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
```

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

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

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

> km.out20$tot.withinss
```

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)
```

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)
```

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)

> cutree(hc.average , 2)
```

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 w
ith Scaled Features ")

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

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

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)
```

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

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)
```

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).

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 )
```

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)
```

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

> table(cl)
```

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

> table(clcor)

> table(clcor,cl)
```

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")
+ }
```