Outline

Administrivia and Introduction

- Course Structure
- Syllabus
- Introduction to Data Mining

Dimensionality Reduction

- Introduction
- Principal Components Analysis
- Singular Value Decomposition
- Multidimensional Scaling
- Isomap

Clustering

- Introduction
- Hierarchical Clustering
- K-means
- Vector Quantisation
- Probabilistic Methods

Outline

Administrivia and Introduction

- Course Structure
- Syllabus
- Introduction to Data Mining

Dimensionality Reduction

- Introduction
- Principal Components Analysis
- Singular Value Decomposition
- Multidimensional Scaling
- Isomap

Clustering

- Introduction
- Hierarchical Clustering
- K-means
- Vector Quantisation
- Probabilistic Methods

Clustering

- Cluster analysis is a range of methods that reveal structural information about high-dimensional data directly.
- Given a set of unclassified points X, cluster analysis seeks to arrange them into clusters based on some notion of between cluster and within cluster distance/dissimilarity.
- Partition based methods:
  - Allocate points into K clusters.
  - The number of cluster is usually fixed beforehand or investigated for various values of K as part of the analysis.
- Hierarchical clustering methods:
  - Allocate points into clusters and clusters into super-clusters forming a hierarchy.
  - Typically the hierarchy forms a binary tree (a dendrogram) where each cluster has two “children”.

Outline

Administrivia and Introduction

- Course Structure
- Syllabus
- Introduction to Data Mining

Dimensionality Reduction

- Introduction
- Principal Components Analysis
- Singular Value Decomposition
- Multidimensional Scaling
- Isomap

Clustering

- Introduction
- Hierarchical Clustering
- K-means
- Vector Quantisation
- Probabilistic Methods

Hierarchical Clustering Methods

- Hierarchically structured data can be found everywhere (measurements of different species and different individuals within species), hierarchical methods attempt to understand data by looking for clusters.
- There are two general strategies for generating hierarchical clusters. Both proceed by seeking to minimize some measure of dissimilarity.
    - Agglomerative / Bottom-Up / Merging
    - Divisive / Top-Down / Splitting

  Hierarchical clusters are generated where at each level, clusters are created by merging clusters at lower levels. This process can easily be viewed by a dendogram/tree.

Agglomerative Strategies

- The essence of agglomerative strategies is very simple: starting with each observation as a separate cluster, recursively merge the two most similar clusters (with the smallest dissimilarity) until we are left with a single cluster.
- The way in which we measure dissimilarity between clusters affects the resulting dendograms in a predictable way. If clusters exist however, it is clear by inspecting the dendograms using whatever way we measure dissimilarity between clusters.

Measuring Dissimilarity

To find hierarchical clusters, we need some way to measure the dissimilarity between clusters

- Given two points x_i and x_j, it is straightforward to measure their dissimilarity, say $d(x_i, x_j) = \|x_i - x_j\|_2$.
- It is unclear however how to extend this to measure dissimilarity between *clusters*, $D(C_i, C_j)$ for clusters C_i and C_j.

Many such proposals though no concensus as to which is best.

(a) *Single-Link Clustering*

$$D(C_i, C_j) = \min_{x,y} \left(d(x, y) | x \in C_i, y \in C_j\right)$$

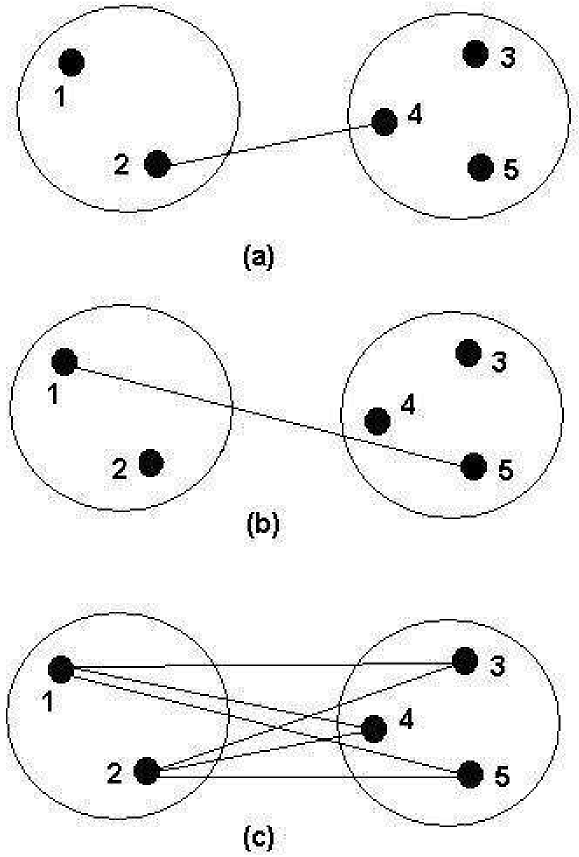
(b) *Complete-Link Clustering*

$$D(C_i, C_j) = \max_{x,y} \left(d(x, y) | x \in C_i, y \in C_j\right)$$

(c) *Group-Average Clustering*

$$D(C_i, C_j) = avg_{x,y} \left(d(x, y) | x \in C_i, y \in C_j\right)$$

Measuring Dissimilarity



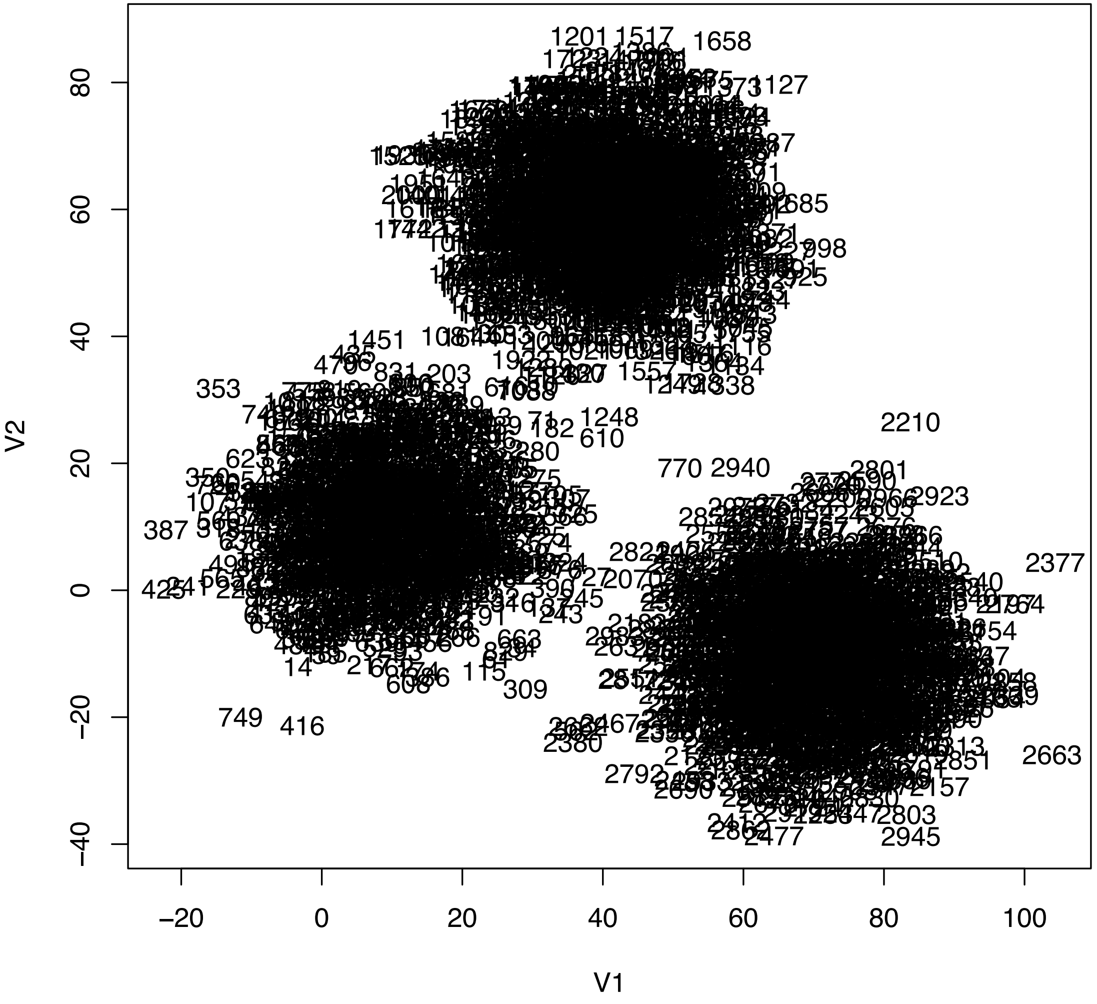
Cluster Distance

d24

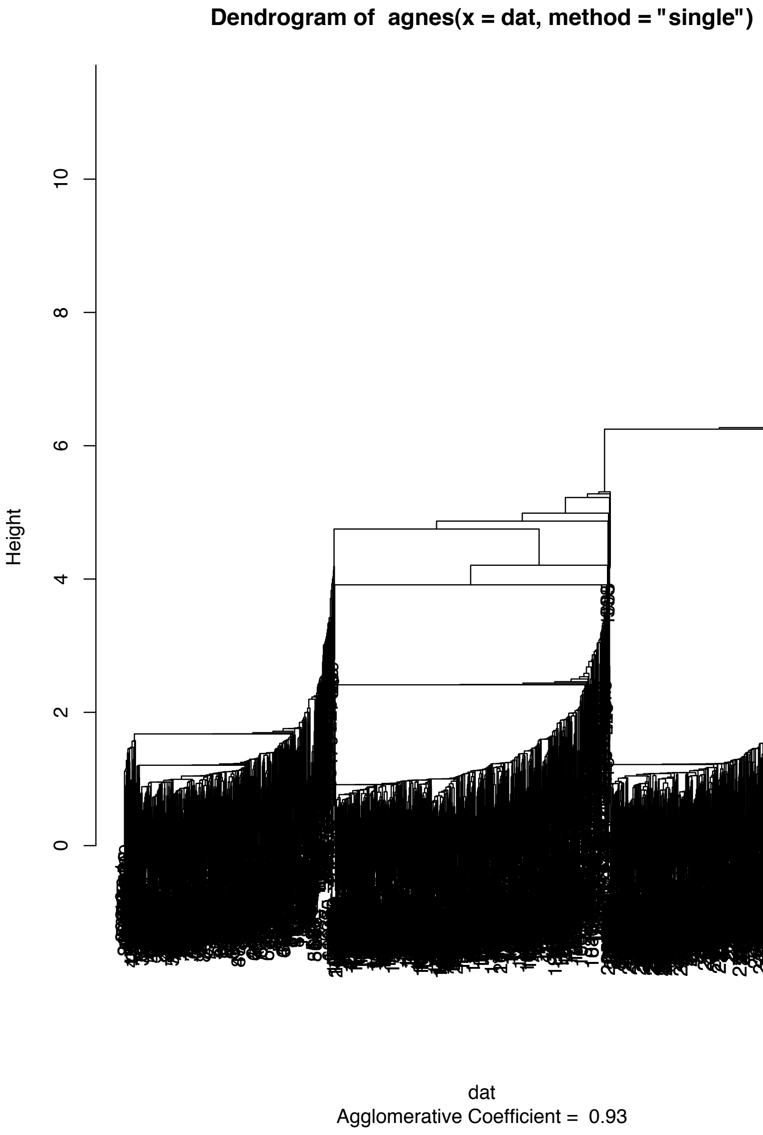
d15

$$\frac{d13+d14+d15+d23+d24+d25}{6}$$

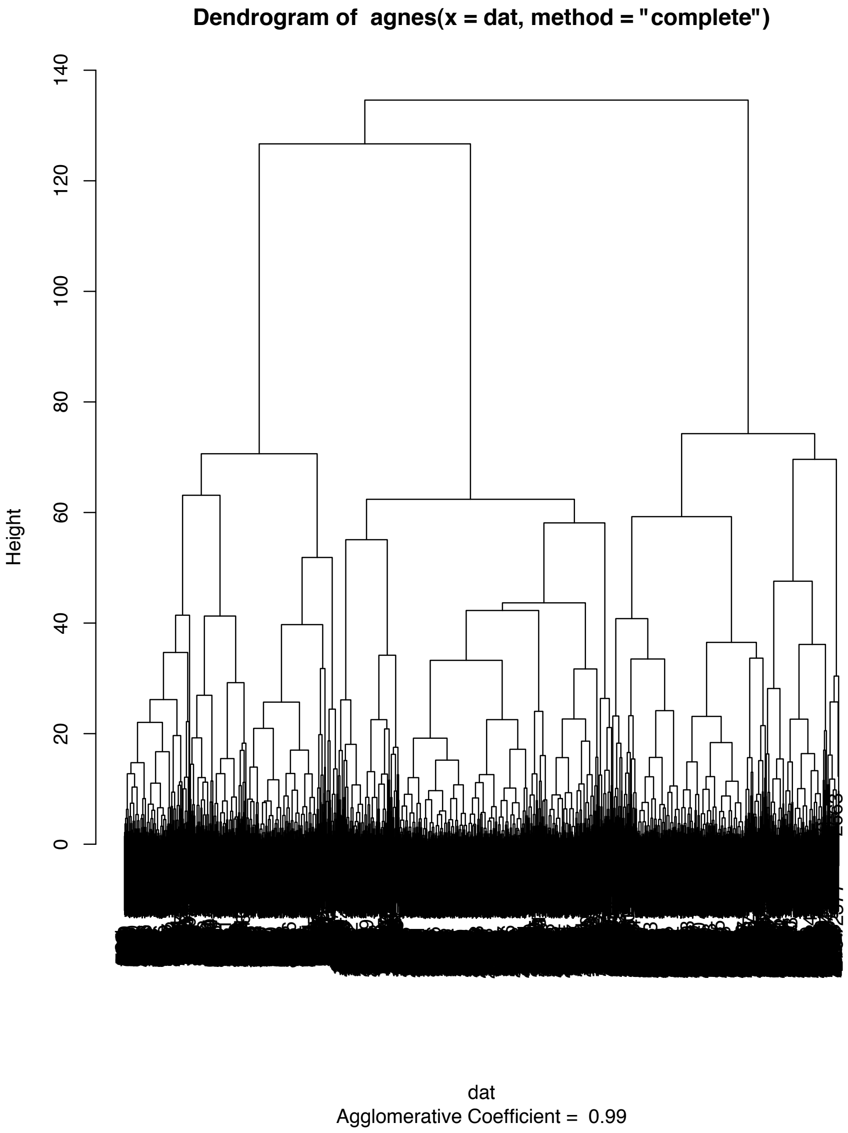
Hierarchical Clustering Example: Artificial Dataset



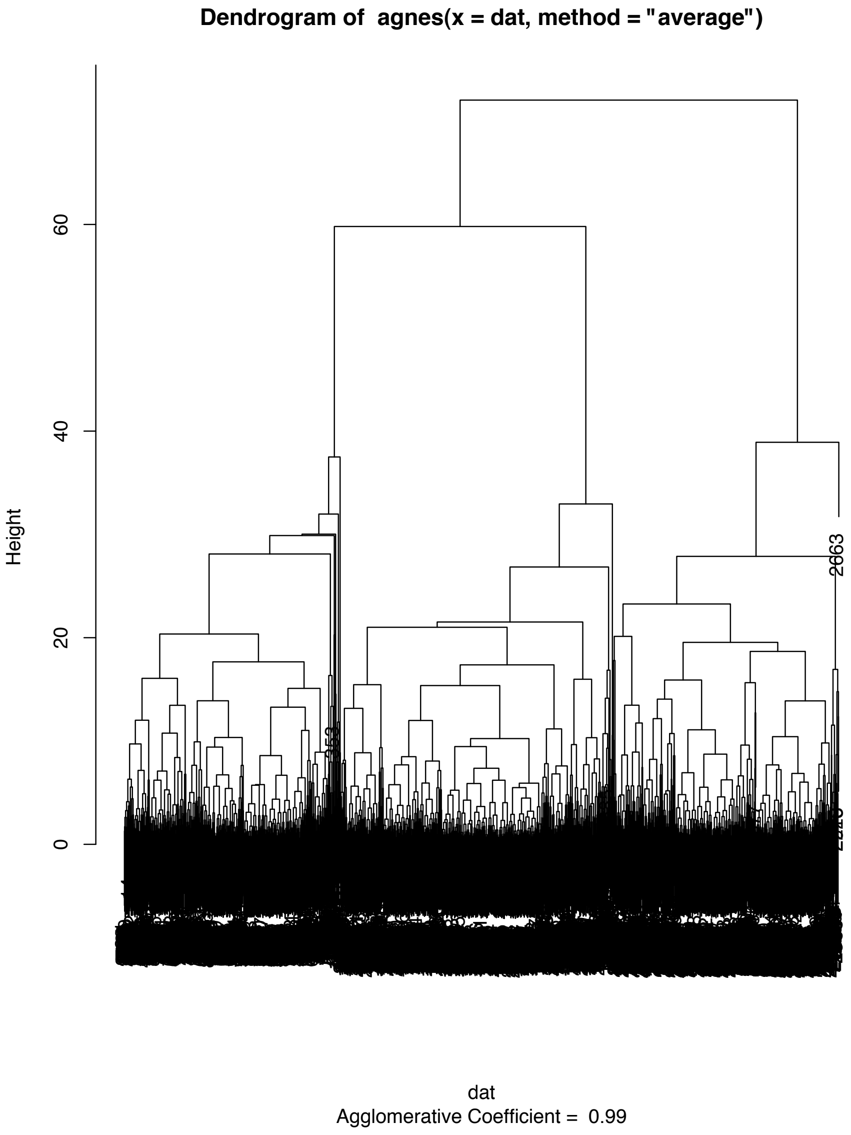
Hierarchical Clustering Example: Artificial Dataset



Hierarchical Clustering Example: Artificial Dataset



Hierarchical Clustering Example: Artificial Dataset



R Code

```
#start afresh
dat=xclara #3000 x 2
library(cluster)

#plot the data
plot(dat,type="n")
text(dat,labels=row.names(dat))

plot(agnes(dat,method="single"))
plot(agnes(dat,method="complete"))
plot(agnes(dat,method="average"))
```

Divisive Strategies

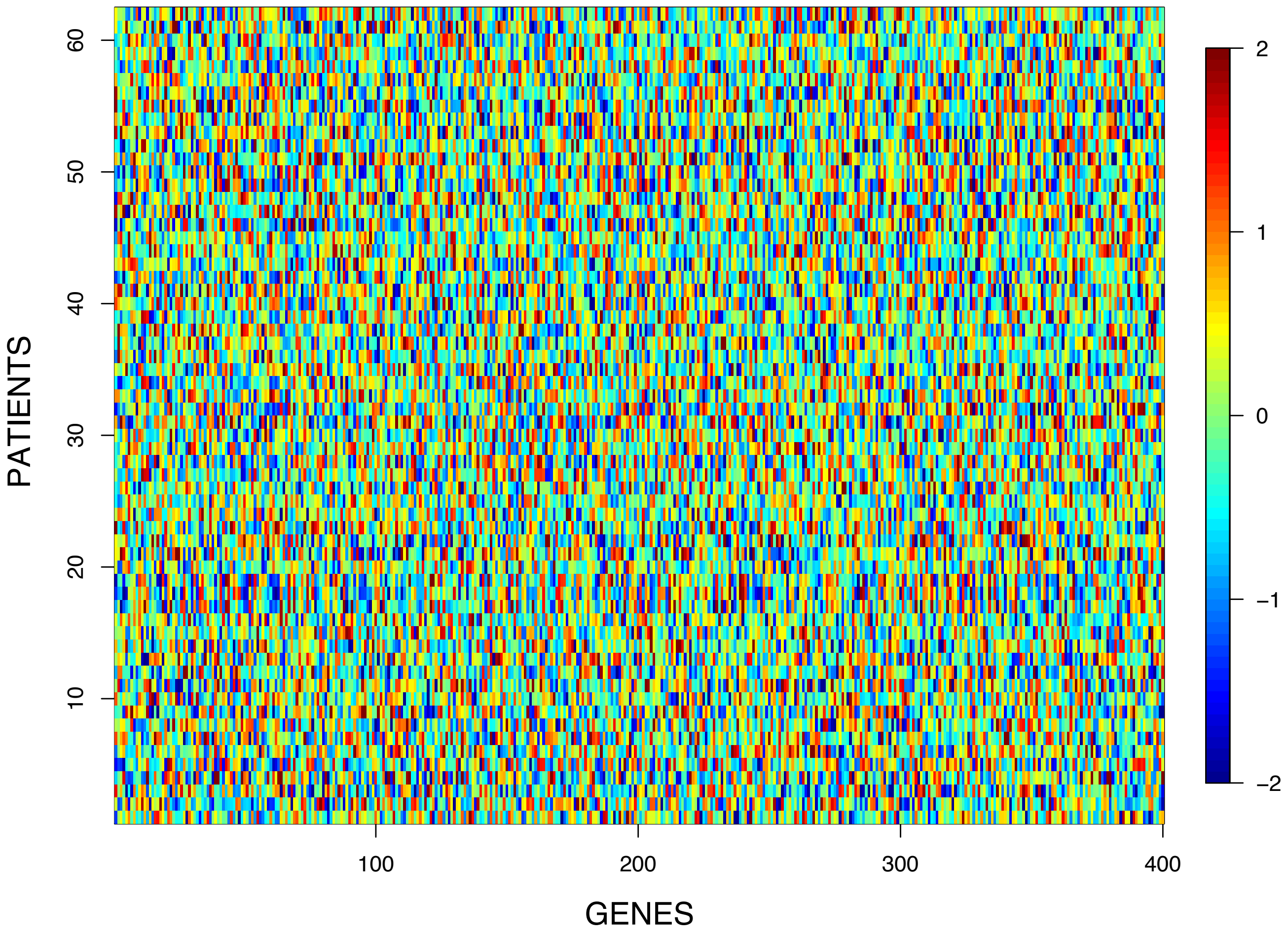
- Divisive strategies work in the opposite direction: starting with a single cluster which holds every observation, we recursively proceed as follows. For all clusters, partition the one that results in the greatest increase in dissimilarity that can arise when it is split in two. This recurses until each observation is a cluster on its own.
- If there are s observations in any cluster, there are $2^{s-1} - 1$ possible ways of partitioning it into two non-empty sets, a computationally infeasible task. Approximate methods are employed to tackle this problem which search though a subset of these possibilities.
- Divisive approaches are better than agglomerative approaches at showing structure near the top of the tree and so are preferred when interest is focused on partitioning data into a relatively small numbers of clusters.
- Divisive approaches are less known and so are much less used than agglomerative strategies.

Using Dendograms

- Different ways of measuring dissimilarity result in different trees.
- Dendograms are useful for getting a feel for the structure of high-dimensional data though they don't represent distances between observations well.
- Dendograms show hierarchical clusters with respect to increasing values of dissimilarity between clusters, cutting a dendogram horizontally at a particular height partitions the data into disjoint clusters which are represented by the vertical lines it intersects. Cutting horizontally effectively reveals the state of the clustering algorithm when the dissimilarity value between clusters is no more than the value cut at.
- Despite the simplicity of this idea and the above drawbacks, hierarchical clustering methods provide users with interpretable dendograms that allow clusters in high-dimensional data to be better understood.

Example: Lymphoma Gene Expression Data

Gene expression values taken of 4026 genes of 62 patients in a lymphoma cancer study. Color coded expression values for 500 randomly chosen genes look as follows.



Example: Lymphoma Gene Expression Data

Figure was generated by `R` code:

```
load(file="lymphoma.rda")
library(fields)

X <- lymphoma.x
X <- scale(X)
X <- X[,sample(1:ncol(X),400)]
for (k in 1:nrow(X)) X[k,] <- pmin(2,pmax(-2,X[k,]))

indn <- sample(1:nrow(X),nrow(X))

image.plot(1:ncol(X),1:nrow(X),t(X[indn,]),
        col=tim.colors(200),
        xlab="GENES", ylab="PATIENTS", cex.lab=1.4)
```

Example: Lymphoma Gene Expression Data

Now lets do hierarchical clustering with function `hclust`.

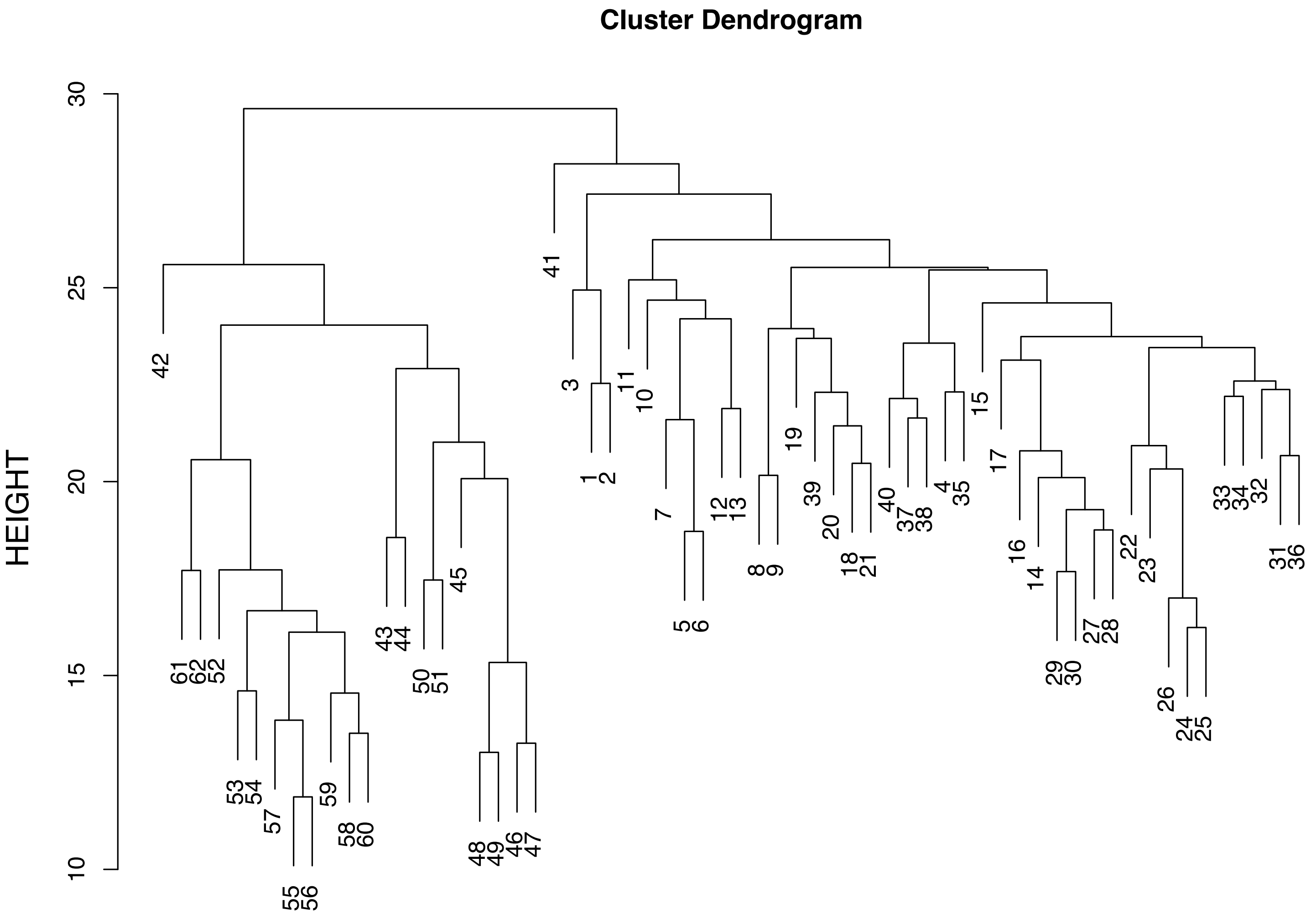
```
dd <- dist(t(X))
hh <- hclust(dd,method="average")

ddn <- dist(X)
hhn <- hclust(ddn,method="average")

plot(hh)
plot(hhn)
```

...or a bit more fancy

```
plot(hh,cex.lab=1.3,xlab="",ylab="HEIGHT",sub="")
plot(hhn,cex.lab=1.3,xlab="",ylab="HEIGHT",sub="")
```

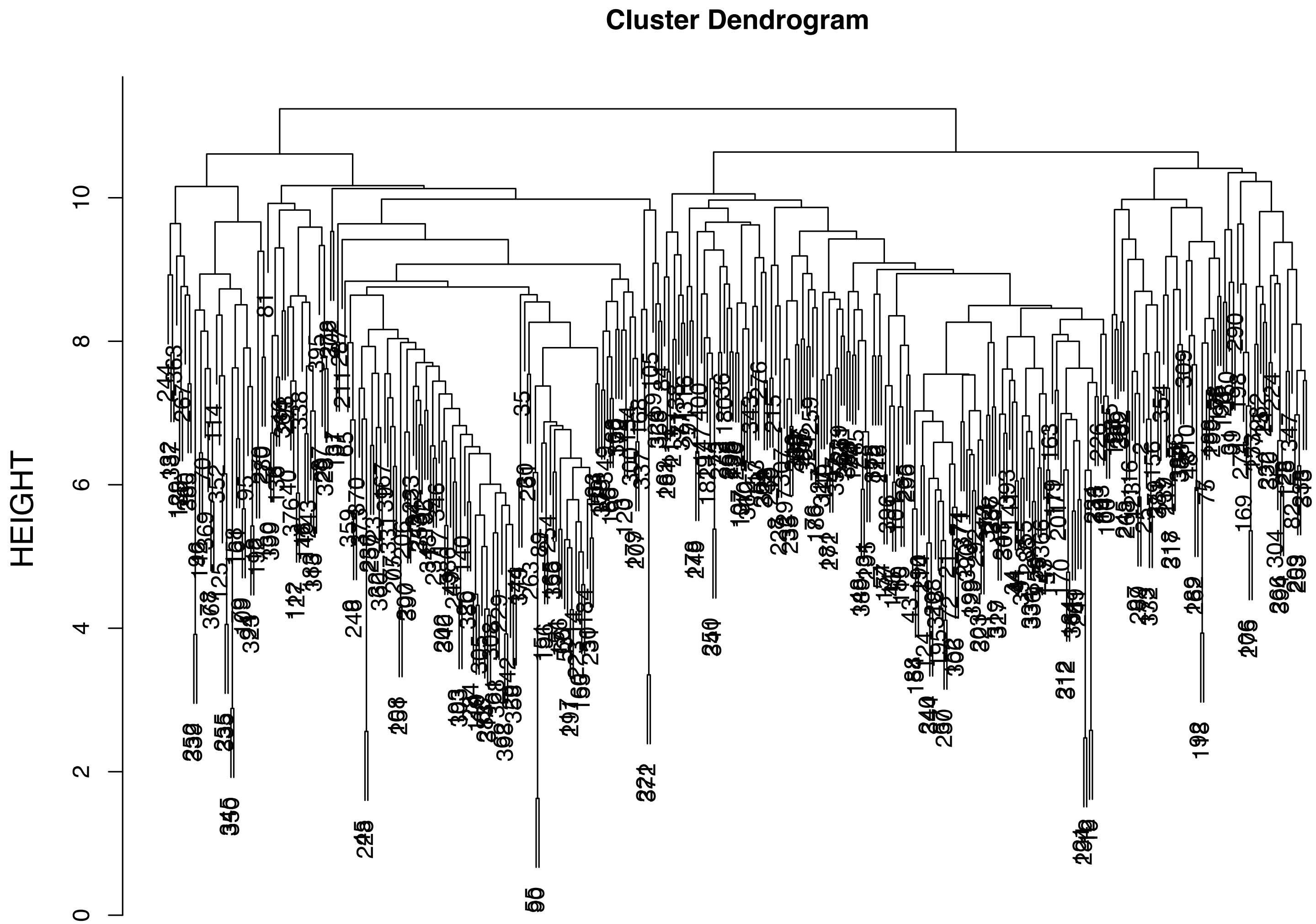
Example: Lymphoma Gene Expression Data

Using hierarchical clustering with average linkage for the 62 patients yields:



Example: Lymphoma Gene Expression Data

Using hierarchical clustering on the genes (instead of patients):



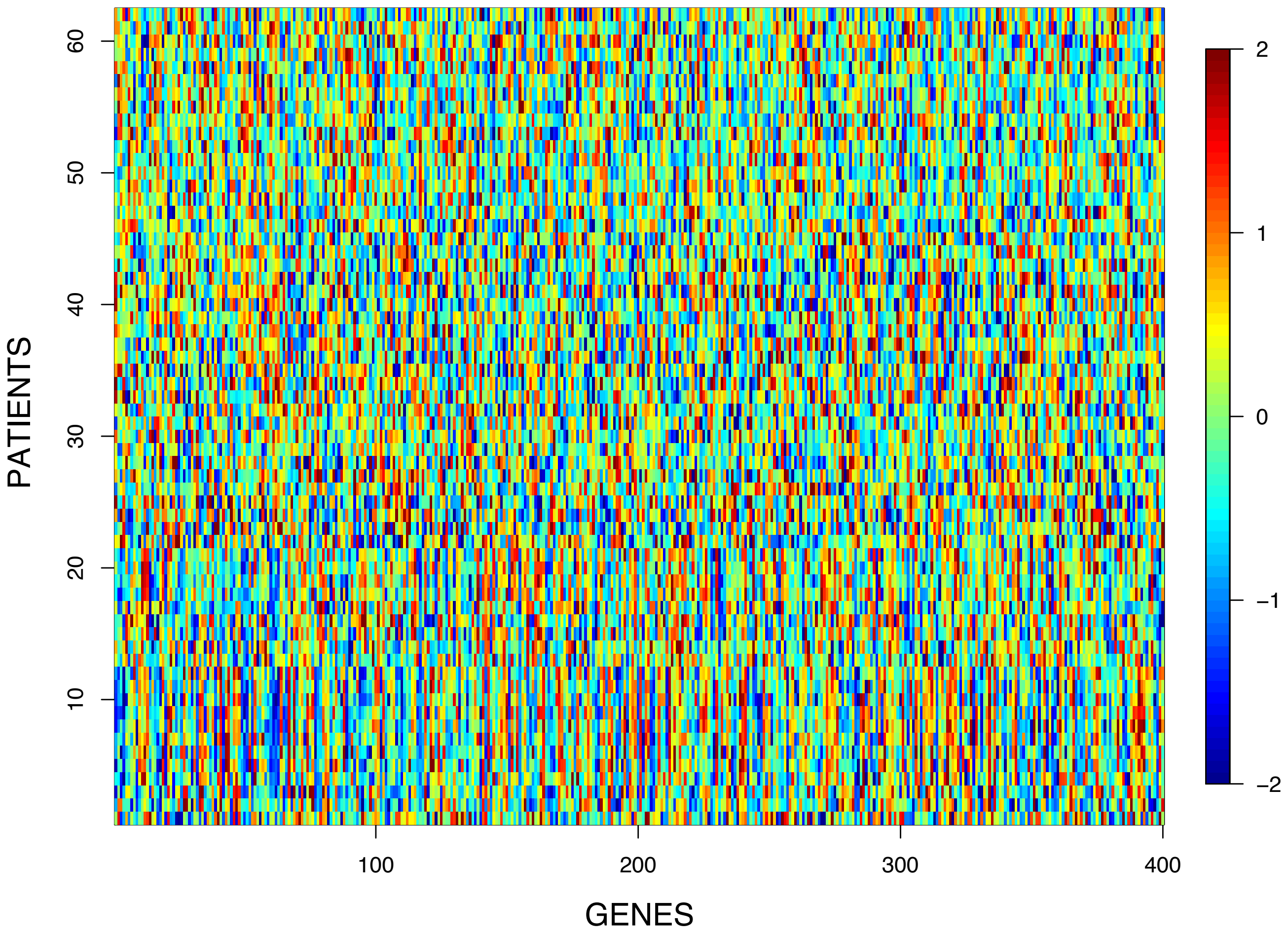
Example: Lymphoma Gene Expression Data

Can order the patients according to the ordering implied by `hclust`.

```
ord <- hh$order
ordn <- hhn$order

image.plot(1:ncol(X),1:nrow(X),t(X[ordn,]),
    col=tim.colors(200),cex.lab=1.4,
    xlab="GENES",ylab="PATIENTS")

image.plot(1:ncol(X),1:nrow(X),t(X[ordn,ord]),
    col=tim.colors(200),cex.lab=1.4,
    xlab="GENES",ylab="PATIENTS")
for (k in 1:nrow(X))
    text( ncol(X)-200, k, labels= "",cex=0)
for (k in 1:nrow(X))
    mtext((lymphoma.y[ordn])[k],side=4,at=k,cex=1.1)
```

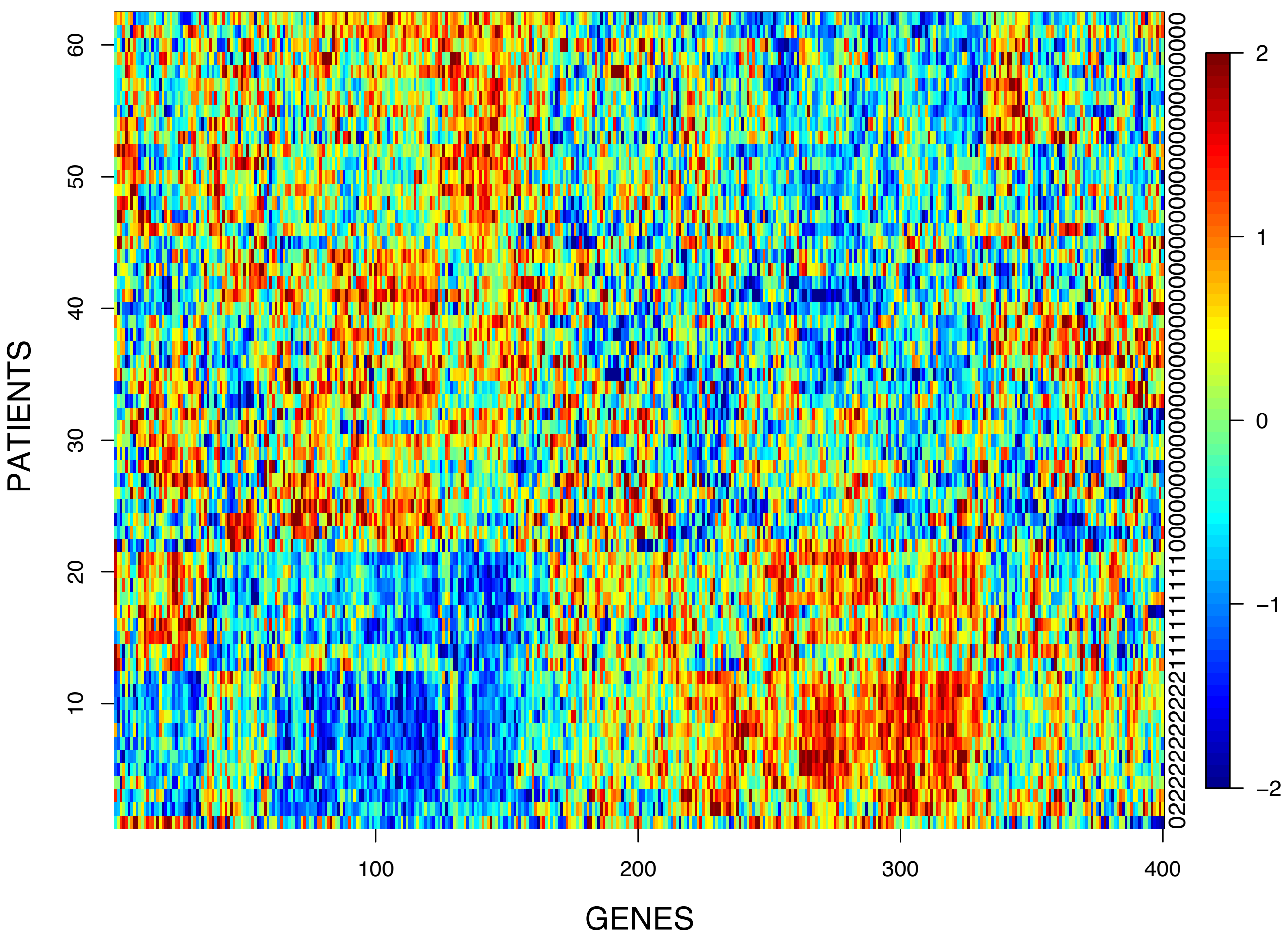
Example: Lymphoma Gene Expression Data

Using the ordering of patients implied by hierarchical clustering yields the following expression matrix.



Example: Lymphoma Gene Expression Data

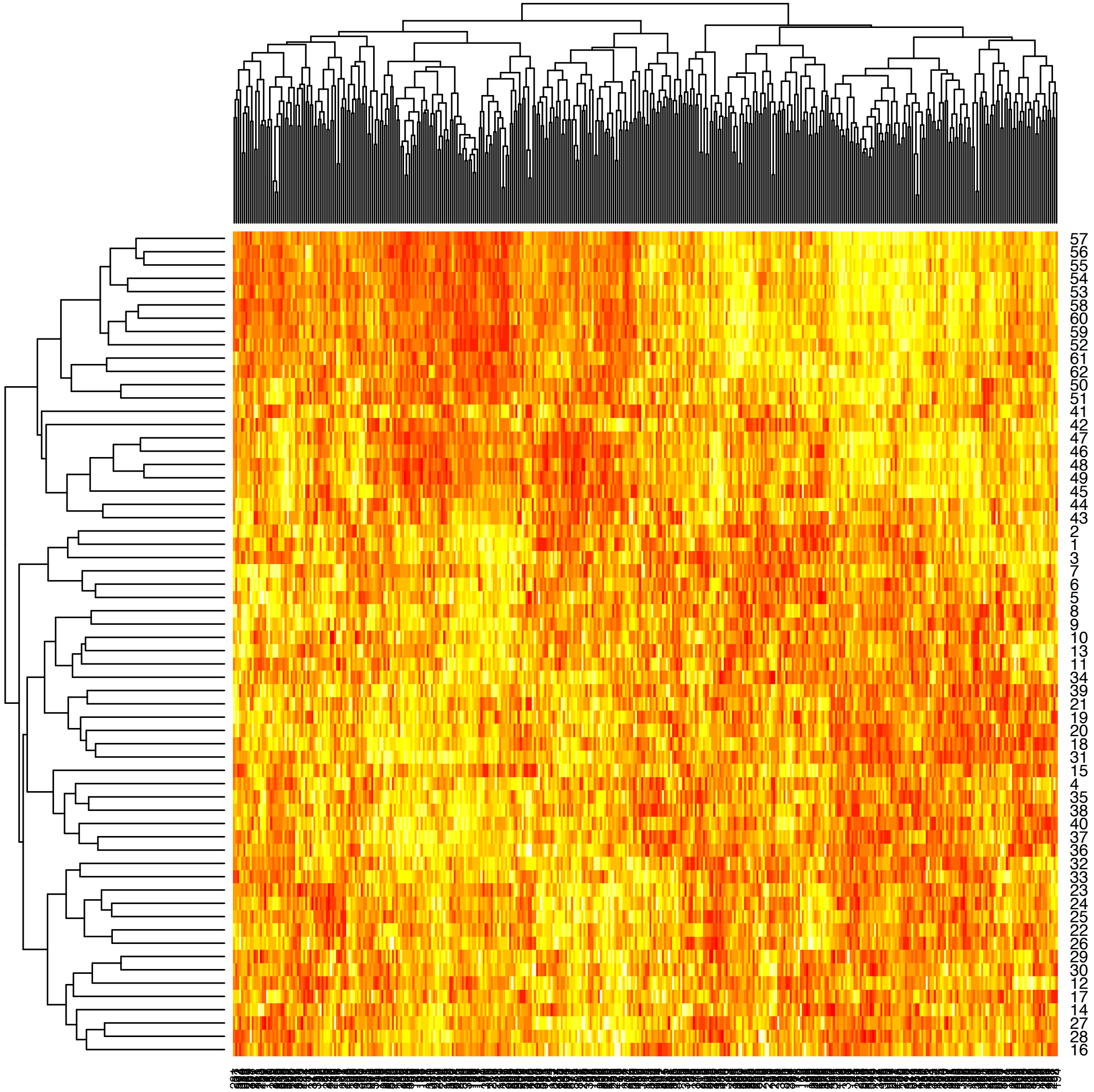
Using the ordering of patients **and** genes implied by hierarchical clustering yields the following expression matrix.



Different subtypes of lymphoma cancer (coded here as classes 0,1,2) are discovered in this way!

Example: Lymphoma Gene Expression Data

Or simply use command `heatmap(X)`.


57
56
55
54
53
58
60
59
52
61
62
50
51
41
42
47
46
48
49
45
44
43
2
1
3
7
6
5
8
9
10
13
11
34
39
21
19
20
18
31
15
4
35
38
40
37
36
32
33
23
24
25
22
26
29
30
12
17
14
27
28
16