

Clustering – Exercises

This exercise introduces some clustering methods available in R and Bioconductor. For this exercise, you'll need the kidney dataset: Go to menu File, and select Change Dir. The kidney dataset is under data-folder on your desktop.

1. Reading the prenormalized data

Read in the prenormalized Spellman's yeast dataset:

```
> d<-read.table("combined.txt", sep="\t", header=T, row.names=1)
```

We want only the cdc15 data, so take only those columns from the data:

```
> names(d)
> da<-data.frame(d[26:49])
```

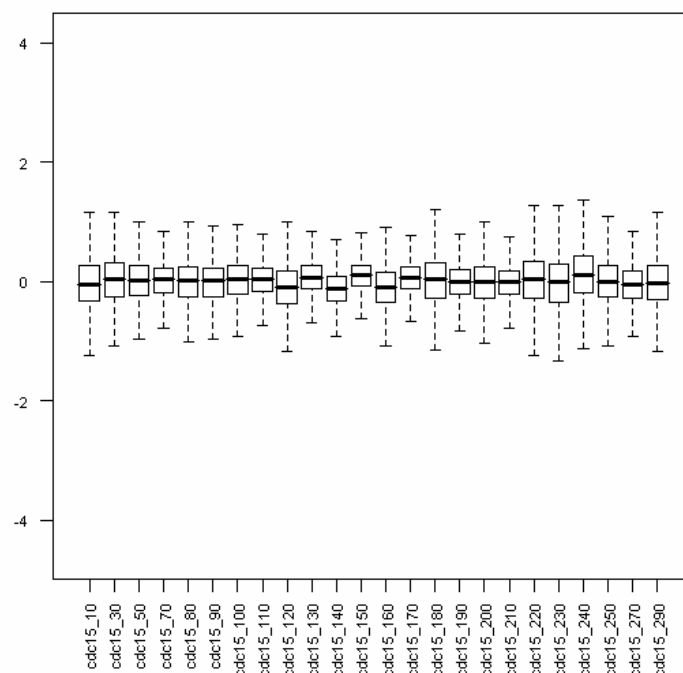
Remove missing values from the data:

```
> dat<-na.omit(da)
```

2. Make some normalization checks:

Make a boxplot of normalized chips. Note the effect of normalization:

```
> boxplot(dat, outline=F, las=2, cex.axis=0.7)
```



3. Filter the genelist by standard deviation

Select only the genes that are among the 0.3% of the highest standard deviations.

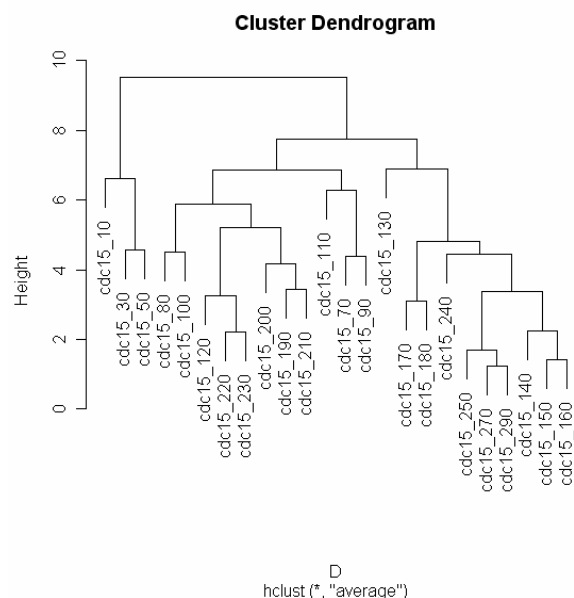
```
> library(genefilter)
> percentage<-c(0.997)
> # This calculates a genewise standard deviation
> sds<-rowSds(dat)
> # Select 0.3% of the genes having the highest SD
> sel<-(sds>quantile(sds,percentage))
> set<-dat[sel, ]
```

How many genes are left after filtering?

4. Visualizing a hierarchical sample tree using euclidian distance

Visualize the dependencies between samples using a hierarchical clustering method (euclidian distances and average linkage). The first step is to calculate the distances between samples using a selected distance method. A tree is drawn from these distances using a selected drawing method.

```
> # Distances between observations are calculated using euclidian
distances
> distmeth<-c("euclidian")
> # To calculate a tree for sample, the data matrix need to be
transposed
> D<-dist(t(set), method=distmeth)
> # Tree is formed using average distance between clusters
> treemeth<-c("average")
> hc<-hclust(D, method=treemeth)
> plot(hc)
```

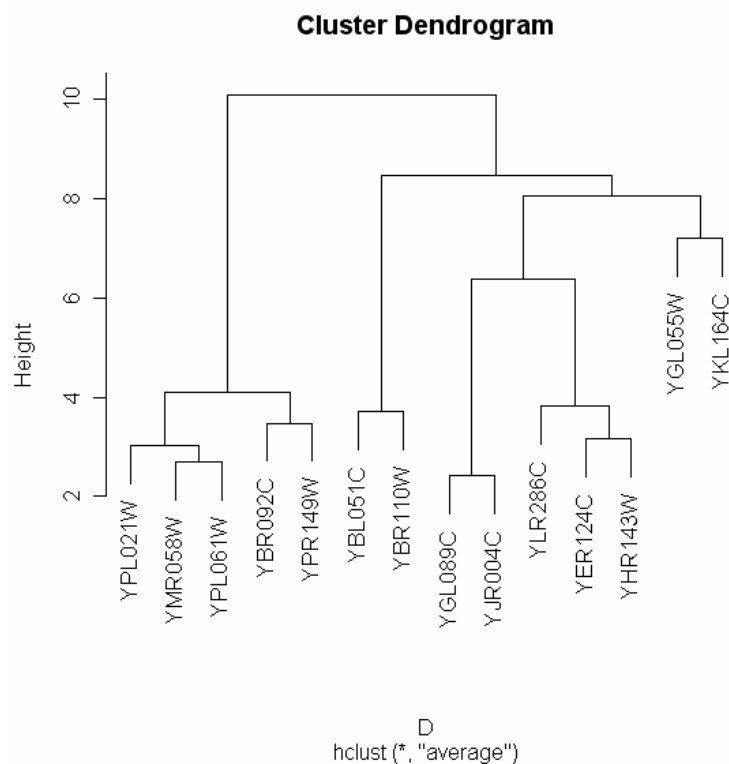


Draw a tree using single and complete linkage. Do the trees differ from each other?

5. Visualizing a hierarchical gene tree using euclidian distance

Visualize the dependencies between samples using a hierarchical clustering method (euclidian distances and average linkage). Here gene names are used as labels on the leaves.

```
> distmeth<-c("euclidian")
> D<-dist(set, method=distmeth)
> treemeth<-c("average")
> hc<-hclust(D, method=treemeth)
> plot(hc, labels=row.names(set))
```

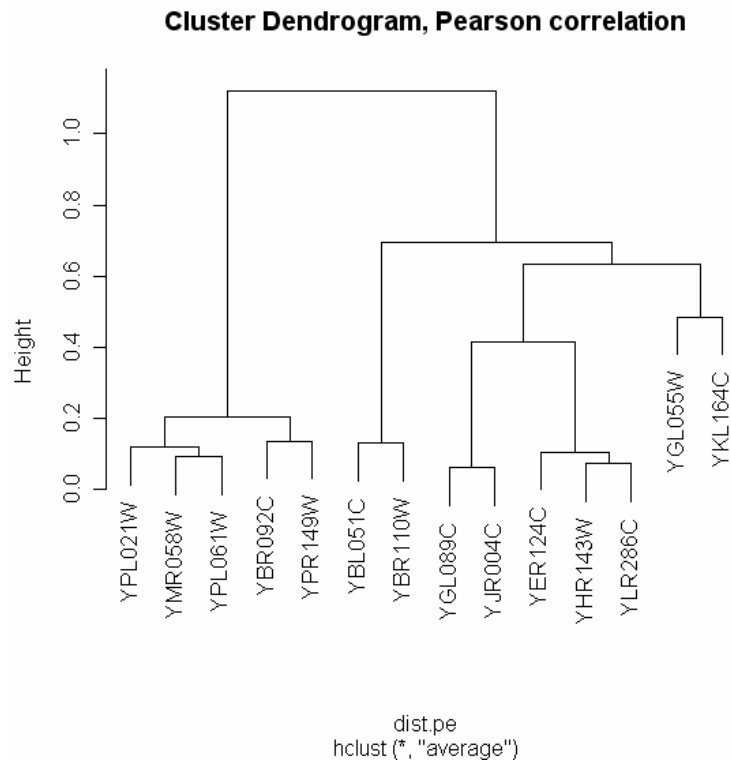


6. Visualizing a hierarchical gene tree using correlation

We used euclidian distance in the examples above. More often gene expression profiles are clustered using correlation coefficients. To do this you need to calculate correlation between the genes, and reformat the correlation matrix into an object containing distances. After that the tree drawing proceeds normally.

```
> cor.pe<-cor(t(set), method=c("pearson"))
> cor.sp<-cor(t(set), method=c("spearman"))
> dist.pe<- as.dist(1-cor.pe)
> dist.sp<- as.dist(1-cor.sp)
> hc<-hclust(dist.pe, method=treemeth)
> plot(hc, labels=row.names(set), main="Cluster Dendrogram,  
Pearson correlation")
```

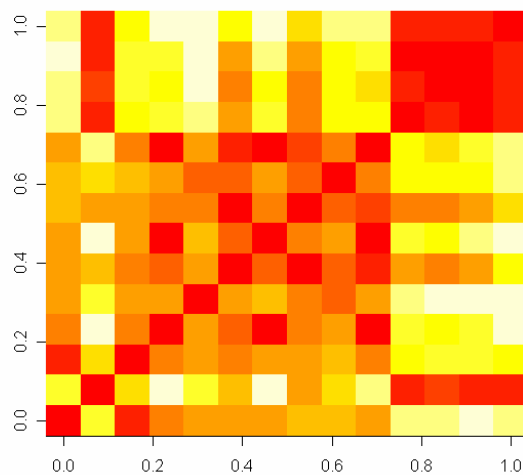
Draw the same tree using Spearman correlation. Do the results differ?



7. Visualizing the correlation matrix

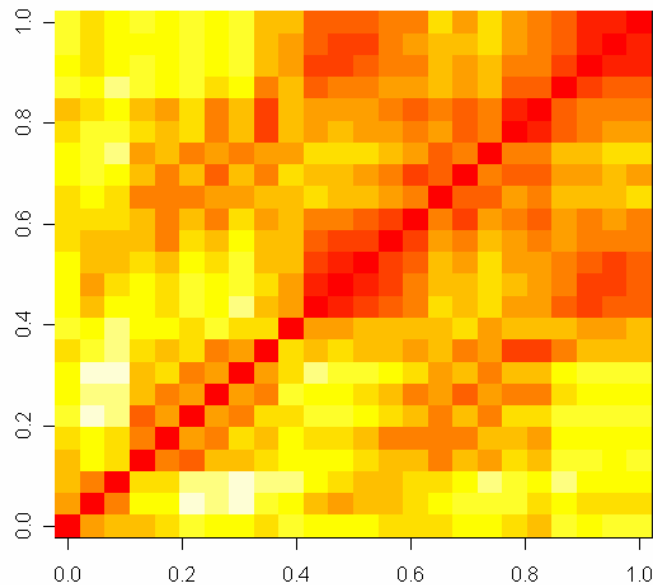
Visualize the correlation matrix between the genes.

```
> # Dists are converted to a matrix, otherwise this doesn't work  
> image(as.matrix(dist.pe))
```



Similarly, visualize the distance matrix between samples:

```
> D<-dist(t(set), method=distmeth)
> image(as.matrix(D))
```



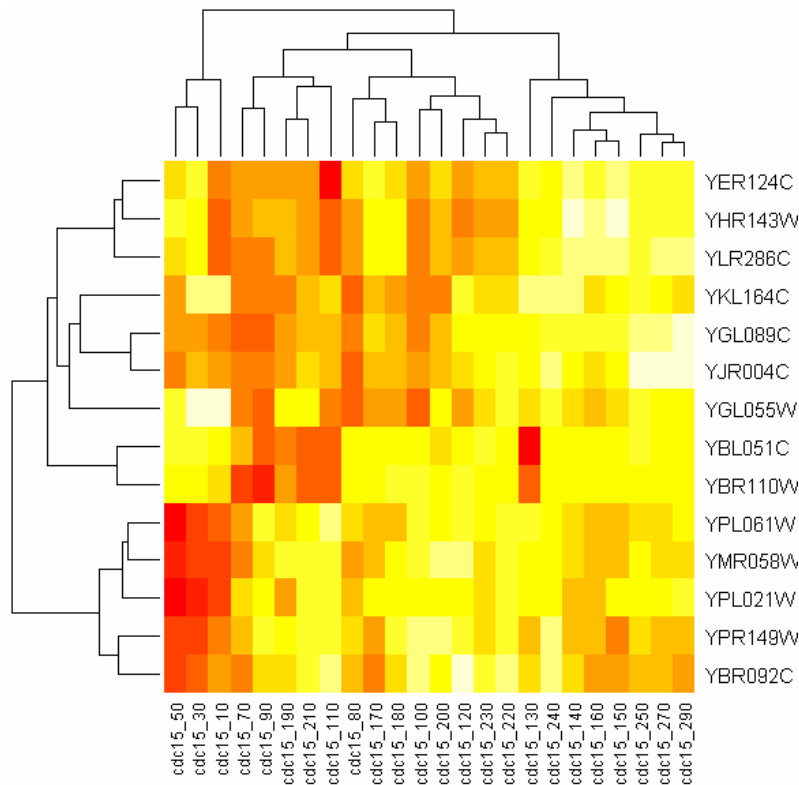
These images give you a view to the distance matrix even without the dendrogram. If you look at the image generated from samples, you'll notice that there are some clusters of highly correlated samples, mostly near the diagonal line running from lower left-hand corner to the upper right-hand corner. In other words, the time points following each other are closer to each other than to other time points (what a surprise!).

8. Visualizing a heatmap

By heatmap we mean a colored figure often appearing in articles. It is basically a hierarchical clustering results, where every gene is represented by a colored bar.

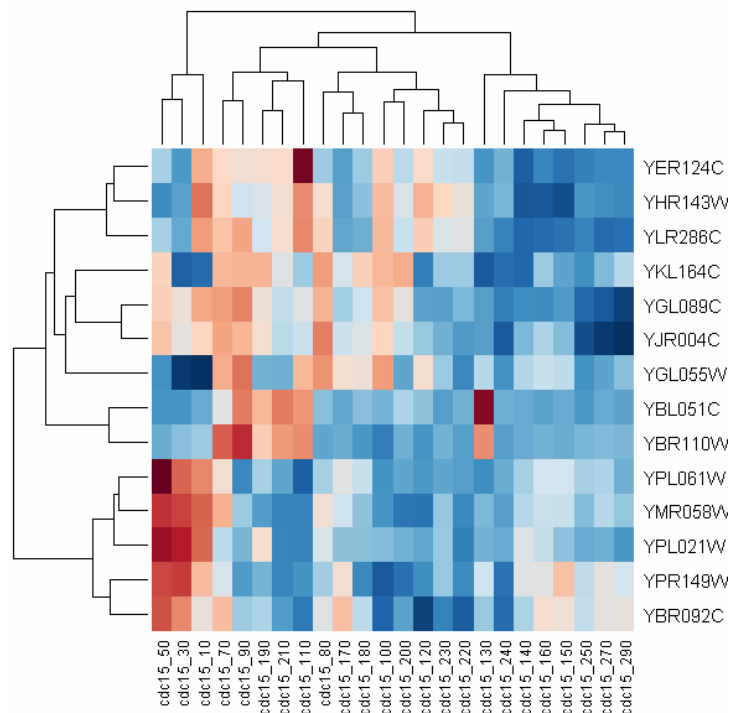
Heatmap can be produced simply by:

```
> # Heatmap needs to get the data as a matrix. Set is a data
frame, and is first converted into a matrix.
> heatmap(as.matrix(set))
```



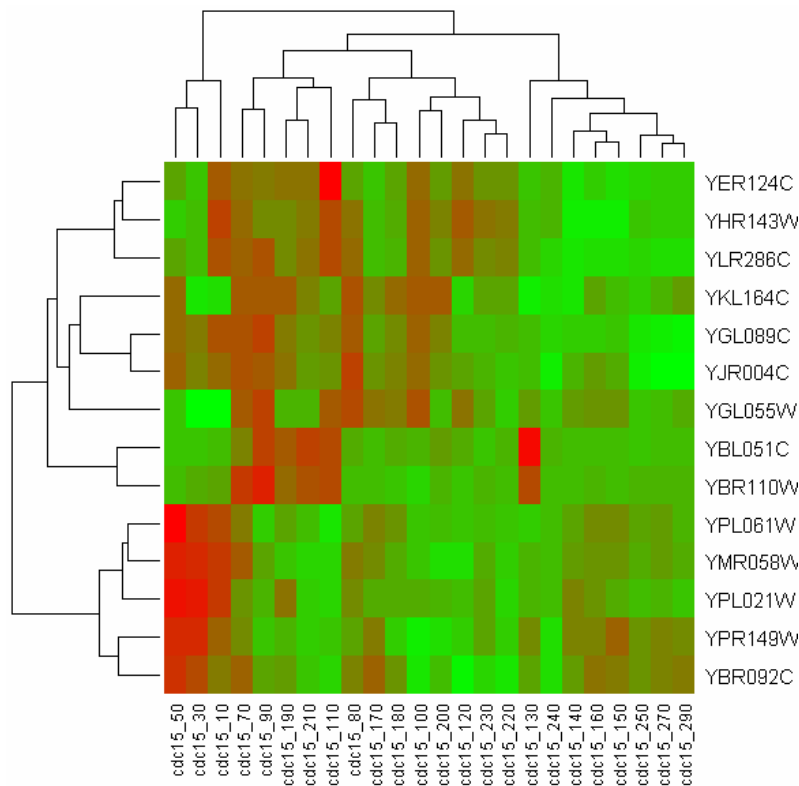
To get other colors in the heatmap, you first need to generate a sequence of colors, and then plot the heatmap using these colors:

```
> library(RColorBrewer)
> Uses 256 shades of Red and Blue
> heatcol<-colorRampPalette(brewer.pal(10, "RdBu"))(256)
> heatmap(as.matrix(set), col=heatcol)
```



Often the heatmaps are represented using red and green colors. To get this kind of an image, use:

```
> heatcol<-colorRampPalette(c("Red", "Green"))(32)
> heatmap(as.matrix(set), col=heatcol)
```



9. Saving the heatmap into a file

For further modifications, the heatmap might need to be saved in a file. This is accomplished with:

```
> cwd=getwd()
> bmp(file.path(cwd, "heatmap.bmp"), width=1800, height=1800)
> heatmap(as.matrix(set), col=heatcol)
> dev.off()
```

This results into about 6*6 inch print quality bitmap image in your data folder. Some papers might want to get a postscript image, and this is accomplished as:

```
> cwd=getwd()
> postscript(file.path(cwd, "heatmap.ps"), width=1800,
  height=1800)
> heatmap(as.matrix(set), col=heatcol)
> dev.off()
```

10. K-means clustering of genes

In K-means clustering you need to pick an artificial number, the number of clusters (K).

To produce a K-means clustering with 5 clusters, type:

```
> k<-c(5)
> km<-kmeans(set, k, iter.max=1000)
> km
```

K-means clustering with 5 clusters of sizes 3, 2, 5, 2, 2

Cluster means:

	cdc15_10	cdc15_30	cdc15_50	cdc15_70	cdc15_80	cdc15_90	cdc15_100
1	-2.320	-2.813333	-3.423333	-0.9033333	-0.420	0.6666667	0.8366667
2	2.350	2.395000	-0.145000	-1.8700000	-2.230	-2.2400000	-1.9650000
3	-1.984	0.254000	-0.290000	-1.6460000	-1.318	-1.5540000	-1.5520000
4	-1.170	-2.620000	-2.870000	-1.0550000	-0.235	0.6000000	1.9600000
5	0.520	0.800000	0.975000	-1.8450000	0.700	-3.0300000	1.1600000

	cdc15_150	cdc15_160	cdc15_170	cdc15_180	cdc15_190	cdc15_200	cdc15_210
1	-0.09666667	-0.300	-0.1933333	0.150	0.006666667	0.960	1.106667
2	0.42000000	-0.015	-0.7800000	-1.185	-0.655000000	-0.620	-0.030000
3	1.48800000	1.376	0.5340000	0.118	-0.806000000	-0.428	-0.754000
4	-1.20000000	-0.855	-1.2400000	0.550	0.315000000	1.280	1.150000
5	0.79500000	0.830	0.8400000	0.825	-1.615000000	0.385	-2.365000

	cdc15_270	cdc15_290
1	0.2066667	0.640
2	0.7300000	0.175
3	1.8680000	1.976
4	-0.7250000	-0.610
5	0.6850000	0.780

Clustering vector:

YBL051C	YBR092C	YBR110W	YER124C	YGL055W	YGL089C	YHR143W	YJR004C	YKL164C	YLR286C
5	4	5	3	2	3	3	3	2	3

Within cluster sum of squares by cluster:

```
[1] 8.585133 25.912400 58.457560 6.067450 6.896450
```

Available components:

```
[1] "cluster" "centers" "withinss" "size"
```

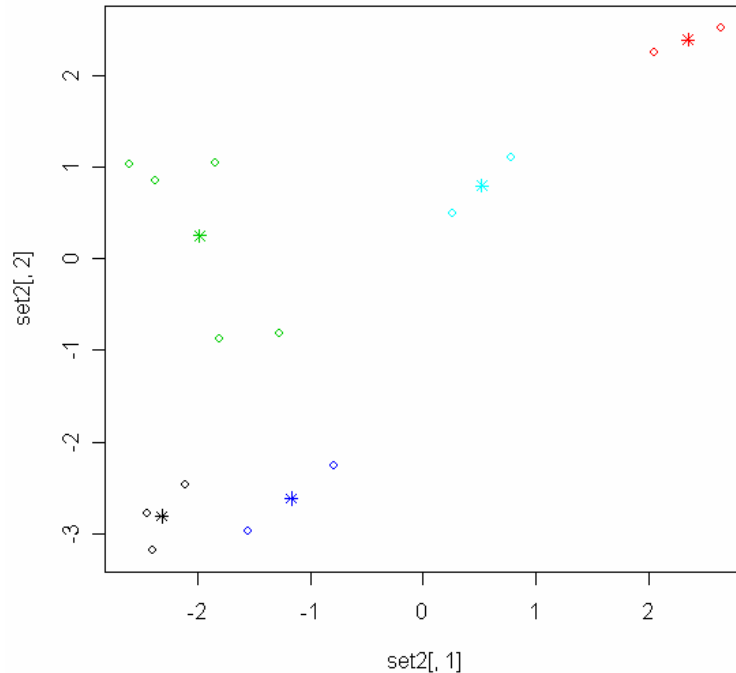
Calculate an average withinness of the results. This is a measure of how close together genes lie inside the clusters.

```
> mean(km$withinss)
[1] 21.1838
```

Run the same K-means analysis several times (save the result into a new object every time). Select the K-means clustering giving the smallest withinness score as the best result.

Visualize the K-means clustering as follows. Save the cluster membership as a new variable, and use it for coloring the data points. Last, add the cluster centers to the image.

```
> cl<-km$cluster
> plot(set[,1], set[,2], col=cl)
> points(km$centers, col = 1:5, pch = 8)
```

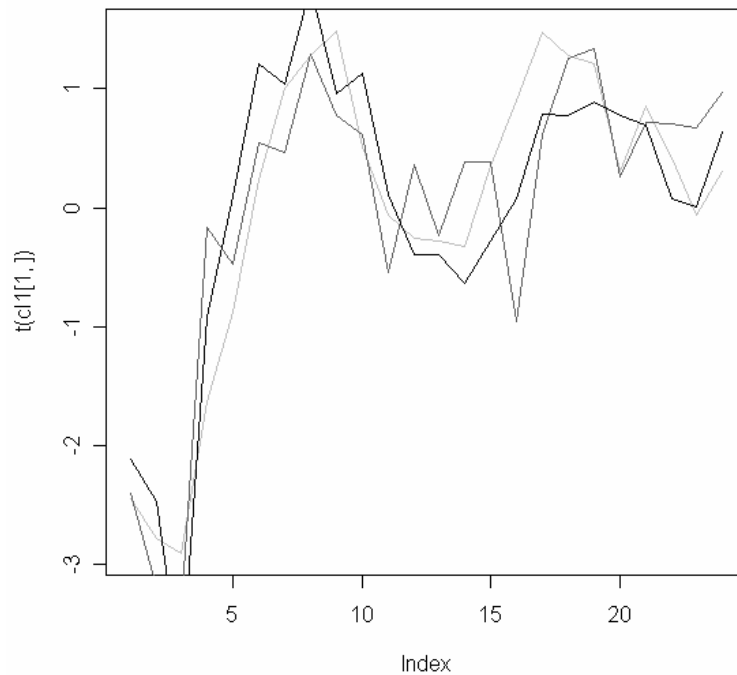


To visualize one cluster, and its expression, select the genes that belong to the same cluster. Draw the expression profile of these genes into the same image using different colors.

```
> set3<-data.frame(set, cl)
> cl1<- set[which(cl==1),] # Here we have 3 genes
> plotcol<-colorRampPalette(c("Grey", "Black"))(5)
> plot(t(cl1[1,]), type="l", col=plotcol[1])
> lines(t(cl1[2,]), type="l", col=plotcol[3])
> lines(t(cl1[3,]), type="l", col=plotcol[5])
```

How could you plot the genes in the same image with different line types (hint: ?par)?

Draw a similar image for the cluster 2.



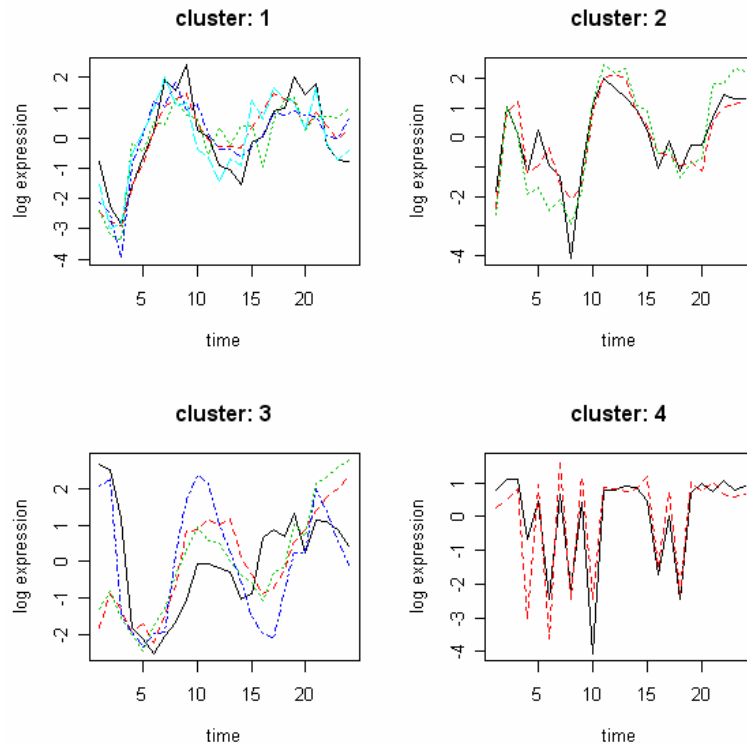
12. Drawing the whole K-means clustering

Let's produce a new K-means clustering result using four clusters:

```
> km<-kmeans(set, 4, iter.max=1000)
```

Next, initiate a 2*2 image area, and draw the expression profiles. We need to apply a for-loop here:

```
> par(mfrow=c(2,2))
> for(i in 1:4) {
>   matplot(t(set[km$cluster==i,]), type="l",
>     main=paste("cluster:", i), ylab="log expression", xlab="time")
> }
```



11. Annotata the results

Now that you have successfully found some interesting clusters, you should check what kind of genes are in the clusters. Load the yeast annotation package YEAST, and extract the gene names and descriptions. This annotates all the genes retained after filtering:

```
> library(YEAST)
> ls(package: YEAST)
> genes<-as.vector(row.names(set))
> annot1<-unlist(mget(genes, YEASTGENENAME))
> annot2<-unlist(mget(genes, YEASTDESCRIPTION))
> annot<-data.frame(rbind(annot1,annot2))
> write.table(annot, "annot.txt")
> annot[,1] # prints gene names
> annot[,2] # prints gene descriptions
```

To get the descriptions of the genes in cluster 1:

```
> c11.annot<-data.frame(rbind(unlist(mget(row.names(c11),
  YEASTGENENAME)), unlist(mget(row.names(c11),
  YEASTDESCRIPTION))))
> c11.annot
```