

# R/BioC Exercises: Unsupervised methods

Perry Moerland

April 20, 2010

☞ Information on how to log on to a PC in the exercise room and the UNIX server can be found here: <http://bioinformaticslaboratory.nl/twiki/bin/view/BioLab/EducationBioinformaticsII>. Don't forget to enable X11 forwarding when starting up PuTTY.

☞ These exercises depend on Chapters 12 (Gentleman et al. 2005) and 13 (Pollard and van der Laan 2005). References to equations are with respect to these chapters.

☞ I would like to urge you to always closely look at the R code included, the data imported, and the values of the variables created. Also use the *help* function to look at all the different options functions offer. Don't forget that the goal of these exercises is to make you *think!*

## Introduction

Unsupervised learning methods aim at detecting structures in data. The term *unsupervised* refers to the fact that these methods do not use gene or sample annotations, only the (normalized) expression intensities or ratios are used. A primary purpose of such methods is to group similar data together (clustering) and provide a visualization of the data in which structures can easily be recognized. These may be relations among genes, among samples, or even between genes and samples. The discovery of such structures can lead to the development of new hypotheses, *e.g.*, the grouping of genes with similar expression profiles may indicate that they are co-regulated and are possibly involved in the same biological process. If one looks at arrays/samples instead of genes, the separation of expression profiles of patient tissue samples may point to a possible refinement of disease taxonomy. On the other hand, unsupervised methods are often used to confirm known differences between genes or samples. If a clustering algorithm groups samples from two different tumor types into distinct clusters, this provides evidence that the tumor types indeed show clearly detectable differences in their global expression profiles.

☞ If gene or sample annotations (or classes) are available the more powerful approach is to use them and apply techniques for *supervised* learning (tomorrow's lab).

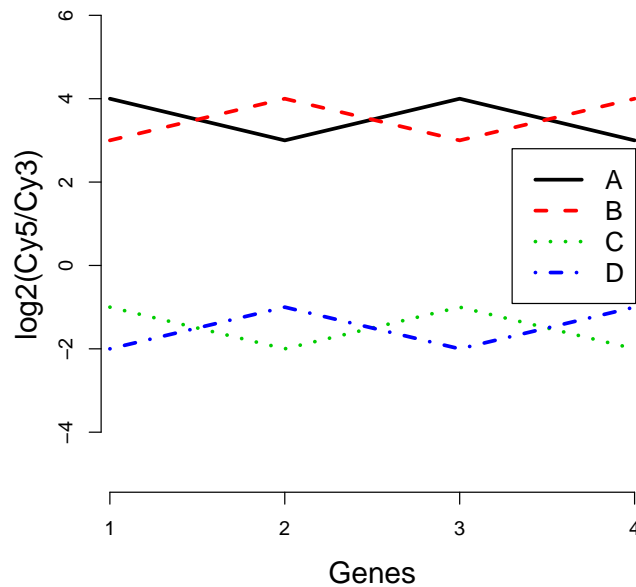
As you might have noticed, in the above the word *similar* was used several times. This is really a central concept in unsupervised learning be it clustering or visualization. In clustering one wants to group similar objects together, whereas in visualization one wants to find a representation of a high-dimensional data set in two or three dimensions while loosing as little

information as possible: objects that are similar in the original data space should also be similar in the low-dimensional space. See (Gentleman et al. 2005) for a more elaborate discussion.

## Distance measures and clustering

We did a mini-experiment with four arrays ( $A, B, C, D$ ) and 4 genes per array. This data set has been checked for low-quality arrays and properly normalized. The resulting log-ratios are given in file `/data/home/stud00/Unsup/hcexample.txt`.

**Exercise 1:** Copy this file to your home directory, start R, read in `hcexample.txt` and store it in a variable  $E$  for later use. Try to make a plot of the array profiles that looks more or less like the one below. You might want to look into some of the more detailed plotting commands in R: `plot.new`, `plot.window`, `axis` etc.



Cluster algorithms group similar data together. What is meant by the word “similar” is formally defined by the notion of a *distance*. In R, the `bioDist` package gives a collection of functions for calculating distance measures. We will have a look at two of them in more detail, first load the package:

```
> library(bioDist)
```

Now calculate the Euclidean (12.2, EUC) and the Pearson sample correlation (12.2, COR) distance between the array profiles.

```
> d.euc <- euc(t(E))  
> d.euc
```

```

      A      B      C
B  2.00000
C 10.00000 10.19804
D 10.19804 10.00000  2.00000

> d.cor <- cor.dist(t(E), abs = FALSE)
> d.cor

  A B C
B 2
C 0 2
D 2 0 2

```

Note that you had to transpose the data matrix  $E$  since the `bioDist` package calculates pairwise distances for all rows of a matrix.<sup>1</sup>

**Exercise 2:** Can you explain the resulting distance matrix when using the Pearson correlation distance?

Such a distance or *dissimilarity* matrix forms the basis for most clustering algorithms. In microarray literature, an agglomerative (*i.e.*, bottom-up) hierarchical approach such as implemented in the `hclust` function is popular. As explained in the lecture, hierarchical clustering tries to find a tree-like representation of the data in which clusters have subclusters, which have subclusters, and so on. The number of clusters depends on in how much detail one looks at the tree. Hierarchical clustering uses two types of distances:

- Distance between two individual data points (Euclidean, correlation etc. )
- Distance between two clusters of data points, also called *linkage* (single, average, etc.).

**Exercise 3:** Draw (just with pencil on paper) the dendrograms for both the Euclidean and the correlation distance matrix generated above. Explain your results. Also write the R code using `hclust` for both cases when using average linkage, store the dendrograms in variables `hc.euc` and `hc.cor` respectively. Plot the resulting dendrograms.

Clusters can be extracted from the resulting dendrograms by a horizontal cut:

```

> cutree(hc.euc, 2)

A B C D
1 1 2 2

> cutree(hc.cor, 2)

A B C D
1 2 1 2

```

These two vectors show to which cluster each array is attributed.

---

<sup>1</sup>Some of the help pages are wrong and claim that distances are calculated column-wise.

## Clustering synthetic data

One peculiarity of clustering algorithms is that they always produce clusters. This happens regardless of whether there is actually any meaningful clustering structure present in the data. Let us now simulate some unstructured data (*rnorm* randomly generates from a normal distribution) and see what happens.

```
> n.genes <- 1000
> n.samples <- 50
> descr <- paste("S", rep(c("0", ""), times = c(9, 41)), 1:50,
+   sep = "")
> set.seed(13)
> dataMatrix <- matrix(rnorm(n.genes * n.samples), nrow = n.genes)
```

**Exercise 4:** Use *hclust* with average, single, and complete linkage to cluster the samples and plot the resulting dendrograms. Which phenomenon do you see with single linkage? How might this make interpretation difficult?

Some of the dendrograms generated above appear to find structure in the data. You might decide that the complete linkage dendrogram contains four clusters. Let us assume that you stored the complete linkage dendrogram in variable *hComEuc* (because of the randomness in this data set results are very likely to be different from the ones shown here).

```
> cutree(hComEuc, 4)

 [1] 1 2 2 3 2 3 2 3 2 2 2 1 2 2 2 2 3 2 3 4 3 3 2 3 1 3 3 3 3 1 3 3 3 4 3 3 4 3
[39] 3 3 2 3 3 3 2 3 2 3 2 3

> table(cutree(hComEuc, 4))

 1  2  3  4
 4 17 26  3
```

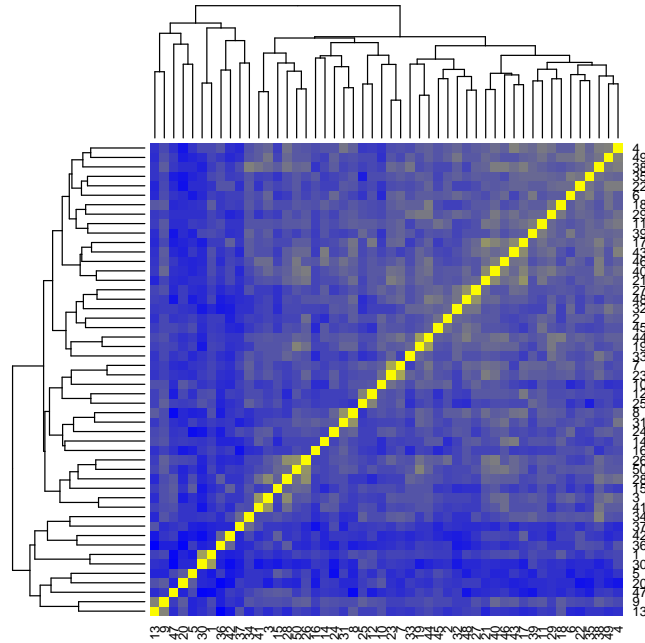
Is this structure real? Since the data generated was completely random, this is not very likely. A nice way of testing the *stability* of a clustering result is to slightly perturb the data numerous times and see whether results are robust with respect to these small perturbations. This can be done by *bootstrapping* the individual genes, or rows of the data matrix (section 13.2.6 in (Pollard and van der Laan 2005)). The idea behind the bootstrap is to create a new data matrix (of the same size as the original) by randomly selecting rows. Sampling is done with replacement, so some rows will be included multiple times and some rows will be omitted. Here is a simple example where we bootstrap row indices for a data matrix with 10 rows:

```
> sample(1:10, replace = T)

 [1] 4 4 8 6 2 8 9 1 2 9
```

We can use the `ClassDiscovery` package to perform bootstrap resampling for clustering. In order to test the stability of  $k = 4$  groups for hierarchical clustering with Euclidean distance and complete linkage using 200 bootstrap samples, we do the following:

```
> library(ClassDiscovery)
> bc <- BootstrapClusterTest(dataMatrix, cutHclust, k = 4, method = "complete",
+   metric = "euclid", nTimes = 200)
> image(bc, col = blueyellow(64))
```



The main result of `BootstrapClusterTest` is that for each pair of columns (arrays in this case) in the original data, the number of times they belong to the same cluster of a bootstrap sample is calculated. This is stored in `bc@result` and used to create the heatmap displayed in the figure. Two samples that often cluster together are shown as yellow and two samples that cluster often apart are shown as blue. The figure clearly illustrates the absence of real structure in the data.

**Exercise 5:** Apply the bootstrap cluster test to the Euclidean dendrogram from the toy example data stored in variable `E` (see previous section). How many clusters would you expect? Is there stable structure in this data set?

Occasionally, you see papers comparing two known classes of samples that perform the following analysis:

- Find a list of differentially expressed genes.
- Cluster the data using only the differentially expressed genes.

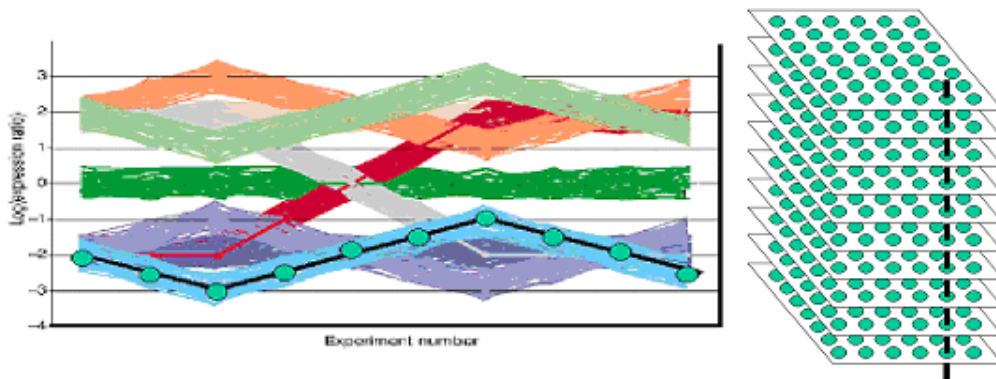


Figure 1: The black line depicts one single gene expression profile and the vertical black line in the right-hand image figure illustrates how the black gene expression profile is constructed: the gene expression levels of the same spot (=gene) over multiple microarray experiments are taken. Only ten arrays are considered in this simulated dataset. Therefore, the left-hand figure has ten experiments on the  $x$ -axis. The data set consists of nine different profiles; round each profile 50 noisy profiles were generated. Therefore, each simulated array contains  $9 \cdot (50+1) = 459$  different profiles. The log-ratio of these gene profiles is indicated on the  $y$ -axis.

- Discover that you can successfully distinguish the known classes.

Is this really surprising?

**Exercise 6:** Let us try this on our simulated data stored in `dataMatrix`. Divide the 50 samples into two classes (the first 25 and the last 25). Next, perform  $t$ -tests (using the function `rowttests` from the `genefilter` package) to see how well each gene separates the two classes, and cluster the data using the top 10 genes (hint: use the function `order`) with Euclidean distance and complete linkage. Use bootstrapping to test the stability of the grouping found.

This example illustrates that it is dangerous to filter on a specific contrast such as differential expression. Just due to chance, this can lead to structure even in random data.

## Clustering less synthetic data

Figure 1 shows the gene profiles of yet another synthetic data set which is slightly closer to biological reality. The data can be found in `/data/home/stud00/Unsup/jq.txt`.

**Exercise 7:** When clustering genes with the Euclidean distance, how many clusters would you expect? Write the code to verify your answer. You might want to explore the `heatmap` function for this purpose. A heat map is a false color image with a dendrogram added to the left side and to the top (like in the previous section). Especially, explore the influence of the `scale` argument of `heatmap`. Also try to relate the clusters found to the gene expression profiles of Figure 1.

**Exercise 8:** Until now we often had to specify the number of clusters in advance. There are many methods to choose the optimal number of clusters (section 13.2.7 in (Pollard and van der Laan 2005)). A graphical procedure could be to plot the heights of the branches of a dendrogram and then to determine where the curve becomes flatter (remember the fusion graph from the lecture). This corresponds to the point where the data is not structured anymore. Can you do this for our synthetic dataset using the function *hclust*? Then use the function *locator* to determine the optimal number of clusters in this dataset.

## ***K*-means clustering**

Another often-used clustering algorithm is the *K*-means algorithm. From the lecture you might remember that the *K*-means algorithm tries to minimize the total within-scatter. We are going to study the behaviour of the algorithm in some more detail on a simple simulated dataset.

**Exercise 9:** • Use *mvnorm* from the MASS package to create a dataset consisting of three spherical clusters of 50 datapoints each. Means of the clusters are at (0,0), (10,10), and (20,20). Covariance matrices are diagonal with ones on the diagonal. Make a scatterplot of the resulting dataset.

- Run the function *kmeans* 50 times and keep track of the total within-scatter. What do you observe? Can you relate this to the clustering solutions found and give an explanation of this behaviour?
- Does hierarchical clustering also suffer from this problem?
- Compute the total within-scatter for *kmeans* with [2,3,5,10,20,50,100,149] clusters and then calculate the normalised within-scatter by dividing all within-scatter values by the within-scatter for two clusters. Repeat this 10 times and plot the average normalised within-scatter as a function of the number of clusters.
- What are the most prominent characteristics of this curve?

## **References**

- Gentleman, R., V. Carey, W. Huber, R. Irizarry, and S. Dudoit (Eds.) (2005). *Bioinformatics and Computational Biology Solutions Using R and Bioconductor*. New York: Springer.
- Gentleman, R., B. Ding, S. Dudoit, and J. Ibrahim (2005). Chapter 12: Distance measures in DNA microarray data analysis. See Gentleman, Carey, Huber, Irizarry, and Dudoit (2005).
- Pollard, K. and M. van der Laan (2005). Chapter 13: Cluster analysis of genomic data. See Gentleman, Carey, Huber, Irizarry, and Dudoit (2005).

This document was generated with

```
> sessionInfo()
```

```
R version 2.10.1 (2009-12-14)  
i386-pc-mingw32
```

```
locale:
```

```
[1] LC_COLLATE=English_United Kingdom.1252  
[2] LC_CTYPE=English_United Kingdom.1252  
[3] LC_MONETARY=English_United Kingdom.1252  
[4] LC_NUMERIC=C  
[5] LC_TIME=English_United Kingdom.1252
```

```
attached base packages:
```

```
[1] stats      graphics  grDevices  utils      datasets  methods    base
```

```
other attached packages:
```

```
[1] genefilter_1.28.2      ClassDiscovery_2.10.1 mclust_3.4.4  
[4] cluster_1.12.1        PreProcess_2.10.0     oompaBase_2.10.1  
[7] bioDist_1.18.0         KernSmooth_2.23-3     Biobase_2.6.1
```

```
loaded via a namespace (and not attached):
```

```
[1] annotate_1.24.1        AnnotationDbi_1.8.2 DBI_0.2-5  
[4] RSQLite_0.8-4         splines_2.10.1      survival_2.35-7  
[7] xtable_1.5-6
```