— Exploratory Data Analysis —
Clustering gene expression data

Jörg Rahnenführer and Florian Markowetz

Practical DNA Microarray Analysis, Heidelberg, March 2005
http://compdiag.molgen.mpg.de/ngfn/pma2005mar.shtml

Abstract. *This is the tutorial for the clustering exercises on day 2 of the course on Practical DNA Microarray Analysis. Topics will be hierarchical clustering, k-means clustering, partitioning around medoids, selecting the number of clusters, reliability of results, identifying relevant subclusters, and pitfalls of clustering.*

We consider expression data from patients with acute lymphoblastic leukemia (ALL) that were investigated using HGU95AV2 Affymetrix GeneChip arrays [Chiaretti et al., 2004]. The data were normalized using quantile normalization and expression estimates were computed using RMA. The preprocessed version of the data is available in the Bioconductor data package `ALL` from `http://www.bioconductor.org`. Type in the following code chunks to reproduce our analysis. At each ">" a new command starts, the "+" indicates that one command spans over more than one line.

```
> library(ALL)
> data(ALL)
> help(ALL)
> cl <- substr(as.character(ALL$BT), 1, 1)
> dat <- exprs(ALL)
```

The most pronounced distinction in the data is between B- and T-cells. You find these true labels in the vector `cl`. In this session we try to rediscover the labels in an unsupervised fashion. The data consist of 128 samples with 12625 genes.

1 Hierarchical clustering of samples

The first step in hierarchical clustering is to compute a matrix containing all distances between the objects that are to be clustered. For the ALL data set, the distance matrix is of size 128×128. From this we compute a dendrogram using complete linkage hierarchical clustering:

```
> d <- dist(t(dat))
> image(as.matrix(d))
> hc <- hclust(d, method = "complete")
> plot(hc, labels = cl)
```

We now split the dendrogram into two clusters (using the function `cutree`) and compare the resulting clusters with the true classes. We first compute a contingency table and then apply an independence test.

```
> groups <- cutree(hc, k = 2)
> table(groups, cl)
```

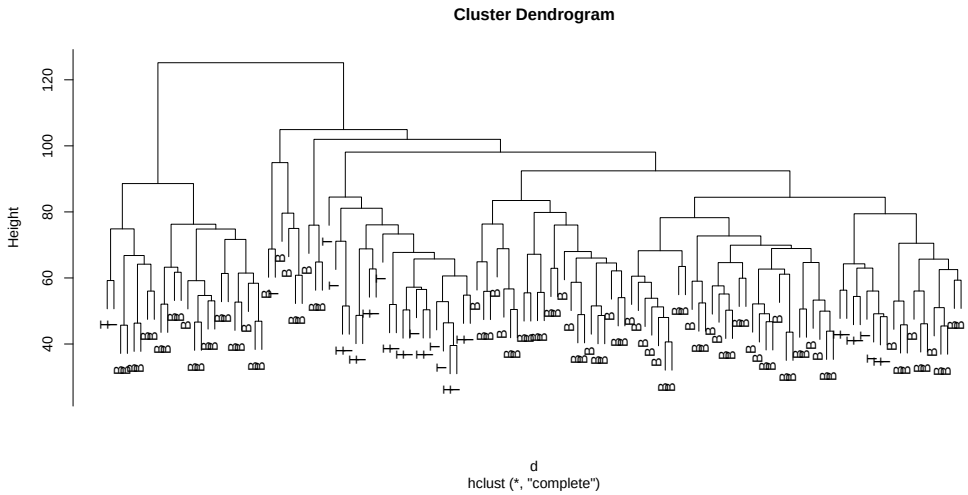

Figure 1: *Dendrogram of complete linkage hierarchical clustering. The two cell types are not well separated.*

```
      cl
groups B  T
     1 73 31
     2 22  2
```

```
> fisher.test(groups, cl)$p.value
```

```
[1] 0.03718791
```

The null-hypothesis of the Fisher test is "the true labels and the cluster results are independent". The p-value shows that we can reject this hypothesis at a significance level of 5%. But this is just statistics: What is *your* impression of the quality of this clustering? Did we already succeed in reconstructing the true classes?

Exercise: In the lecture you learned that hierarchical clustering is an agglomerative bottom-up process of iteratively joining single samples and clusters. Plot `hc` again, using as parameter `labels=1:128`. Now, read the help text for `hclust`: `hc$merge` describes the merging of clusters. Read the details carefully. Use the information in `hc$merge` and the cluster plot to explain the clustering process and to retrace the algorithm.

Exercise: Cluster the data again using `single` linkage and `average` linkage. Do the dendrogram plots change?

2 Gene selection before clustering samples

The dendrogram plot suggests that the clustering can still be improved. And the p-value is not *that* low. To improve our results, we should try to avoid using genes that contribute just noise and no information. A simple approach is to exclude all genes that show no variance across all samples. Then we repeat the analysis from the last section.

```
> genes.var <- apply(dat, 1, var)
> genes.var.select <- order(genes.var, decreasing = T)[1:100]
> dat.s <- dat[genes.var.select, ]
```

```
> d.s <- dist(t(dat.s))
> hc.s <- hclust(d.s, method = "complete")
> plot(hc.s, labels = cl)
```

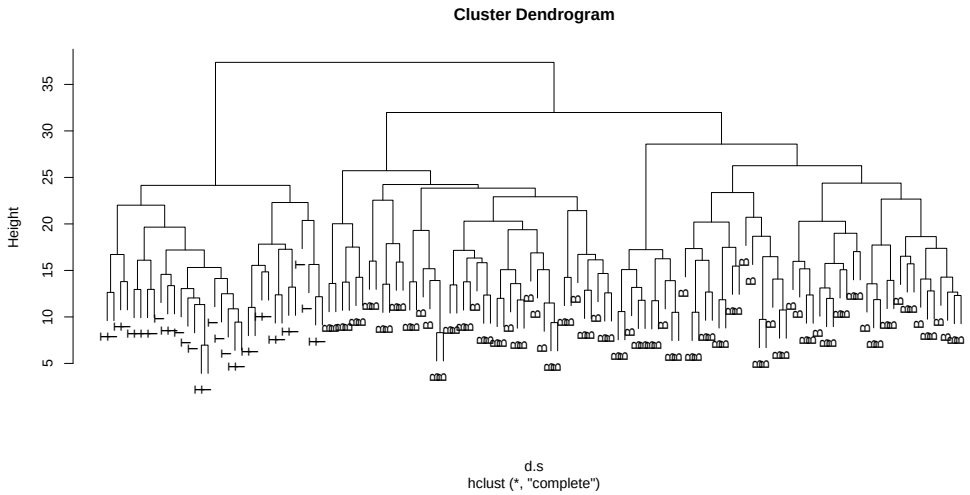

Figure 2: *Dendrogram of hierarchical clustering on 100 genes with highest variance across samples. The two classes are clearly separated.*

Plot the distance matrix `d.s`. Compare it to `d`. Then split the tree again and analyze the result by a contingency table and an independence test:

```
> groups.s <- cutree(hc.s, k = 2)
> table(groups.s, cl)
```

```
        cl
groups.s B  T
       1 95  0
       2  0 33
```

```
> fisher.test(groups.s, cl)$p.value
```

```
[1] 2.326082e-31
```

It seems that reducing the number of genes was a good idea. But where does the effect come from: Is it just dimension reduction (from 12625 to 100) or do high-variance genes carry more information than other genes? We investigate by comparing our results to a clustering on 100 *randomly* selected genes:

```
> genes.random.select <- sample(nrow(dat), 100)
> dat.r <- dat[genes.random.select, ]
> d.r <- dist(t(dat.r))
> image(as.matrix(d.r))
> hc.r <- hclust(d.r, method = "complete")
> plot(hc.r, labels = cl)
> groups.r <- cutree(hc.r, k = 2)
> table(groups.r, cl)
> fisher.test(groups.r, cl)$p.value
```

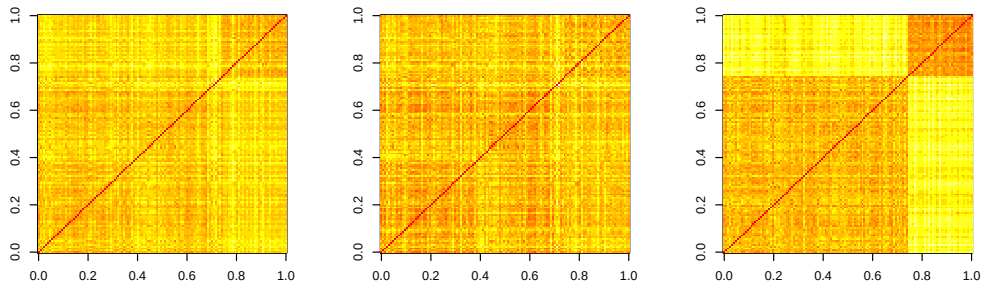

Figure 3: *Distance matrices for 128 samples, using all genes (left), 100 random genes (middle), and 100 high-variance genes (right).*

Plot the distance matrix `d.r` and compare it against the other two distance matrices `d` and `d.s`. Repeat the random selection of genes a few times. How do the dendrogram, the distance matrix, and the p-value change? How do you interpret the result?

Finally, we compare the results on high-variance and on random genes in a cluster plot in two dimensions:

```
> library(cluster)
> par(mfrow = c(2, 1))
> clusplot(t(dat.r), groups.r, main = "100 random genes", color = T)
> clusplot(t(dat.s), groups.s, main = "100 high-variance genes", color = T)
```

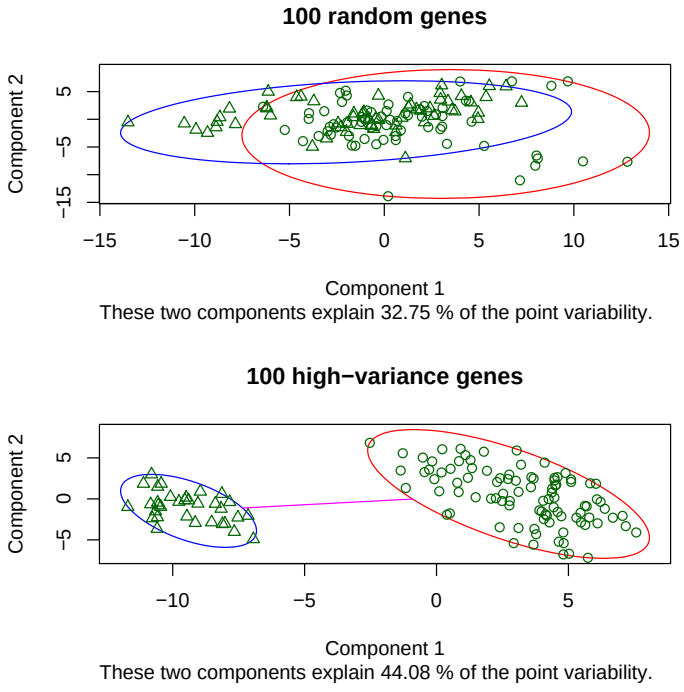

Figure 4: *You see the projection of the data onto the plane spanned by the first two principal components (= the directions in the data which explain the most variance). The group labels result from hierarchical clustering on 100 genes. In the upper plot the genes are selected randomly, in the lower plot according to highest variance across samples. The red and blue ellipses indicate class boundaries. Do we need both principal components to separate the data using high-variance genes?*

3 Partitioning methods

In hierarchical methods, the samples are arranged in a hierarchy according to a dissimilarity measure. Extracting clusters means cutting the tree-like hierarchy at a certain level. Increasing the number of clusters is just subdividing existing clusters – clusters are nested.

In partitioning methods this is different. Before the clustering process you decide on a fixed number `k` of clusters, samples are then assigned to the cluster where they best fit in. This is more flexible than hierarchical clustering, but leaves you with the problem of selecting `k`. Increasing the number of clusters is not subdivision of existing clusters but can result in a totally new allocation of samples. We discuss two examples of partitioning methods: *K-means* clustering and *PAM* (Partitioning Around Medoids).

3.1 k-means

First look at `help(kmeans)` to become familiar with the function. Set the number of clusters to `k=2`. Then perform k-means clustering for all samples and all genes. k-means uses a random starting solution and thus different runs can lead to different results. Run k-means 10 times and use the result with smallest within-cluster variance.

```
> k <- 2
> withinss <- Inf
> for (i in 1:10) {
+     kmeans.run <- kmeans(t(dat), k)
+     print(sum(kmeans.run$withinss))
+     print(table(kmeans.run$cluster, cl))
+     cat("----\n")
+     if (sum(kmeans.run$withinss) < withinss) {
+         result <- kmeans.run
+         withinss <- sum(result$withinss)
+     }
+ }
> table(result$cluster, cl)
```

The last result is the statistically best out of 10 tries – but does it reflect biology? Now do k-means again using the 100 top-variance genes. Compare the results.

```
> kmeans.s <- kmeans(t(dat.s), k)
> table(kmeans.s$cluster, cl)
```

3.2 PAM: Partitioning Around Medoids

As explained in today's lecture, PAM is a generalization of k-means. Look at the help page of the R-function. Then apply `pam` for clustering the samples using all genes:

```
> result <- pam(t(dat), k)
> table(result$clustering, cl)
```

Now use `k=2:50` top variance genes for clustering and calculate the number of misclassifications in each step. Plot the number of genes versus the number of misclassifications. What is the minimal number of genes needed for obtaining 0 misclassifications?

```
> ngenes <- 2:50
> o <- order(genes.var, decreasing = T)
> miscl <- NULL
> for (k in ngenes) {
+     dat.s2 <- dat[o[1:k], ]
+     pam.result <- pam(t(dat.s2), k = 2)
+     ct <- table(pam.result$clustering, cl)
+     miscl[k] <- min(ct[1, 2] + ct[2, 1], ct[1, 1] + ct[2, 2])
+ }
> xl = "# genes"
> yl = "# misclassification"
> plot(ngenes, miscl[ngenes], type = "l", xlab = xl, ylab = yl)
```

4 How many clusters are in the data?

Both `kmeans` and `pam` assume that the number of clusters is known. The methods will produce a result no matter what value of `k` you choose. But how can we decide, which choice is a good one?

4.1 The objective function

Partitioning methods try to optimize an objective function. The objective function of k-means is the sum of all within-cluster sum of squares. Run k-means with `k=2:20` clusters and 100 top variance genes. For `k=1` calculate the total sum of squares. Plot the obtained values of the objective function.

```
> totalsum <- sum(diag((ncol(dat.s) - 1) * cov(t(dat.s))))
> withinss <- numeric()
> withinss[1] <- totalsum
> for (k in 2:20) {
+     withinss[k] <- sum(kmeans(t(dat.s), k)$withinss)
+ }
> plot(1:20, withinss, xlab = "# clusters", ylab = "objective function", type = "b")
```

Why is the objective function not necessarily decreasing? Why is `k=2` a good choice?

4.2 The silhouette score

First read the *Details* section of `help(silhouette)` for a definition of the silhouette score. Then compute silhouette values for PAM clustering results with `k=2:20` clusters. Plot the silhouette widths. Choose an optimal `k` according to the maximal average silhouette width. Compare silhouette plots for `k=2` and `k=3`. Why is `k=2` optimal? Which observations are misclassified? Which cluster is more compact?

```
> asw <- numeric()
> for (k in 2:20) {
+     asw[k] <- pam(t(dat.s), k)$silinfo$avg.width
+ }
> plot(1:20, asw, xlab = "# clusters", ylab = "average silhouette width", type = "b")
> plot(silhouette(pam(t(dat.s), 2)))
```

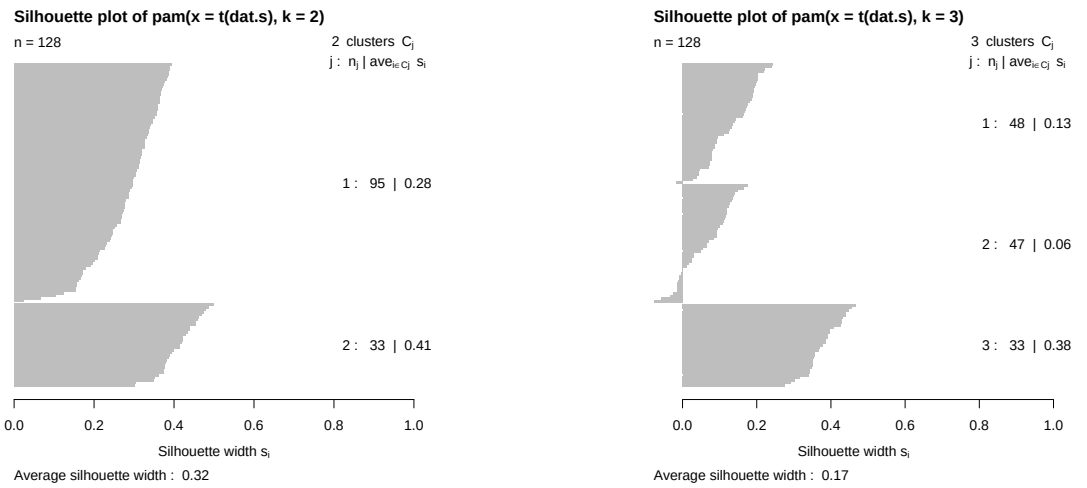

Figure 5: *Silhouette plot for k=2 and k=3*

5 How to check significance of clustering results

Randomly permute class labels 20 times. For k=2, calculate the average silhouette width (asw) for pam clustering results with true labels and for all random labelings. Generate a box-plot of asw values obtained from random labelings. Compare it with the asw value obtained from true labels.

```
> d.s <- dist(t(dat.s))
> cl.true <- as.numeric(cl == "B")
> asw <- mean(silhouette(cl.true, d.s)[, 3])

> asw.random <- rep(0, 20)
> for (sim in 1:20) {
+     cl.random = sample(cl.true)
+     asw.random[sim] = mean(silhouette(cl.random, d.s)[, 3])
+ }
> symbols(1, asw, circles = 0.01, ylim = c(-0.1, 0.4), inches = F, bg = "red")
> text(1.2, asw, "Average silhouette value\n for true labels")
> boxplot(data.frame(asw.random), col = "blue", add = T)
```

You will see: The average silhouette value of the pam clustering result lies well outside the range of values achieved by random clusters.

6 Clustering of genes

Cluster the 100 top variance genes with hierarchical clustering. You can get the gene names from the annotation package hgu95av2.

```
> library(hgu95av2)
> gene.names <- sapply(geneNames(ALL), get, hgu95av2SYMBOL)
> gene.names <- gene.names[genes.var.select]
> d.g = dist(dat.s)
> hc = hclust(d.g, method = "complete")
> plot(hc, labels = gene.names)
```

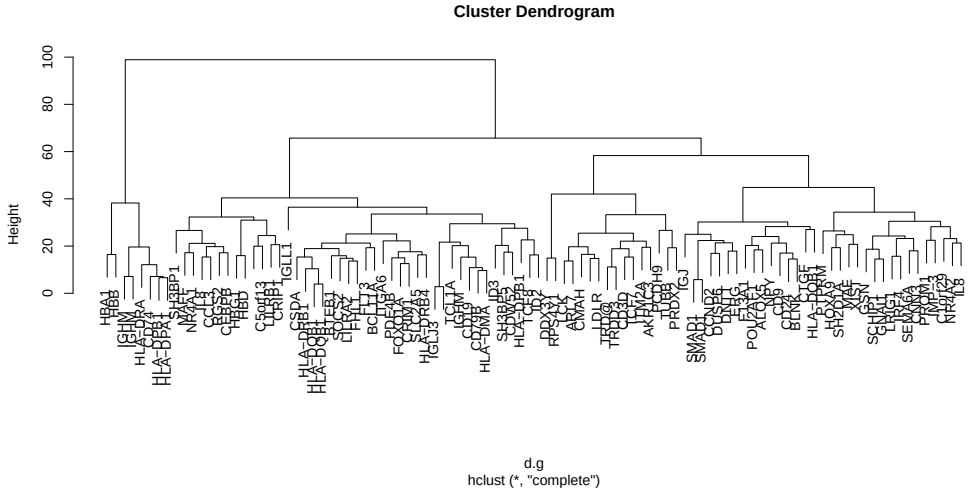

Figure 6: *Hierarchical clustering of 100 top variance genes*

Now plot the average silhouette widths for `k=2:20` clusters using `pam` and have a look at the silhouette plot for `k=2`. What do you think: Did we find a structure in the genes?

```
> asw = numeric()
> for (k in 2:20) {
+     asw[k] = pam(dat.s, k)$silinfo$avg.width
+ }
> plot(1:20, asw, xlab = "# clusters", ylab = "average silhouette width", type = "b")
> plot(silhouette(pam(dat.s, 2)))
```

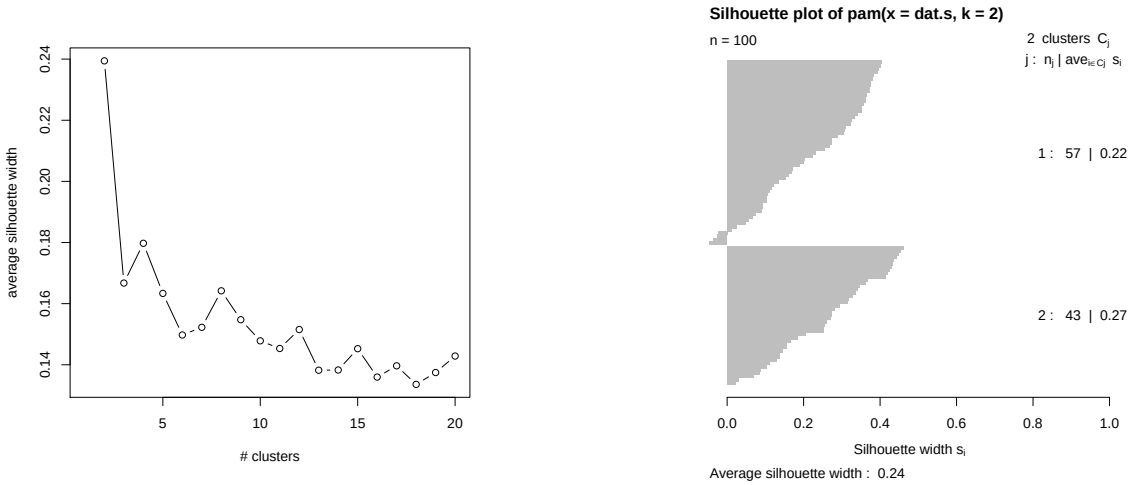

Figure 7: *Left: Average silhouette width of the 100 top-variance genes for `k=2:20` clusters. Right: Silhouette plot for `k=2`.*

7 Presenting results

In the last section we saw that a substructure in the genes is not easy to find. When interpreting heatmap plots like the following, it is important to keep in mind: Any clustering method will eventually produce

a clustering; before submitting a manuscript to Nature you should check the validity and stability of the groups you found.

```
> help(heatmap)
> heatmap(dat.s)
```

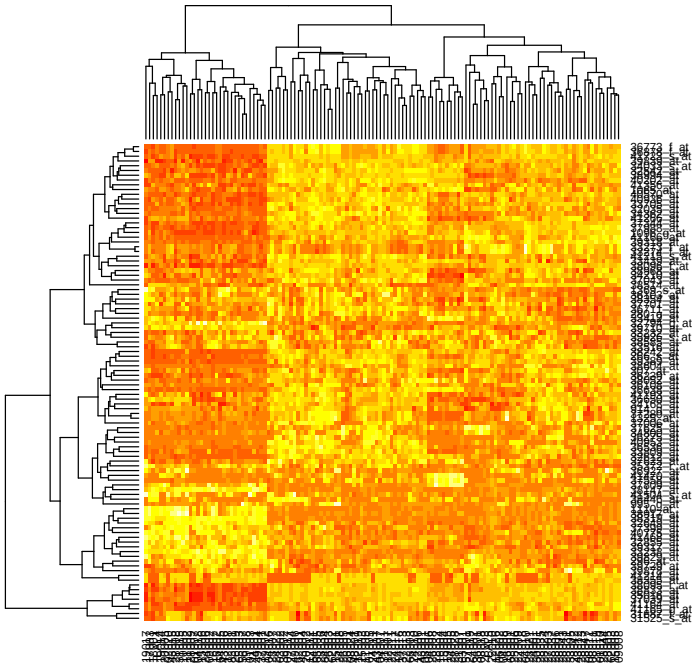

Figure 8: *Heatmap representation of all samples and 100 selected high-variance genes. Rows and columns are hierarchically clustered.*

8 How to fake clusterings

For publication of your results it is advantageous to show nice looking cluster plots supporting your claims. Here we demonstrate how to produce almost *any clustering you like* in a situation with few samples but many genes.

First reduce the data set to the first 5 B-cell and T-cell samples. Then permute class labels randomly. The permuted labels do not correspond to biological entities. Nevertheless we will demonstrate how to get a perfect clustering result.

The strategy is easy: Just select 100 genes according to differential expression between the groups defined by the random labels, then perform hierarchical clustering of samples with these genes only.

```
> select = c(which(cl == "B")[1:5], which(cl == "T")[1:5])
> dat.small = dat[, select]
> cl.random = sample(cl[select])
> library(multtest)
> tstat = mt.teststat(dat.small, classlabel = (cl.random == "B"), test = "t")
> genes = order(abs(tstat), decreasing = T)[1:100]
> dat.split = dat.small[genes, ]
> d = dist(t(dat.split))
```

```
> hc = hclust(d, method = "complete")
> par(mfrow = c(1, 2))
> plot(hc, labels = cl.random, main = "Labels used for gene selection", xlab = "")
> plot(hc, labels = cl[select], main = "True labels", xlab = "")
```

Repeat this experiment a few times and compare the two plots. You should also have a look at the heatmap representation of `dat.split`.

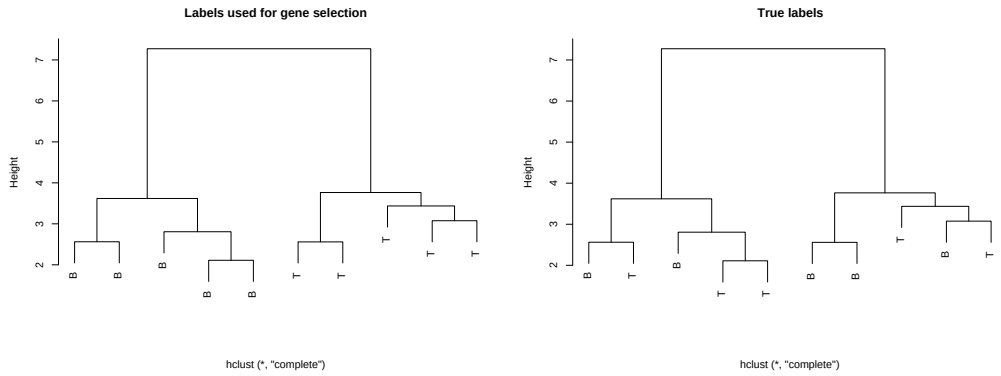

Figure 9: *Selecting only highly differential genes leads to well separated groups of samples — even if these groups are defined by random assignments of labels without any biological meaning.*

9 Ranking is no clustering!

A typical situation in microarray analysis is the following. You are interested in one specific gene and want to identify genes with a similar function. A reasonable solution to this problem is to calculate the distance of all other genes to the gene of interest and then analyze the list of the closest (top-ranking) genes.

We choose the fifth gene out of our 100 high variance genes and print the distances and the names of the 10 genes that are closest to it.

```
> list.distances = as.matrix(d.g)[5, ]
> o = order(list.distances)
> list.distances[o][1:10]
> gene.names[o][1:10]
```

You can see that the original gene of interest is ranked first with a distance 0 to itself, and the second and the seventh gene are replicates of this gene. This is a plausible result.

Another popular approach in this situation is to first cluster all genes and then analyze the genes that are in the same cluster as the original gene of interest. In general this strategy is not suitable since even two objects that are very close to each other can belong to two different clusters if they are close to a cluster boundary line.

In our example, look at the result of the hierarchical clustering of genes obtained above. Define three clusters based on this result and print the clusters of the 10 closest genes identified above.

```
> hc = hclust(d.g, method = "complete")
> cutree(hc, 3)[o][1:10]
```

To which clusters do the original gene of interest and the replicates belong? You can see that even on this low clustering level the replicates are separated. Thus remember: If you are interested in detecting genes that are similar to a gene of your choice, you should always prefer a ranked list to a clustering approach.

10 ISIS: Identifying splits with clear separation

At the end of this exercise we present a method, which addresses two problems of clustering samples:

1. Only a subset of genes may be important to distinguish one group of samples from another.
2. Different sets of genes may determine different splits of the sample set.

The method ISIS (von Heydebreck *et al.*, 2001) searches for binary class distinctions in the set of samples that show clear separation in the expression levels of specific subsets of genes. Several mutually independent class distinctions may be found, which is difficult to obtain from the most commonly used clustering algorithms. Each class distinction can be biologically interpreted in terms of its supporting genes.

```
> library(isis)
> help(isis)
```

Read the help text and find out about the parameters of the function `isis`. Now we use it on the 1000 top variance genes:

```
> dat.isis <- dat[order(apply(dat, 1, var), decreasing = T)[1:1000], ]
> splits <- isis(dat.isis)
```

The matrix `splits` contains in each row a binary class separation of samples and in the last column the corresponding score.

Exercise: Use `table()` to compare the found splits with the vector `cl`. You will see: One of the high-scoring splits perfectly recovers the difference between B- and T-cells.

Exercise: Find out what the parameter `p.off` does. Run `isis` again with `p.off=10`. How does the result change?

11 Summary

Scan through this paper once again and identify in each section the main message you have learned. Some of the most important points are:

- You learned about hierarchical clustering, k-means, partitioning around medoids, and silhouette scores.
- Selecting high-variance genes before clustering improves the result.
- Selecting only highly differential genes leads to well separated groups of samples — even if these groups are defined by random assignments of labels without any biological meaning.

If you have any comments, criticisms or suggestions on our lectures, please contact us:
`rahnenfj@mpi-sb.mpg.de` and `florian.markowetz@molgen.mpg.de`

12 References

Chiaretti S, Li X, Gentleman R, Vitale A, Vignetti M, Mandelli F, Ritz J, Foa R. Gene expression profile of adult T-cell acute lymphocytic leukemia identifies distinct subsets of patients with different response to therapy and survival. Blood. 2004 Apr 1;103(7):2771-8.

Hastie T, Tibshirani R, Friedman J. The Elements of Statistical Learning. Springer, 2001.

von Heydebreck A, Huber W, Poustka A, Vingron M. Identifying splits with clear separation: a new class discovery method for gene expression data. Bioinformatics 2001 17: 107-114S.