Cluster Analysis and Applications to Genomics

Sandrine Dudoit

PB HLTH 240D — Spring 2005

Lecture 18

Acknowledgments

- Jane Fridlyand, Department of Epidemiology and Biostatistics and Comprehensive Cancer Center, UC San Francisco.
- Robert C. Gentleman, Department of Biostatistics, University of Washington, Seattle.
- Mark J. van der Laan, Division of Biostatistics, UC Berkeley.
- Katherine S. Pollard, Center for Biomolecular Science & Engineering, UC Santa Cruz.

References

www.stat.berkeley.edu/~sandrine

www.bepress.com/ucbbiostat, www.bepress.com/sagmb

1. S. Dudoit and J. Fridlyand (2002). A prediction-based resampling method to estimate the number of clusters in a dataset. *Genome Biology*, Vol. 3, No. 7, p. 0036.1–0036.21. [Tech. report #600, Stat. Dept.]
2. S. Dudoit and J. Fridlyand (2003). Bagging to improve the accuracy of a clustering procedure. *Bioinformatics*, Vol. 19, No. 9, p. 1090–1099. [Tech. report #600, Stat. Dept.]
3. L. Kaufman and P. J. Rousseeuw (1990). *Finding Groups in Data: An Introduction to Cluster Analysis*, Wiley-Interscience. [Details on PAM, AGNES, CLARA, silhouette widths.]
4. M. J. van der Laan and J. Bryan (2001). Gene expression analysis with the parametric bootstrap. *Biostatistics*, Vol 2, No. 4, p. 445–461. [Tech. report #86]
5. M. J. van der Laan and K. S. Pollard (2003). Hybrid clustering of gene expression data with visualization and the bootstrap. *Journal of Statistical Planning and Inference*, Vol. 117, p. 275–303. [Tech. report # 93]

References

6. M. J. van der Laan, K. S. Pollard, and J. Bryan (2003). A new partitioning around medoids algorithm. *Journal of Statistical Computation and Simulation*, Vol. 73, No. 8, p. 575–584. [Tech. report # 105]
7. K. S. Pollard and M. J. van der Laan (2002). A method to identify significant clusters in gene expression data. *Proceedings of SCI 2002*, Vol. II, p. 318–325. [Tech. report # 107]
8. K. S. Pollard and M. J. van der Laan (2005). Cluster analysis of genomic data. In R. C. Gentleman, V. J. Carey, S. Dudoit, W. Huber, and R. Irizarry (eds), *Bioinformatics and Computational Biology Solutions Using R and Bioconductor*, Springer-Verlag, New York. [Tech. report #167] [R]

N. B. Lecture notes on *Distances*, from PB HLTH 143, Spring 2004, are posted in the "References" section of the class website. The notes also provide an overview of principal component analysis (PCA) and multidimensional scaling (MDS).

Outline

- Overview of Cluster Analysis.
- Clustering Microarray Data.
- Clustering Methods
  - Partitioning Methods: PAM;
  - Hierarchical Methods: Agglomerative and Divisive;
  - Hybrid Methods: HOPACH.
- Estimating the Number of Clusters.
- Other Topics: Inference; Bagged Clustering.

Cluster Analysis vs. Class Prediction

Task. Assign observational units to classes on the basis of variables describing/characterizing these observations.

Cluster analysis. The classes are unknown a priori and need to be "discovered" from the data.
A.k.a. class discovery; unsupervised learning; unsupervised pattern recognition.

Class prediction. The classes are predefined and the task is to understand the basis for the classification from a set of labeled observations (learning set). This information is then used to predict the class of future observations.
A.k.a. classification; discriminant analysis; supervised learning; supervised pattern recognition.

Overview of Cluster Analysis

Associated with each observational unit in the population or sample of interest is a J–dimensional vector of variables describing this observation: $x = (x(j) : j = 1, \ldots, J)$.

Depending on the context, the entries $x(j)$ can be referred to as covariates, explanatory variables, features, measurements.

E.g. x could refer to a vector of microarray gene expression measures and clinical covariates (e.g., age, sex, blood pressure, censored survival time).

Overview of Cluster Analysis

The variables $x(j)$ can be (a mix of)

- quantitative/numerical
  - continuous — e.g., height, microarray measures;
  - discrete — e.g., SAGE counts;
- qualitative/categorical
  - ordinal — e.g., tumor grade;
  - nominal — e.g., ethnicity.

Overview of Cluster Analysis: Distances

Given vectors of explanatory variables, $x_1, \ldots, x_n$, recorded on n observations, the task is to identify clusters, i.e., groups or sets, of similar observations and/or similar explanatory variables.

E.g. Cluster target samples (= observations) and/or genes (= explanatory variables) in a microarray experiment.

Inherent in cluster analysis is a notion of distance or similarity between observations/features to be clustered.

Let $D = (d(x_i, x_j) : i, j = 1, \ldots, n)$ denote an $n \times n$ distance matrix of pairwise distances between the n observations to be clustered.

E.g. Euclidean distance matrix, one-minus-correlation matrix.

Overview of Cluster Analysis: Distances

Once a distance measure between individual observations has been chosen, one must often also define a measure of distance between clusters, or groups, of observations (e.g., average, single, and complete linkage agglomeration).

The choice of distance is important and can have a large impact on the results of cluster analyses.

In some cases, the Euclidean metric will be sensible, while in others, a distance based on correlations will be a better choice.

Subject matter knowledge is very helpful in selecting an appropriate distance for a given project.

Greater detail in the lecture on *Distances*.

Overview of Cluster Analysis: Distances

There are a number of different ways of defining a distance between two clusters or between a single observation and a cluster of observations.

Single linkage. The distance between two clusters is the minimum distance between two observations, one from each cluster.

Average linkage. The distance between two clusters is the average of all pairwise distances between the members of both clusters.

Complete linkage. The distance between two clusters is the maximum distance between two observations, one from each cluster.

Centroid distance. The distance between two clusters is the distance between their centroids. The definition of centroid may depend on the clustering algorithm being used, e.g., medoid, average, median.

Single linkage tends to lead to long, thin clusters, while average linkage leads to round clusters.

Overview of Cluster Analysis

Clustering (unsupervised) is in some sense a more difficult problem than class prediction (supervised). In general, all the issues that must be addressed for class prediction must also be addressed for clustering. In addition, with clustering,

- there is no learning set of labeled observations;
- the number of classes is usually unknown;
- implicitly, one must have already selected the relevant explanatory variables, standardization, and distance measure;
- the goals can be quite vague: *"Find some interesting and important clusters in my data"*;
- many of the algorithms that are appealing are computationally too complex to have exact solutions; often, only approximate solutions are available and reproducibility becomes an issue.

Overview of Cluster Analysis

Clustering involves several distinct steps.

1. Identify relevant explanatory variables.
2. Select a proper standardization and suitable measure of distance between observations to be clustered.
3. Apply an appropriate clustering algorithm.

The results of a clustering procedure can include both the number of clusters K (if not prespecified) and a set of n cluster labels $\in \{1, \ldots, K\}$, one for each of the n observations to be clustered.

Choices will depend on the question being asked, i.e., on the parameter of interest.

Overview of Cluster Analysis

One usually distinguishes between the following two broad types of clustering procedures.

- Hierarchical methods, either divisive or agglomerative. These methods provide a hierarchy of clusters, from the smallest set, where all observations are in one cluster, through to the largest set, where each observation is in its own cluster.
- Partitioning methods. These methods partition the observations into disjoint clusters (fuzzy versions available) and usually require specification of the number of clusters.

Most methods used in microarray data analysis are agglomerative hierarchical methods. In large part, this is due to the availability of efficient exact algorithms and to the appeal of dendrograms and heatmaps for visualizing microarray data.

Overview of Cluster Analysis: R Software

Package	Functions	Description
cclust		Convex clustering methods
class	SOM	Self-organizing maps
cluster	agnes	AGglomerative NESting
	clara	Clustering LARge Applications
	diana	DIvisive ANAlysis
	fanny	Fuzzy Analysis
	mona	MONothetic Analysis
	pam	Partitioning Around Medoids
e1071	bclust	Bagged clustering
	cmeans	Fuzzy C-means clustering
flexmix		Flexible mixture modeling
fpc		Fixed point clusters, clusterwise regression and discriminant plots
hopach	hopach, boothopach	Hierarchical Ordered Partitioning and Collapsing Hybrid
mclust		Model-based cluster analysis
stats	hclust, cophenetic	Hierarchical clustering
	heatmap	Heatmaps with row and column dendrograms
	kmeans	k-means

Specialized classes and methods (e.g., print, summary, and plot) are provided for handling clustering results.

Case Study: Golub et al. (1999) Leukemia Microarray Dataset

The Golub et al. (1999) leukemia microarray dataset is used as a case study. These data are included in the Bioconductor R data package golubEsets. Only consider data on the 38 learning set patients. These data are stored in the object `golubTrain` of class *exprSet*.

Following Golub et al. (1999), apply the following three pre-processing steps:
(i) *thresholding*, floor of 100 and ceiling of 16,000;
(ii) *filtering*, exclusion of genes with $\max / \min \leq 5$ or $(\max - \min) \leq 500$, where max and min refer, respectively, to the maximum and minimum intensities for a particular gene across the 38 mRNA target samples;
(iii) *base-2 logarithmic transformation*.

Case Study: Golub et al. (1999) Leukemia Microarray Dataset

```
> library(golubEsets)
> library(genefilter)
>
> X<-exprs(golubTrain)
>
> # Thresholding
> X[X<100]<-100
> X[X>16000]<-16000
>
> # Filtering
> mmfilt <- function(r=5, d=500, na.rm=TRUE) {
+ function(x) {
+   minval <-  min(x, na.rm=na.rm)
+   maxval <-  max(x, na.rm=na.rm)
+   (maxval/minval > r) && (maxval-minval > d)
+   }
+ }
> mmfun <- mmfilt()
> ffun <- filterfun(mmfun)
> good <- genefilter(X, ffun )
> sum(good) ## Should get 3051
[1] 3051
> X <- X[good,]
```


Case Study: Golub et al. (1999) Leukemia Microarray Dataset

```
> # Log transformation
> X <- log2(X)
>
> # Class labels
> Y <- golub$ALL.AML
> Y<-paste(golubTrain$ALL.AML,golubTrain$T.B.cell)
> Y<-sub("NA","",Y)
> table(Y)
Y
ALL B-cell ALL T-cell       AML
        19          8         11
>
> # Numeric class labels
> Ynum<-rank(unique(Y))[factor(Y)]
> Ynum
 [1] 1 2 2 1 1 2 1 1 2 2 2 1 1 2 1 1 1 1 1 1 1 1 2 1 1 1 1 3 3 3 3 3 3 3 3 3 3 3
```


Case Study: Golub et al. (1999) Leukemia Microarray Dataset

```
> # New exprSet object
> golub<-golubTrain[good,]
> gnames<-dimnames(X)[[1]]
> slot(golub,"exprs")<-X
> golub
Expression Set (exprSet) with
        3051 genes
        38 samples
                phenoData object with 11 variables and 38 cases
         varLabels
                Samples: Sample index
                ALL.AML: Factor, indicating ALL or AML
                BM.PB: Factor, sample from marrow or peripheral blood
                T.B.cell: Factor, T cell or B cell leuk.
                FAB: Factor, FAB classification
                Date: Date sample obtained
                Gender: Factor, gender of patient
                pctBlasts: pct of cells that are blasts
                Treatment: response to treatment
                PS: Prediction strength
                Source: Source of sample
```


Clustering Microarray Data

Expression measures on G genes/probe sequences (= explanatory variables) for n microarrays/target samples (= observations)

$$X_{G\times n} = \begin{array}{c} \text{Microarrays} \\ \left[\begin{array}{cccc} X(1,1) & X(1,2) & \dots & X(1,n) \\ X(2,1) & X(2,2) & \dots & X(2,n) \\ \vdots & \vdots & \ddots & \vdots \\ X(G,1) & X(G,2) & \dots & X(G,n) \end{array}\right] \end{array} \text{Genes}$$

$X(g,i)$ = expression measure of gene g for microarray i.

Clustering Microarray Data

Some clustering and class prediction methods require complete data, i.e., expression measures on all G genes for each of the n microarrays.

However, missing data are a common problem in microarray (and other) experiments.

A simple, intuitive imputation approach is k-nearest neighbor imputation.

Applications of k-nearest neighbor imputation to microarray data are discussed in Troyanskaya et al. (2001), *Bioinformatics*.

The method is implemented in the R software package impute available on CRAN.

Clustering Microarray Data

In k-nearest neighbor imputation, one first computes a matrix of pairwise distances between genes, by ignoring the missing values (see help file for `dist` function for details on missing value handling).

Then, for imputing the expression measure $X(g, i)$ of gene g for microarray i, one identifies the k-nearest neighbors of gene g having data for microarray i.

One can impute $X(g, i)$ by the average of the expression measures of these k neighboring genes for microarray i. One can also use weighted averages, with distance-based weights, or the median of the expression measures.

Clustering Microarray Data

- One can cluster genes and/or microarrays/target samples.
- Clustering leads to appealing graphical displays and summaries of data, e.g., heatmaps, dendrograms.
- Clustering may strengthen the signal when averages are taken within clusters of genes (Eisen et al., 1998).
- Clustering can be helpful for identifying gene expression patterns in time or space.
- Clustering is useful, perhaps essential, when seeking new subclasses of cell samples (tumors classification, etc).
- Clustering can be used for quality assessment: relate microarray/gene clustering results to experimental variables, such as print-run, mRNA amplification protocol, experimenter, lab, etc.

Clustering Microarray Data

Cluster genes (rows)

- to identify groups of co-regulated genes, e.g., using a large number of yeast experiments;
- to identify spatial or temporal gene expression patterns;
- to reduce redundancy in prediction (cf. variable selection);
- to detect experimental artifacts;
- for visualization purposes.

Transformations of the matrix of expression measures using linear (or other) models may be useful in this context:
genes $\times$ microarrays/target samples $\Longrightarrow$ genes $\times$ estimated effects.

Greater detail in the lecture on *Distances*.

Clustering Microarray Data

Cluster microarrays/target samples (columns)

- to identify new classes of biological samples, e.g., new tumor classes, new cell types;
- to detect experimental artifacts;
- for visualization purposes.

Cluster both rows and columns at once.

Clustering Microarray Data

Clustering can be employed for quality assessment purposes. The clusters that obtain from clustering microarrays and/or genes should be compared with different experimental conditions, such as:

- print-run or production order of the microarrays;
- batch of reagents;
- mRNA amplification protocol;
- technician;
- plate origin of probe sequences, etc.

Any relationships observed here should be considered as a potentially serious source of bias.

Clustering Microarray Data

Most efforts to date have involved clustering microarrays/target samples or genes based only on microarray expression measures.

However, there are likely benefits from incorporating other data into the analysis, such as clinical covariates/outcomes (age, sex, treatment, survival) and biological metadata (GO annotation, PubMed literature).

Clustering Microarray Data

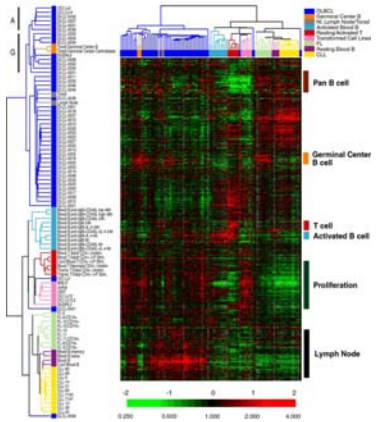

Figure 1: *Alizadeh et al. (2000).* Distinct types of diffuse large B-cell lymphoma identified by gene expression profiling.

Clustering Methods

Preliminary questions

- Which explanatory variables (i.e., which genes or microarrays)?
- Which transformation/standardization?
- Which distance function $d(\cdot, \cdot)$?
- Which clustering algorithm?

Answers will depend on the biological question, i.e., on the parameter of interest.

Clustering Methods

- How many clusters?
- How reliable are the clustering results?
  - Statistical inference: distributional properties of clustering results.
  - Assessing the strength/confidence of cluster assignments for individual observations.
  - Assessing cluster homogeneity.

Partitioning Methods

- Partition the data into a prespecified number of clusters K, that are mutually exclusive and exhaustive.
- Often involves iteratively reallocating observations to clusters until some criterion is optimized, e.g., minimize within-clusters sums-of-squares or sum of distances from nearest medoid.
- Examples:
  - k-means, extension to fuzzy k-means;
  - Partitioning around medoids (PAM) (Kaufman & Rousseeuw, 1990);
  - Self-organizing maps (SOM) (Kohonen, 2001);
  - Model-based clustering, e.g., Gaussian mixtures (Fraley & Raftery, 1998, 2000; McLachlan et al., 2001).

Partitioning Methods: PAM

Partitioning around medoids (PAM) of Kaufman & Rousseeuw (1990) is a partitioning method which operates on a distance matrix, $D = (d(x_i, x_j))$, of pairwise distances between the n observations to be clustered.

For a prespecified number of clusters K, the PAM procedure is based on the search for K representative observations, or medoids, among the observations to be clustered.

After finding a set of K medoids, K clusters are constructed by assigning each observation to the cluster with the nearest medoid.

Partitioning Methods: PAM

The goal is to find a set of K medoids, $\mathcal{M}_K = \{m_1, \ldots, m_K\} \subseteq \{x_1, \ldots, x_n\}$, which minimizes the sum of the distances of the observations to their closest medoid. That is, the optimal set of medoids $\mathcal{M}_K$ is defined as

$$\mathcal{M}_K \equiv \text{argmin}_{\mathcal{M} \subseteq \{x_1, \ldots, x_n\}, |\mathcal{M}| = K} \sum_{i=1}^{n} \min_{m \in \mathcal{M}} d(x_i, m).$$

That is, the PAM algorithm seeks to minimize the objective function $f : \mathcal{M} \to I\!R$ defined by

$$f(\mathcal{M}) \equiv \sum_{i=1}^{n} \min_{m \in \mathcal{M}} d(x_i, m),$$

where $\mathcal{M} \subseteq \{x_1, \ldots, x_n\}$ and $|\mathcal{M}| = K$.

Partitioning Methods: PAM

There are $\binom{n}{K} = n!/(K!(n-K)!)$ possible choices for the K medoids.

The PAM algorithm first looks for a good initial set of medoids (`build` phase). It then finds a local minimum for the objective function f, that is, a solution such that there is no single switch of an observation with a medoid that will decrease the objective function (`swap` phase).

N. B. PAM can be applied to general data types and tends to be more robust than k-means.

Partitioning Methods: Silhouette Widths

Rousseeuw (1987) suggests a graphical display, the silhouette plot, which can be used to: (i) select the number of clusters and (ii) assess how well individual observations are clustered.

The silhouette width of observation i is defined as

$$sil(i) \equiv \frac{b(i) - a(i)}{\max(a(i), b(i))} \in [-1, 1],$$

where
$a(i)$ denotes the average distance between the ith observation and all other observations in the cluster to which i belongs;
$b(i)$ denotes the minimum average distance of the ith observation to observations in other clusters.

Partitioning Methods: Silhouette Widths

Intuitively, observations with large silhouette widths (almost one) are well-clustered, those with silhouette widths around zero tend to lie between clusters, and those with negative silhouette widths are probably placed in the wrong cluster.

N. B. Silhouette widths may be computed for the results of any partitioning clustering algorithm.

Partitioning Methods: Silhouette Widths

For a given number of clusters K, the overall average silhouette width for a clustering is simply the average of $sil(i)$ over all observations i,

$$\overline{sil}_K \equiv \frac{1}{n} \sum_{i=1}^{n} sil(i).$$

Kaufman & Rousseeuw (1990) suggest estimating the number of clusters K by that which gives the largest average silhouette width, that is, by

$$K^* \equiv \text{argmax}_K \, \overline{sil}_K.$$

PAM: Case Study in R

```
> library(cluster)
> set.seed(99)
>
> # Correlation-based distance matrix
> corr<-cor(exprs(golub))
> d<-1-corr
> dimnames(d)<-list(as.vector(Y),as.vector(Y))
>
> # PAM, K=2
> pam2 <- pam(as.dist(d), k=2, diss=TRUE)
> table(pam2$clustering, Y)
   Y
    ALL B-cell ALL T-cell AML
  1 16          7          0
  2  3          1         11
> # PAM, K=3
> pam3 <- pam(as.dist(d), k=3, diss=TRUE)
> table(pam3$clustering, Y)
   Y
    ALL B-cell ALL T-cell AML
  1 16          0          0
  2  3          1         11
  3  0          7          0
```


PAM: Case Study in R

```
> # Graphical summaries
>
> pdf(file="pamGolub%d.pdf", onefile=FALSE)
>
> clusplot(d, pam2$clustering, diss=TRUE, labels=3, col.p=1, col.txt=rank(unique(Y))[factor
>
> plot(pam2,which.plots=2,main="Golub et al. (1999). Silhouette plot for PAM \n Correlation-
>
> clusplot(d, pam3$clustering, diss=TRUE, labels=3, col.p=1, col.txt=rank(unique(Y))[factor
>
> plot(pam3,which.plots=2,main="Golub et al. (1999). Silhouette plot for PAM \n Correlation-
>
> dev.off()
```


PAM: Case Study in R

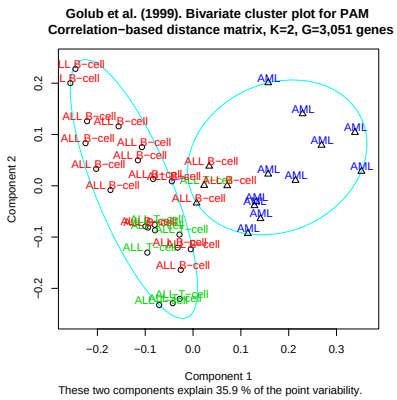

Figure 2: *Golub et al. (1999) leukemia dataset.* Bivariate cluster plot for PAM, correlation-based distance matrix, $K = 2$, $G = 3,051$ genes.

PAM: Case Study in R

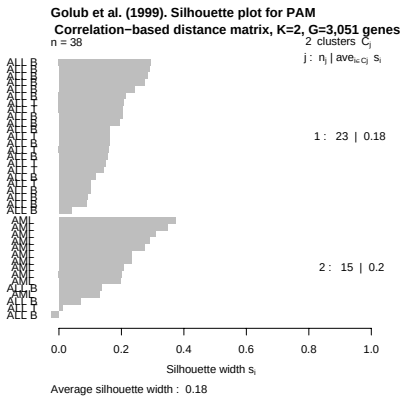

Figure 3: *Golub et al. (1999) leukemia dataset.* Silhouette plot for PAM, correlation-based distance matrix, $K = 2$, $G = 3,051$ genes.

PAM: Case Study in R

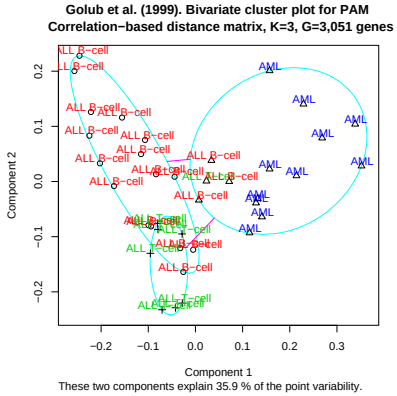

Figure 4: *Golub et al. (1999) leukemia dataset.* Bivariate cluster plot for PAM, correlation-based distance matrix, $K = 3$, $G = 3,051$ genes.

PAM: Case Study in R

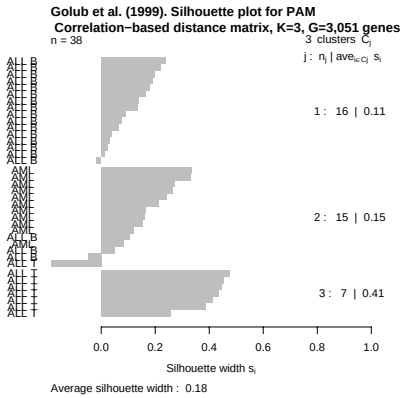

Figure 5: *Golub et al. (1999) leukemia dataset.* Silhouette plot for PAM, correlation-based distance matrix, $K = 3$, $G = 3,051$ genes.

PAM: Case Study in R

```
> # Average silhouette widths for K = 2, ...,7
> K<-2:7
> avgSil<-rep(NA, length(K))
> for(k in K)
+   avgSil[k-1]<-pam(as.dist(d), k=k, diss=TRUE)$silinfo$avg.width
>
> # Graphical summaries
> pdf(file="pamAvgSilGolub%d.pdf", onefile=FALSE)
> barplot(avgSil, names.arg=K, xlab="Number of clusters, K", ylab="Average silhouette width")
> plot(K, avgSil, pch=16, xlab="Number of clusters, K", ylab="Average silhouette width")
> dev.off()
>
> # PAM, K=4
> pam4 <- pam(as.dist(d), k=4, diss=TRUE)
> table(pam4$clustering, Y)
   Y
    ALL B-cell ALL T-cell AML
  1  8          0          0
  2  2          1         11
  3  0          7          0
  4  9          0          0
```

PAM: Case Study in R

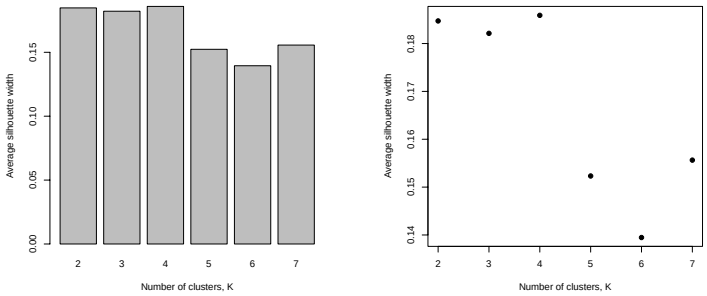

Figure 6: *Golub et al. (1999) leukemia dataset.* Average silhouette widths for $K = 2, 3, \ldots, 7$ clusters, PAM, correlation-based distance matrix, $G = 3,051$ genes.

Partitioning Methods: PAMSIL

PAMSIL (van der Laan, Pollard, & Bryan, 2003).

Replace the PAM criterion or objective function (based on the sum of the distances from the nearest medoid) with the average silhouette width criterion.

	PAM	**PAMSIL**
Criterion	$-\sum_i \min_m d(x_i, m)$	$\sum_i sil(i)$
Algorithm	Steepest ascent	Steepest ascent
Starting values	Build	PAM, random
K	Given or data-adaptive	Given or data-adaptive
Overall performance	"Robust"	"Efficient"
Splitting large clusters	Yes	No
Outliers	Ignore	Identify

Hierarchical Methods

- Hierarchical clustering methods produce a hierarchy or tree of nested clusters.
- They do not require prespecifying the number of clusters K.
- They provide a partition for each K, by "cutting" the tree at different levels.
- One distinguishes between
  - agglomerative, i.e., bottom-up, clustering;
  - divisive, i.e., top-down, clustering.

Hierarchical Methods: Agglomerative Methods

- Start with each observation in its own cluster.
- At each step, merge the two closest clusters using a measure of between-cluster distance.
  - Average linkage: average of pairwise distances;
  - Single linkage: minimum of pairwise distances;
  - Complete linkage: maximum of pairwise distances.

Greater detail in the lecture on *Distances*.

Hierarchical Methods: Divisive Methods

- Start with only one cluster.
- At each step, split clusters into two (or more) clusters.
- Advantages: Obtain the main structure of the data, i.e., focus on upper levels of the tree.
- Disadvantages: Computational difficulties when considering all possible divisions into two (or more) groups.
- Examples
  - Self-organizing tree algorithm (SOTA) (Dopazo & Carazo, 1997);
  - Hierarchical ordered partitioning and collapsing hybrid (HOPACH) (van der Laan & Pollard, 2003; Pollard & van der Laan, 2005);
  - Divisive analysis (DIANA) (Kaufman & Rousseeuw, 1990).

Hierarchical Methods: Dendrograms

Dendrograms are often used to visualize the nested sequence of clusters resulting from hierarchical clustering.

While dendrograms are quite appealing because of their apparent ease of interpretation, they can be misleading.

Firstly, the dendrogram corresponding to a given hierarchical clustering is not unique. For each merge, one needs to specify which subtree should go on the left and which on the right
$\Longrightarrow$ there are $2^{(n-1)}$ different dendrograms.

The default in the R function `hclust` (`stats` package) is to order the subtrees so that the tighter cluster is on the left.

Hierarchical Methods: Dendrograms

A second, and perhaps less recognized shortcoming of dendrograms, is that they impose structure on the data, instead of *revealing* structure in these data.

Such a representation is valid only to the extent that the pairwise distances possess the hierarchical structure imposed by the clustering algorithm.

Hierarchical Methods: Dendrograms

The cophenetic correlation coefficient can be used to measure how well the hierarchical structure from the dendrogram represents the actual distances.

This measure is defined as the correlation between the $n(n-1)/2$ pairwise between-observation distances and their cophenetic distances from the dendrogram, i.e., the between-cluster distances at which two observations are first joined together.

The cophenetic distances have a great deal of structure, e.g., there are many ties.

R function: `cophenetic` in `stats` package.

Hierarchical Methods: Case Study in R

```
> # Correlation-based distance matrix
> corr<-cor(exprs(golub))
> d<-1-corr
> dimnames(d)<-list(as.vector(Y),as.vector(Y))
>
> # Hierarchical clustering
> hc <- hclust(as.dist(d), method="average")
> hc
Call:
hclust(d = as.dist(d), method = "average")
Cluster method   : average
Number of objects: 38
>
> # Dendrogram
> pdf("hclustGolub.pdf")
> plot(hc, labels=Y, main="Golub et al. (1999). Hierarchical clustering dendrogram",sub="Ave
> dev.off()
>
> # Cophenetic correlation coefficient
> round(cor(cophenetic(hc),as.dist(d)),2)
[1] 0.74
```


Hierarchical Methods: Case Study in R

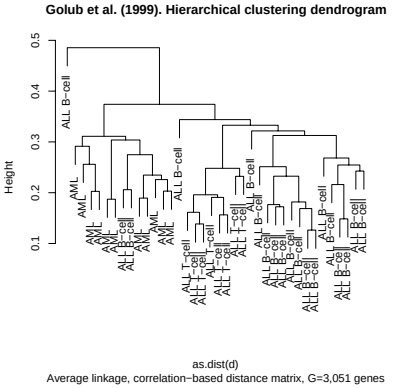

Figure 7: *Golub et al. (1999) leukemia dataset.* Dendrogram for average linkage agglomerative hierarchical clustering, correlation-based distance matrix, $G = 3,051$ genes.

Heatmaps

The `heatmap` function from the `stats` package produces a heatmap, i.e., a pseudo-color image, of the entries of a matrix. Dendrograms are added to the left side and to the top of the image, displaying clustering results for the rows and columns of the matrix, respectively.

The function builds on the `image` function. The default clustering method is complete linkage hierarchical clustering using the Euclidean distance.

Many arguments are available to customize the display. In particular, one can use the `col` argument of the `image` function to specify a color palette.

Heatmaps: Case Study in R

```
> # Select 50 genes with largest coefficient of variation
> X<-exprs(golub)
> cv<-apply(X,1, function(z) sd(z)/mean(z))
> Xsub<-X[rev(order(cv))[1:50],]
> dimnames(Xsub)[[2]]<-Y
>
> # Heatmaps
> args(heatmap)
function (x, Rowv = NULL, Colv = if (symm) "Rowv" else NULL,
    distfun = dist, hclustfun = hclust, reorderfun = function(d,
        w) reorder(d, w), add.expr, symm = FALSE, revC = identical(Colv,
        "Rowv"), scale = c("row", "column", "none"), na.rm = TRUE,
    margins = c(5, 5), ColSideColors, RowSideColors, cexRow = 0.2 +
        1/log10(nr), cexCol = 0.2 + 1/log10(nc), labRow = NULL,
    labCol = NULL, main = NULL, xlab = NULL, ylab = NULL, keep.dendro = FALSE,
    verbose = getOption("verbose"), ...)
>
> pdf(file="heatmapGolub%d.pdf", onefile=FALSE)
> heatmap(Xsub,ColSideColors=c("red","green","blue")[rank(unique(Y))[factor(Y)]])
> heatmap(Xsub,col=rainbow(30),ColSideColors=c("red","green","blue")[rank(unique(Y))[factor(
> dev.off()
```

Heatmaps: Case Study in R

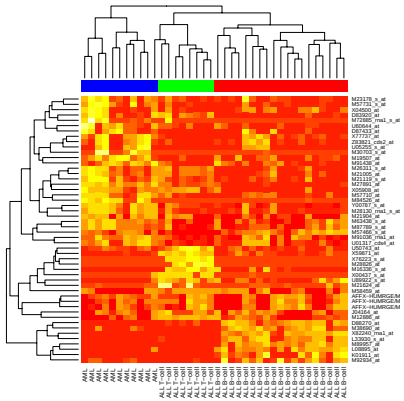

Figure 8: *Golub et al. (1999) leukemia dataset.* Heatmap for the 50 genes with largest coefficient of variation (among 3,051), Euclidean distance, complete linkage hierarchical clustering.

Heatmaps: Case Study in R

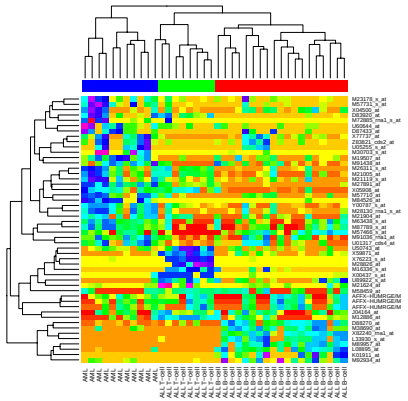

Figure 9: *Golub et al. (1999) leukemia dataset.* Heatmap for the 50 genes with largest coefficient of variation (among 3,051), Euclidean distance, complete linkage hierarchical clustering.

Partitioning vs. Hierarchical Methods

- Partitioning
  - Advantages: Provide clusters that (approximately) satisfy an optimality criterion.
  - Disadvantages: Often need to specify the number of clusters K; possibly long computation time.
- Hierarchical
  - Advantages: Fast computation (for agglomerative clustering).
  - Disadvantages: Can be rigid, i.e., cannot correct for erroneous decisions made early in the clustering process.

Hybrid Methods: HOPACH

Hierarchical ordered partitioning and collapsing hybrid (HOPACH) (van der Laan & Pollard, 2003; Pollard & van der Laan, 2005).

- Operates on a matrix of pairwise distances between the observations to be clustered.
- A hierarchical tree of clusters is built by recursively partitioning the observations.
- At each node, a cluster is split into two or more smaller clusters using a partitioning clustering procedure such as PAM. Splits are not restricted to be binary.
- Clusters are ordered and possibly collapsed at each level of the tree.

Hybrid Methods: HOPACH

- Hierarchical. One can examine clusters at increasing levels of detail. Unlike other hierarchical clustering methods, HOPACH builds a tree of clusters in which each cluster can have more than two children.
- Ordered. At each level of the tree, the clusters and observations are ordered with a data-adaptive, deterministic algorithm, based on the same distance function that is used for clustering. The ordering of observations at any level of the tree can be used to reorder the data or distance matrices and to visualize the cluster structure. The ordering produced in the final level of the tree does not depend on the original order of the observations.

Hybrid Methods: HOPACH

- Partitioning. A cluster is split into two or more smaller clusters using a partitioning clustering procedure such as PAM.
- Collapsing. Clusters can be collapsed at any level of the tree, to join similar clusters and to correct for errors made in earlier partitioning steps.
- Hybrid. HOPACH combines the strengths of both partitioning and hierarchical clustering methods. It is a hybrid between a divisive (top-down) and an agglomerative (bottom-up) algorithm, i.e., the tree is built from the root node (all observations in one cluster) down to the leaf nodes, but collapsing steps are allowed at any level.

Hybrid Methods: HOPACH

The mean/median split silhouette (MSS) criterion, a measure of cluster heterogeneity proposed by Pollard & van der Laan (2002), is used by HOPACH to:
(i) determine the optimal number of children at each node;
(ii) decide which pairs of clusters to collapse at each level; and
(iii) identify the first level of the tree with maximally homogeneous clusters.

Hybrid Methods: HOPACH

van der Laan & Pollard (2003) propose a bootstrap procedure to assess confidence in cluster membership.

Consider clustering J features (e.g., genes = rows), based on n observational units (e.g., microarrays/target samples = columns).

The approach relies on the medoids (i.e., cluster profiles) obtained by applying HOPACH (or any partioning clustering procedure) to the entire sample of n observations.

For each bootstrap sample of the n observations, each feature is assigned to the cluster with the closest medoid.

Bootstrap cluster membership probabilities, defined as the proportions of bootstrap samples in which each feature appears in each cluster, can be viewed as measuring confidence in cluster membership, i.e., as fuzzy clustering results.

Hybrid Methods: HOPACH

N. B. 1. When clustering features (e.g., genes = rows), one resamples observations (e.g., microarrays/target samples = columns).

For each bootstrap iteration, distances between features are computed based on a subsample of the original n observations.

N. B. 2. Bootstrap cluster membership probabilities can be computed for the results of any partitioning clustering procedure.

HOPACH: R Software

The HOPACH procedure is implemented in the main function `hopach`, from the R package hopach (Pollard & van der Laan, 2005).

In this implementation of HOPACH, the partitioning steps are performed using PAM. In principle, any other partitioning procedure could be used.

The `hopach` function allows the user to specify: a distance matrix (`dmat` and `d` arguments); a cluster validity criterion (`mss`), for determining the number of children at each node, what collapsing should be performed at each level, and the main clusters; criteria for the collapsing steps (arguments `coll` for determining which pairs of clusters to consider and `newmed` for the computation of medoids for collapsed clusters); criteria for ordering clusters and observations within clusters (`initord` and `ord`).

Please consult the documentation (helpfiles, vignettes) for details.

HOPACH: R Software

The function returns a list with four elements: `clustering`, `final`, `call`, and `metric`.

The main results are stored in the list `clustering`, which consists of the following components.

- `k`: the number of clusters.
- `medoids`: a vector of indices for the k cluster medoids.
- `sizes`: a vector of cluster sizes.
- `labels`: a vector of cluster labels for each observation, which indicate the cluster membership for each level of the tree.
  E.g. 123 means that an observation belongs to the first (i.e., leftmost) cluster at level 1, to the second child of cluster 1 at level 2, and to the third child of cluster 12 at level 3.
- `order`: a vector containing the ordering of variables within the main clusters.

HOPACH: R Software

The function `dplot` produces a pseudo-color image of the distance matrix, with row and column orderings based on HOPACH clustering results.

The function `boothopach` performs non-parametric bootstrap resampling of HOPACH clustering results to assess confidence in cluster membership. The `bootmedoids` function is designed to work for any partitioning clustering procedure; the user inputs the data, the medoid row indices, and a distance function.

These bootstrap functions return a matrix of bootstrap cluster membership probabilities, with rows corresponding to the observations to be clustered and columns to the original clusters (i.e., one cluster for each supplied medoid).

HOPACH: R Software

In addition, as detailed in the vignette, the hopach package provides a number of functions for managing clustering results.

The makeoutput function writes tab delimited text files with hopach and boothopach clustering results. In the case of gene clustering, the argument gene.names can be used to insert additional gene annotation.

The functions boot2fuzzy and hopach2tree allow interfacing with MapleTree (mapletree.sourceforge.net), an open-source, cross-platform visualization tool for interactive browsing of clustering results, TreeView (rana.lbl.gov/EisenSoftware.htm), jtreeview (sourceforge.net/projects/jtreeview), and GeneXPress (genexpress.stanford.edu).

HOPACH: Case Study in R

```
> library(hopach)
> set.seed(99)
>
> X<-exprs(golub)
>
> # Correlation-based distance matrix
> d<-distancematrix(t(X), d="cor")        # sqrt of 1-cor
> dimnames(d)<-list(as.vector(Y),as.vector(Y))
>
> # HOPACH, K=1 level
> hp1 <- hopach(t(X), d="cor", K=1)
Searching for main clusters...
Level  1
Identified 5  main clusters in level 1 with MSS = 0.06764675
Running down without collapsing from Level 1
```


HOPACH: Case Study in R

```
> # HOPACH results
> class(hp1)
[1] "list"
> names(hp1)
[1] "clustering" "final"      "call"       "metric"
> names(hp1$clustering)
[1] "k"       "medoids" "sizes"   "labels"  "order"
>
> # Number of clusters
> hp1$clustering$k
[1] 5
>
> # Cluster labels
> cl1<-hp1$clustering$labels
> table(cl1, Y)
   Y
cl1 ALL B-cell ALL T-cell AML
  1 9          0          0
  2 9          0          0
  3 0          8          0
  4 1          0          6
  5 0          0          5
```


HOPACH: Case Study in R

```
> # Distance matrix plots
> pdf("hopachDplotGolub.pdf")
> dplot(d, hp1, col=topo.colors(15), showclusters=TRUE, main="Golub et al. (1999). Correlati
> axis(2, labels=rev(Y[hp1$clustering$order]), at=1:38, cex.axis=0.75, col.axis=2, las=2)
> axis(1, labels=Y[hp1$clustering$order], at=1:38, cex.axis=0.75, col.axis=2, las=2)
> dev.off()
>
> # Bootstrap cluster membership probabilities
> pdf("hopachBootGolub.pdf")
> boot1<-boothopach(t(X), hp1, I=100)
> bootplot(boot1,hp1, main="Golub et al. (1999). Barplot of bootstrap cluster membership pro
> dev.off()
```

HOPACH: Case Study in R

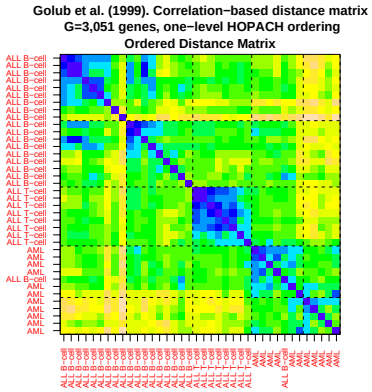

Figure 10: *Golub et al. (1999) leukemia dataset.* Correlation-based distance matrix, $G = 3,051$ genes, one-level HOPACH ordering.

HOPACH: Case Study in R

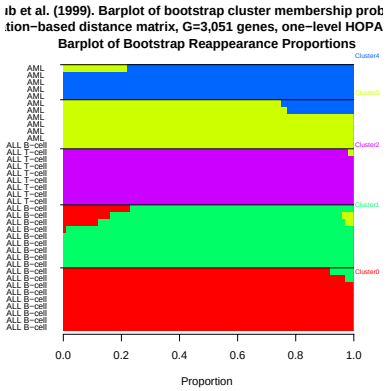

Figure 11: *Golub et al. (1999) leukemia dataset.* Barplot of bootstrap cluster membership probabilities, correlation-based distance matrix, $G = 3,051$ genes, one-level HOPACH ordering.

Estimating the Number of Clusters

- Internal indices. Statistics based on within- and between-clusters matrices of sums-of-squares and cross-products (30 methods reviewed in Milligan & Cooper (1985)). Estimate is the number of clusters K which minimizes or maximizes one of these indices.
- Average silhouette width. (Kaufman & Rousseeuw, 1990).
- Model-based methods. EM algorithm for Gaussian mixtures (Fraley & Raftery, 1998, 2000; McLachlan et al., 2001).
- Gap statistics. For each K, compare an observed internal index to its expected value under a resampling-based reference distribution and look for K which maximizes the difference (Tibshirani et al., 2001).

Estimating the Number of Clusters: MSS

Mean split silhouette (MSS) (Pollard & van der Laan, 2002).

Given K clusters, consider each cluster k, $k = 1, \ldots, K$, separately.

- Apply the clustering algorithm to the elements of cluster k.
- Choose the number of child clusters that maximizes the average silhouette width. Define this maximum as the split silhouette, SS_k.

Define the mean split silhouette, a measure of average cluster heterogeneity, by

$$MSS(K) \equiv \frac{1}{K} \sum_{k=1}^{K} SS_k.$$

Choose the number of clusters K which minimizes $MSS(K)$. One can also use the median split silhouette robust version.

Estimating the Number of Clusters: MSS

- Identifies finer structure in gene expression data. When clustering genes, existing criteria tend to identify global structure only.
- Provides a measure of cluster heterogeneity.
- Computationally easy.

Estimating the Number of Clusters: Clest

Clest (Dudoit & Fridlyand, 2002). Resampling method which estimates the number of clusters based on prediction accuracy.

- For each number of clusters K, repeatedly randomly divide the original dataset into two non-overlapping sets, a training set and a validation set. For the bth split, $b = 1, \ldots, B$,
  - apply the clustering algorithm to observations in the training set;
  - build a classifier using the class labels from the training set clustering;
  - apply the classifier to the validation set;
  - compute a similarity score $s(K, b)$, comparing the validation set class labels from prediction and clustering.

Estimating the Number of Clusters: Clest

- The similarity score for K clusters is the median of the B similarity scores: $s(K) = \text{median}(s(K, 1), \cdots, s(K, B))$.
- The number of clusters is estimated by comparing, for each K, the observed similarity score $s(K)$ to its expected value under a suitable reference distribution with $K = 1$ cluster.

Applies to any partitioning algorithm and any classifier.

Better suited for clustering microarrays/target samples than clustering genes.

Inference

van der Laan & Bryan (2001). General framework for statistical inference in cluster analysis.

View clustering as a deterministic rule that can be applied to parameters (or estimates thereof) of any distribution.

Parameters of interest include covariance matrices, e.g., covariance matrix between the expression measures of different genes.

The bootstrap can be used to study distributional properties (bias, variance) of the clustering results.

Bagged Clustering

Leisch (1999). Hybrid method combining partitioning and hierarchical procedures. A partitioning method is applied to bootstrap samples and the resulting partitions are combined by performing hierarchical clustering of the cluster centers.

Dudoit & Fridlyand (2003). Apply a partitioning clustering method to bootstrap samples of the learning set. Combine the resulting partitions by (i) voting or (ii) the creation of a new distance matrix. Assess confidence in the clustering results using cluster votes.