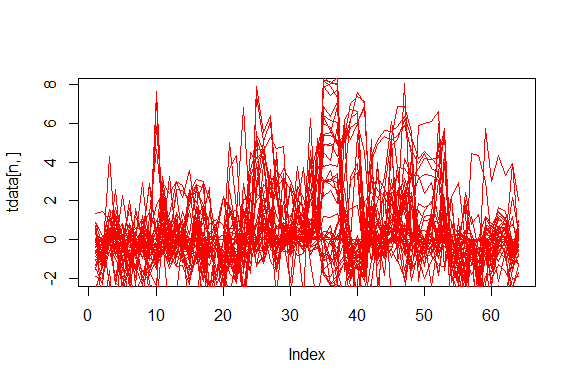
## cl  
## clcor 1 2 3 4  
## 1 3 16 0 0  
## 2 3 5 24 18  
## 3 0 19 0 0  
## 4 0 3 9 0

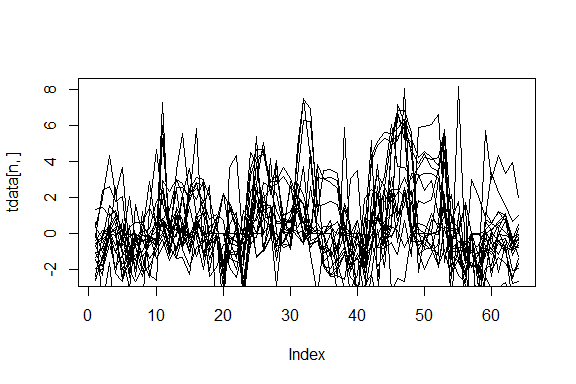
**Question 16.** Plot the expression of some of the genes in one of the clusters obtained using the Euclidean based distance. Similarly, plot the expression of some of the genes in one of the clusters obtained using the correlation based distance. What do you observe?

> n=which(cl==1)[1]  
> plot(tdata[n,],type="l")  
> for (i in 2:length(which(cl==1))) {  
+ n=which(cl==1)[i]  
+ points(tdata[n,],type="l")  
+ }

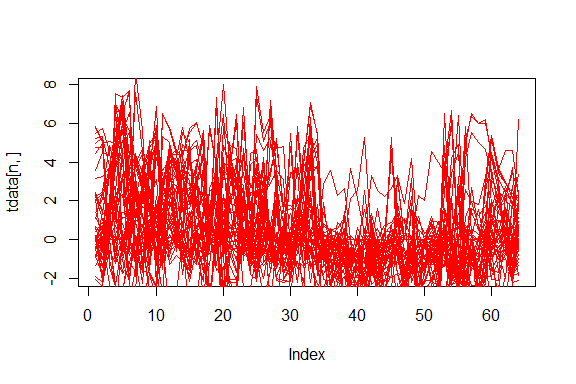


> n=which(cl==2)[1]  
> plot(tdata[n,],type="l",col="red")  
> for (i in 2:length(which(cl==2))) {  
+ n=which(cl==2)[i]  
+ points(tdata[n,],type="l",col="red")  
+ }



> n=which(clcor==1)[1]  
> plot(tdata[n,],type="l")  
> for (i in 2:length(which(clcor==1))) {  
+ n=which(clcor==1)[i]  
+ points(tdata[n,],type="l")  
+ }

> n=which(clcor==2)[1]  
> plot(tdata[n,],type="l",col="red")  
> for (i in 2:length(which(clcor==2))) {  
+ n=which(clcor==2)[i]  
+ points(tdata[n,],type="l",col="red")  
+ }



*Genes in a cluster have similar expression patterns.*