R dependencies

Solve dependencies with other R packages (using internet connection).

```
# set working directory
setwd("directory_of_molmed_clustering_course") # set working directory


# open link to bioconductor
source("http://bioconductor.org/biocLite.R")


# limma package for plotlines function
biocLite("limma") ; library(limma)


# packages necessary for the cluster validation part
biocLite("MASS") ; library(MASS)
biocLite("cluster") ; library(cluster)
biocLite("rgl") ; library(rgl)
biocLite("clusterSim") ; library(clusterSim)
source('DBindex.r') # include DBindex function



# packages necessary for heatmap function
biocLite("heatmap.plus") ; library(heatmap.plus)


# packages necessary for scatterplot3d function
biocLite("scatterplot3d") ; library(scatterplot3d)
```

Solve dependencies with other R packages (using local packages, for windows).

```
# set working directory

setwd("directory_of_molmed_clustering_course") # set working directory


# limma package for plotlines function


install.packages("./Rpackages/limma_3.20.3.zip", repos = NULL, type =
"source") ; library(limma)


# packages necessary for the cluster validation part

install.packages("./Rpackages/MASS_7.3-33.zip", repos = NULL, type =
"source") ; library(MASS)

install.packages("./Rpackages/cluster_1.15.2.zip", repos = NULL, type =
"source") ; library(cluster)

install.packages("./Rpackages/rgl_0.93.996.zip", repos = NULL, type =
"source")  ; library(rgl)

install.packages("./Rpackages/clusterSim_0.43-4.zip", repos = NULL, type =
"source")  ; library(clusterSim)

source('DBindex.r') # include DBindex function




# packages necessary for heatmap function

install.packages("./Rpackages/heatmap.plus_1.3.zip", repos = NULL, type =
"source") ; library(heatmap.plus)

# packages necessary for scatterplot3d function

install.packages("./Rpackages/scatterplot3d_0.3-35.zip", repos = NULL, type
= "source") ; library(scatterplot3d)
```

Clustering & PCA

Clustering is the process of **grouping a set of data objects** into multiple groups or clusters so that objects within a cluster **have high similarity**. It is used as a method to see if natural groupings are present in the data. If these groupings do emerge, these may be named and their properties summarized. All the different clustering methods can produce a partitioning of the dataset. However, different methods will often yield different groupings since each implicitly imposes a structure on the data.

1 Hierarchical clustering

Hierarchical clustering groups data over a variety of scales by creating a cluster tree or *dendrogram*. The tree is not a single set of clusters, but rather a multilevel hierarchy, where clusters at one level are joined as clusters at the next level. This allows you to decide the level or scale of clustering that is most appropriate for your application.

Hierarchical clustering is based on three major steps:

1. Finding the **similarity** or **dissimilarity** between every pair of objects in the data set. In this step, you calculate the distance between objects.
2. Grouping the objects into a binary, hierarchical cluster tree (**dendrogram**). In this step, you link pairs of objects that are in close proximity (have a high similarity) using different linkage methods. The **linkage** method uses the distance information generated in step 1 to determine the similarity of objects to each other. As objects are paired into binary clusters, the newly formed clusters are grouped into larger clusters until a hierarchical tree is formed
3. Determine where to **cut** the hierarchical tree into clusters. In this step the obtained dendrogram is **analyzed** in order to decide which cutting level best suits the data.

1.1 Similarity-Dissimilarity

A similarity measure for two objects, *j*, and *i* will be large when the objects are very similar. On the other hand, a **dis**similarity measure will be large when the two objects are very **dis**similar and small when they are similar, just like a distance measure between objects. Several distance measures are commonly used for computing the dissimilarity of objects described by numeric attributes. Below we describe some examples.

Let $i = (x_{i1}, x_{i2}, \dots, x_{in},)$ and $j = (x_{j1}, x_{j2}, \dots, x_{jn},)$ be two objects described by *n* numeric attributes.

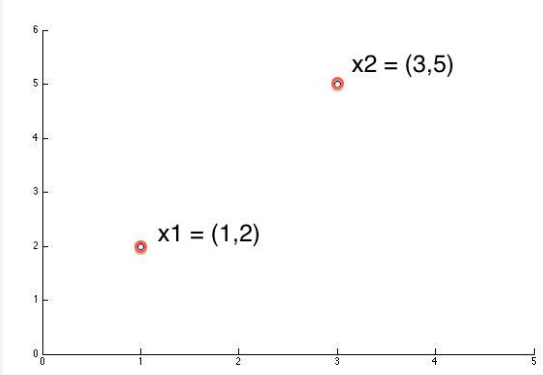
- The **Euclidean** distance between the two objects is defined as
  $d(i,j) = \sqrt{(x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2 + \cdots + (x_{in} - x_{jn})^2}$
- The **Minkowski** distance is a generalization of the Euclidean distance:
  $d(i,j) = \sqrt[h]{|x_{i1} - x_{j1}|^h + |x_{i2} - x_{j2}|^h + \cdots + |x_{in} - x_{jn}|^h}$

- To base a distance measure on the **correlation** between the objects we need to convert the similarity measure into a dissimilarity measure (by negation):

$$d(i,j) = 1 - (|x_{i1} * x_{j1}| + |x_{i2} * x_{j2}| + \cdots + |x_{in} * x_{jn}|)$$

Exercise 1.

Given x1 and x2 in the figure below, what is the Euclidean Distance between x1 and x2?



a) 3.61 b) 25 c) 5

The function to calculate distances between objects in *R* is *dist*. The *dist* function calculates the distance between object 1 and object 2, object 1 and object 3, and so on until the distances between all the pairs have been calculated.

Let's use the following toy dataset as an example:

object	feature 1	feature 2
A	1	2
B	2,5	4,5
C	2	2
D	4	1,5
E	4	2,5

```
data = cbind(c(1,2.5,2,4,4),c(2,4.5,2,1.5,2.5)) # Make small data matrix

plot(data) # Plot to screen

labs = c('A','B','C','D','E') # Define labels

rownames(data)<-labs # Set new labels

text(data+0.05, labs) # Plot the labels
```

Use the following code to determine the Euclidean distance between A=(1,2) and C=(2,2).

```
d.eucl = dist(data, method = "euclidean") # the function that calculates
the Euclidean distance

d.eucl # this will show the following distance matrix

          A          B          C          D
B  2.915476
C  1.000000  2.549510
D  3.041381  3.354102  2.061553
E  3.041381  2.500000  2.061553  1.000000

# The Euclidean distance between (A,C) can be read, and is 1.
```

Check "? *dist*" to see all the distances it can compute, or the website http://stat.ethz.ch/R-manual/R-patched/library/stats/html/dist.html

1.2 Building the dendrogram – Linkage methods

Once the dissimilarity between objects in the data set has been computed, you can determine how objects in the data set should be grouped into clusters by choosing one of the linkage methods. *R* function *hclust* takes the distance information generated by *dist* and links pairs of objects that are close together into binary clusters. The *linkage* function then links these newly formed clusters to each other and to other objects to create bigger clusters until all objects in the original data set are linked together in a hierarchical tree.

1.2.1 Single link method

The distance between two groups is defined as the distance between their two closest members.

```
sl.eucl = hclust(d.eucl,method="single")

plot(as.dendrogram(sl.eucl)) # plot dendrogram
```

1.2.2 Complete link method

This method defines the distance between two groups as the distance between their two farthest-apart members. This method usually yields clusters that are well separated and compact.

```
cl.eucl = hclust(d.eucl,method="complete")

plot(as.dendrogram(cl.eucl)) # plot dendrogram
```

1.2.3 Average link method

This algorithm defines the distance between groups as the average distance between each of the members, weighted so that the two groups have an equal influence on the final result.

```
al.eucl = hclust(d.eucl,method="average")

plot(as.dendrogram(al.eucl)) # plot dendrogram
```

Exercise 2.

Inspect the dendrogram with respect to the distances calculated earlier. Can you reconstruct the distances between the data points from dendrogram?

Exercise 3.

Suppose you want to group the objects of the toy example into 2 clusters. Depending on the linkage methods you use, how are the objects grouped?

Linkage	**Answer**	**cluster 1**	**cluster 2**
Single	a)	A C D E	B
	b)	A B	C D E
	c)	A C	D E
Complete	a)	B C D	A E
	b)	A C D E	B
	c)	A C B	D E
Average	a)	A C	B D E
	b)	A C D E	B
	c)	A C	B D E

Hierarchical clustering produces a dendrogram. To give the desired number of clusters, the tree can be cut at a desired horizontal level. The number of vertical stems of the dendrogram intersected by the horizontal line, corresponds to the number of clusters. A scatterplot of the smaples, colour-coded by cluster membership, reveals the clustering (note that when there are more than two features in the dataset, the clustering is performed taking all features into account, but that only the first two features are displayed).

Load the *triclust* dataset and cluster by means of euclidean distance and complete linkage.

```
data = read.delim("triclust.txt", header=F, dec=",")

d = dist(as.matrix(data), method = "euclidean") # Euclidean distance
```

```
hc = hclust(d,method="complete") # hierarchical clustering, complete linage

plot(hc) # plot dendogram

cl <- cutree(hc, 3) #cut the dendrogram at two clusters

cl #This indicates cluster membership

plot(data,col=cl) # plot the points, with cluster membership as colour
```

1.2.4 Hierarchical clustering of the "Messy data"

Load the messy data set:"messy.txt". This is not a real gene expression matrix, but simply a data set of 300 points in a two dimensional space, e.g. the measurements of 300 genes using two microarrays. The purpose of this exercise is to get you acquainted with clustering gain insight in the different parameters that you can set.

```
data = read.delim("messy.txt", header=T, dec=".")

plot(data[,2:3]) # plot the data
```

This plot visualizes the 300 genes as 300 circles. In this 2D scatterplot, `Measurement 1' (the first microarray) is indicated on the X-axis (horizontal) while `Measurement 2' (the second microarray) is indicated on the Y-axis (vertical).

```
labs= data[,1] # define labels

rownames(data)<-labs # set new labels

text(data[,2],data[,3],labs) # plot the labels
```

Exercise 4.

The *messy* dataset shows two noisy clusters that are almost overlapping. Which linkage works best to obtain the two underlying noisy clusters?

a) single b) complete c) average

Exercise 5.

What is the optimal number of clusters based on the scatter plot when choosing complete linkage?

a) 2 b) 10 c) 3

Exercise 6.

If you cut the euclidean distance dendogram at 1.5. How many clusters do you obtain?

a) 3 b) 4 c) 5 d) 8

```
d = dist(as.matrix(data[,2:3]), method = "euclidean") # Euclidean distance

hc = hclust(d, method="complete") # hierarchical clustering, complete
linkage

cl = cutree(hc, , 1.5) # cut the dendrogram at distance 1.5

plot(data[,2:3],col=cl) # Select only the points that fall in the two
largest clusters
```

Exercise 7.

Explain why selecting two clusters does not work to obtain the two underlying noisy clusters.

Exercise 8.

What would you do to obtain the core of the two underlying clusters, given the single linkage result?

Code for removing outliers:

```
d = dist(as.matrix(data[,2:3]), method = "euclidean") # Euclidean distance

hc = hclust(d, method="single") # hierarchical clustering, complete linkage

plot(as.dendrogram(hc)) # plot dendrogram

cl = cutree(hc, 20) # cut the dendrogram at e.g. 20 clusters

plot(data[,2:3],col=cl) # Select only the points that fall in the two
largest clusters

# inspect the clustering (cl); this shows two large clusters (1 and 2) and
the rest small clusters of “outliers”

# select the two largest clusters (numbers from inspection or next line of
code)

ind=unique(cl) ; m=NULL ; for (i in 1:length(ind)) {
m=cbind(m,sum(ind[i]==cl)) ; s=sort(m,decreasing=TRUE) } ;
ind=cbind(ind[m==s[1]],ind[m==s[2]]) # ind stores the two indices of the
two largest clusters

data.new=data[cl==ind[1] | cl==ind[2],] # Select only the points that fall
in the two largest clusters

plot(data.new[,2:3]) # plot the data

# this new data can then be clustered, and shows more clearly 2 clusters

d.new = dist(as.matrix(data.new[,2:3]), method = "euclidean")
```

```
hc.new = hclust(d.new,method="complete")

plot(as.dendrogram(hc.new))

cl.new <- cutree(hc.new,2)

plot(data.new[,2:3],col=cl.new)
```

Exercise 9.

Load the *cigars* dataset. Which linkage works best for this dataset?

a) single b) complete c) average

```
data = read.delim("cigars.txt", header=T, dec=".")
```

1.2.5 Hierarchical clustering of the "Easy data"

Create a dendrogram of the "*easy*" data set using hierarchical clustering using Euclidean distance and complete linkage (again this is not a real gene expression matrix).

```
data = read.delim("easy.txt", header=T, dec=".")

d = dist(as.matrix(data), method = "euclidean") # Euclidean distance

hc = hclust(d,method="complete") # hierarchical clustering, complete linage

plot(as.dendrogram(hc)) # plot dendogram

rect(0,-1, 49,1, border = "orange", lwd=2) # Highlight cluster 1
```

Exercise 10.

What is the optimal number of clusters and how is this visible in the dendrogram?

Cut the tree at 3 clusters.

```
cl <- cutree(hc, 3) #cut the dendrogram at two clusters

cl #This indicates cluster membership

plot(data[,2:3],col=cl) # plot the points, with cluster membership as
colour
```

Exercise 11.

Infer the maximal distance between the genes in the cluster positioned at (0,0) from the dendrogram (it is the left most highlighted cluster in de dendrogram).

```
plot(hc) # plot dendogram

plot(as.dendrogram(hc), xlim = c(0,49), ylim = c(-1,1)) # zoom in cluster 1
```

Exercise 12.

Which two genes are employed to compute this distance?

Exercise 13.

Check this distance by calculating the distance between those two genes (using the *dist* function).

1.2.6 Pearson and Mixed correlation

Load the "*distance*" data set. This dataset contains 5 signals especially designed to show the effects of the Euclidean and Correlation-based distance measures when using hierarchical clustering.

```
data = read.delim("distance.txt")

data = data[,-1] # get only the data part (exclude names)

colors=rainbow(5) # define 5 colors

plotlines(data,col=colors) # plot the data as lines over the different
dimensions, color the 5 signals differently
```

The plot shows each signal independently over the 10 measurements.

Exercise 14.

Which two signals will be first clustered when using the **Euclidean** distance as dissimilarity measure (thus what will be the four clusters).

Check the result.

```
d_eucl = dist(data, method = "euclidean") # Euclidean distance

hc.eucl = hclust(d_eucl,method="complete") # Cluster, Euclidean

cl <- cutree(hc.eucl, 4) # cut the Euclidean-based dendrogram at 4 clusters

plotlines(data,col=colors[cl]) # plot the signals and color according to
the cluster they belong to
```

Exercise 15.

In which 3 clusters will the 5 signals be clustered?

Check the result.

```
cl <- cutree(hc.eucl, 3) #cut the Euclidean-based dendrogram at 3 clusters

plotlines(data,col=colors[cl])
```

Exercise 16.

Now we switch the dissimilarity measure to the **correlation**-based distance. Which two signals will now be first clustered (thus what will be the four clusters).

Check the result.

```
d_corr = as.dist(1-cor(t(data))) # distance based on correlation

hc.corr = hclust(d_corr,method="complete") # Cluster, correlation-based

cl = cutree(hc.corr, 4) #cut the dendrogram at 4 clusters

plotlines(data,col=colors[cl]) #This function requires the "limma" library
```

Exercise 17.

In which 3 clusters will the 5 signals be clustered?

Check the result.

```
cl = cutree(hc.corr, 3) # cut the correlation-based dendrogram at 3
clusters

plotlines(data,col=colors[cl])
```

Exercise 18.

Now we switch the dissimilarity measure to the **mixed-correlation**-based distance. Which two signals will now be first clustered (thus what will be the four clusters).

Check the result.

```
d_mixcorr = as.dist(1-abs(cor(t(data[,-1])))) # distance based on
correlation

hc_mixcorr = hclust(d_mixcorr,method="complete") # Cluster, mix-correlation

cl <- cutree(hc_mixcorr, 4) #cut the dendrogram at 4 clusters

plotlines(data,col=colors[cl])
```

Exercise 19.

In which 3 clusters will the 5 signals be clustered?

Check the result.

```
cl <- cutree(hc_mixcorr, 3) # cut the mix-correlation based dendrogram at 5
clusters

plotlines(data,col=colors[cl])
```

2 K-means clustering

k-means clustering is a partitioning method. This method partitions data into *k* mutually exclusive clusters, and returns the index of the cluster to which it has assigned each observation. Unlike hierarchical clustering, k-means clustering operates on actual observations (rather than the larger set of dissimilarity measures), and creates a single level of clusters. The Hierarchic clustering method starts from the bottom, and combines observations to clusters. At higher levels it can only fuse existing clusters to others, but it cannot break old groups to build better ones.

2.1 K-means Algorithm

The k-means algorithm defines the centroid of a cluster as the mean value of the points within the cluster. First, it randomly selects *k* of the objects in the dataset, each of which initially represents a cluster mean or center. For each of the remaining objects, an object is assigned to the cluster to which it is most similar, based on the Euclidean distance between the object and the cluster mean. The k-mean algorithm then iteratively improves the within-cluster variation. For each cluster, it computes the new mean using the objects assigned to the cluster in the previews iteration. All the objects are then reassigned using the updated means as the new cluster centers. The iterations continue until the assignment is stable, that is the clusters formed in the current round are the same as those formed in the previous round.

k: the number of clusters

D: a dataset containing *n* objects

1. arbitrarily choose k objects from *D* as the initial cluster centers;
2. **repeat**
    3. (re)assign each object to the cluster to which the object is the most similar, based on the mean value of the objects in the cluster;
    4. update the cluster means;
5. **until** no change;

The k-means method is not guaranteed to converge to the global optimum and often terminates at a local optimum. The results may depend on the initial random selection of cluster centers. To obtain good results in practice, it is common to run the k-mean algorithm multiple times with different initial cluster centers.

The k-means method can be applied only when the mean of a set of objects is defined.

2.2 K-means in R

A function in *R* is provided to perform k-means clustering: *kmeans.* For more information type "? kmeans" or go the website: http://stat.ethz.ch/R-manual/R-devel/library/stats/html/kmeans.html

Exercise 20.

Import the *messy* data, apply a kmeans clustering and plot data.

```
data = read.delim("messy.txt", header=T, dec=".") #read data

data = data[,2:3] # get only data

cl <- kmeans(data, 7) # kmeans clustering starting with 7 centers

plot(data, col = cl$cluster) #Plot data

points(cl$centers, col = 1:2, pch = 8, cex = 2) #Show the centers
```

Exercise 21.

Run the *kmeans* a few times and see what happens to the clustering and how the random initialization affects the final clustering. Do you always obtain the same clustering, i.e. with one prototype per cluster? Try to explain why this happens.

Inspect the different update steps of the kmeans algorithm. Run the following code.

```
SEED = 6 # set seed for random initialization

WAITTIME = 1 # waiting time between updates (enlarge in case it goes to
fast)

for (i in 1:10) { set.seed(SEED) ; cl<-kmeans(data,7,iter.max=i) ;
plot(data, col = cl$cluster) ; points(cl$centers, col = 1:2, pch = 8, cex =
2) ; Sys.sleep(WAITTIME)}

# optionally you can use "par(mfrow=c(2,5))" before the for loop so that
all results are put in the same figure
```

Exercise 22.

Change the SEED (different positive number) and see how the initialization and updates change.

Exercise 23.

How can we overcome the local minimum problem of kmeans?

```
# Finding optimum over a number of random initialization in R is done by
setting the nstart argument of kmeans operator

cl <- kmeans(data, 7, nstart = 1000) # kmeans with 7 centers

plot(data, col = cl$cluster) #Plot data

points(cl$centers, col = 1:2, pch = 8, cex = 2) #Show the centers
```

Exercise 24.

Read the *hall* dataset that consists of 14 clusters. Try to cluster this data with kmeans (play around with the initialization and restarts). Compare the kmeans clustering also with a hierarchical clustering of the data.

```
data = read.delim("hall.txt", header=T, dec=".") #read data

data = data[,2:3] # get only data


# kmeans clustering

cl <- kmeans(data, 14) # kmeans clustering starting with 7 centers

plot(data, col = cl$cluster) #Plot data

points(cl$centers, col = 1:2, pch = 8, cex = 2) #Show the centers


# hierarchical clustering

d = dist(as.matrix(data), method = "euclidean") # Euclidean distance

hc = hclust(d,method="complete") # hierarchical clustering, complete linage

cl <- cutree(hc, 14) # cut the Euclidean-based dendrogram at 4 clusters

plot(data,col=cl) # plot the points, with cluster membership as colour
```

3 Cluster validation: Davies-Bouldin index

The Davies-Bouldin criterion is based on a ratio of within-cluster and between-cluster distances. The Davies-Bouldin index is defined as

$$DB = \frac{1}{k}\sum_{i}^{k} max_{j \neq i}\{D_{i,j}\}$$

where D_{ij} is the within-to-between cluster distance ratio for the *i*th and *j*th clusters.

In mathematical terms,

$$D_{i,j} = \frac{\bar{d}_i + \bar{d}_j}{d_{i,j}}$$

$\bar{d}_i$ is the avarege distance between each point in the *i*th cluster and the centroid of the *i*th cluster.

$\bar{d}_j$ is the avarege distance between each point in the *j*th cluster and the centroid of the *j*th cluster.

$d_{i,j}$ is the Euclidean distance between the centroids of the *i*th and *j*th clusters.

The maximum value of D_{ij} represents the worst-case within-to-between cluster ratio for cluster *i*. The optimal clustering solution has the smallest Davies-Bouldin index value.

In *R*, more information can be found at http://artax.karlin.mff.cuni.cz/r-help/library/clusterSim/html/index.DB.html or http://cran.r-project.org/web/packages/clusterSim/clusterSim.pdf

Exercise 25.

Evaluate the optimal number of clusters using the Davies-Bouldin clustering evaluation criterion on the *hall* dataset and using a hierarchical clustering.

```
data = read.delim("hall.txt", header=T, dec=".") #read data

data = data[,2:3] # get only data

d.eucl = dist(data, method="euclidean") # compute distance

hc.eucl = hclust(d.eucl,method="complete")   # hierarchical clustering

# Compute Davies-Bouldin index for min_nc to max_nc clusters

DBindex(data, d.eucl, hc.eucl, 2, 50) # 2 and 50 define lower and upper
bound of number of clusters tested
```

Exercise 26.

Evaluate the optimal number of clusters using the Davies-Bouldin clustering evaluation criterion on the *messy* dataset using hierarchical clustering with *complete linkage*.

Do the results change if you use different linkage type or distance measure?

4 Leukemia dataset

The leukemia data set is part of the Golub dataset and contains two types of leukemia (ALL and AML) patients. Each microarray, measured on a specific patient, represents a row in the data; each column represents the transcript levels of a specific gene over all different patients. Since this dataset contains two distinct groups of patients (ALL and AML), we are interested in the results of clustering this data with respect to the patients (patients with a similar expression profile in the same group).

Exercise 27.

Read leukemia data set. The *image* function lets you display the whole matrix. How many genes are measured for how many microarrays. (hint: inspect also the data object using

Exercise 28.

Read leukemia data set. The *image* function lets you display the whole matrix. How many genes are measured for how many microarrays. (hint: inspect also the data object using the *dim* and *summary* functions)

```
data = read.delim("leukemia.txt", header=T, dec=".") # read leukemia data

data.only = data[,-1] # store data only

image(as.matrix(data.only)) # Show heatmap
```

Exercise 29.

Inspect how the patients are clustered. (Hint: use the *dist* and *hclust* functions)

Instead of viewing the gene expression matrix as an image one can also look at it as a set of signals. Such a signal represents the expression profile of one gene over the different patients.

```
d = dist(data.only, method="euclidean") # eucludean distance patients

hc = hclust(d, method="complete") # complete linkage clustering

plot(as.dendrogram(hc)) # Plot dendrogram

cl = cutree(hc.corr, 2) #cut the dendrogram at 2 patient clusters

colors=rainbow(2) # define 10 colors
```

```
# Plot signal from cluster 1

plotlines(data.only[cl==1,],col=colors[1])

# make new window / or / plot in same window (par(new=TRUE) command)

par(new=TRUE)

# Plot signal from cluster 2

plotlines(data.only[cl==2,],col=colors[2])
```

Exercise 30.

Which group is associated with upregulation?

R's *heatmap* function automatically clusters both the rows (patients) and columns (genes) of a matrix. Defaults that it uses are the Euclidean distance function and Complete linkage.

```
heatmap(as.matrix(data.only),labRow=data[,1], scale="none")
```

Exercise 31.

Which cluster corresponds best to which patient group?

Exercise 32.

Which patients are incorrectly clustered in the opposite group of patients?

Exercise 33.

Which genes are uninformative with respect to the class labels? Name a few

Exercise 34.

Which genes are uninformative with respect to the class labels? Name a few

One can also change the way that the rows and columns are clustered (i.e. change the choice for the distance and linkage methods).

```
# choose clustering for rows
```

```
row.dist= as.dist(1-cor(t(data.only))) # distance based on correlation
         # alternative: row.dist = dist(data.only, method="manhattan")
row.clust = hclust(row.dist, method="complete") # complete linkage
# choose clustering for colums; note we need to transpose the data !
col.dist = dist(t(data.only), method="euclidean") # euclidean distance
col.clust = hclust(col.dist, method="average") # average linkage

# show heatmap according to selected row and column clustering
heatmap.plus(as.matrix(data.only), Rowv=as.dendrogram(row.clust),
Colv=as.dendrogram(col.clust), labRow=data[,1], scale="none")
```

Exercise 35.

Which two patients have the most similar expression profile? (*Hint*: use single linkage)

5 Principal Component Analysis (PCA)

Principal component analysis (PCA) involves a mathematical procedure that transforms a number of (possibly) correlated variables into a (smaller) number of uncorrelated variables called *principal components*. The first principal component accounts for as much of the variability in the data as possible, and each succeeding component accounts for as much of the remaining variability as possible.

Objectives of principal component analysis

- To discover or to reduce the dimensionality of the data set.
- To identify new meaningful underlying variables.

Exercise 36.

Read leukemia data set and perform a PCA.

```
data = read.delim("leukemia.txt", header=T, dec=".")
pca <- prcomp(data[,-1], scale=TRUE) # do PCA
summary(pca) # this gives a summary of the PCA components
plot(pca) # this plots the variances covered by each of the PCA components
```

Exercise 37.

How much variance is covered by the first three components?

Exercise 38.

Plot the first two principle components and why are the first principle components more important than the last ones?

```
plot(pca$x[,1], pca$x[,2], col=(data$Leukemia.type=="AML")+1) # plots
samples in space spanned by PC1 and PC2, and colors the points according to
the leukemia type (AML or ALL)
```

Exercise 39.

Plot the first 10^{th} and 11^{th} principle components. Why do we not see differences between the two leukemia classes in this plot?

```
plot(pca$x[,10], pca$x[,11], col=(data$Leukemia.type=="AML")+1) # plots
samples in space spanned by PC10 and PC11, and colors the points according
to the leukemia type (AML or ALL)
```

Exercise 40.

Plot the loadings of for the first principal component. What can you conclude from that?

```
pc1loadings <- pca$rotation[,1] # get loadings first PC component

plot(pc1loadings) # Show the first principal component
```

Exercise 41.

Plot the first three principle components

```
scatterplot3d(pca$x[,1], pca$x[,2], pca$x[,3]) # plot first 3 PC components
```

6 ADDITIONAL DATASET

The DMD data described by Pescatori et al. (FASEB J 2007, 21:1210-26) consists of rma-normalized Affymetrix hg133a expression data of 33 muscle samples from 2 different groups: 19 Duchenne muscular dystrophy (DMD) patients and 14 controls. In principle, all Affymetrix probesets could be used (22K), but here we use the significantly detected probe sets between the two groups (877 probe sets) to shorten the calculation time.

```
data = read.delim("DMD_data_significant.txt")

data.only = data.matrix(data[,6:ncol(data)])
```

Exercise 42.

Cluster the DMD data. Examine the patient clustering (do you find back the two groups). Also examine how the genes are clustered. What is a good distance measure for the genes? (Hint: make use of the *heatmap* function)