A new algorithm for hybrid clustering of gene expression data with visualization and the bootstrap

Mark J. van der Laan and Katherine S. Pollard

Revised for Journal of Statistical Planning and Inference, 2003, 117, p. 275–303

Copy with figures

http://www.stat.berkeley.edu/~laan/Research/Research_subpages/Papers/hopach.pdf

Abstract

An important goal with large-scale gene expression studies is to find biologically important subsets and clusters of genes. In this paper, we propose a hybrid clustering method, Hierarchical Ordered Partitioning And Collapsing Hybrid (HOPACH), which is a hierarchical tree of clusters. The methodology combines the strengths of both partitioning (or divisive) and agglomerative clustering methods. At each node, a cluster is split into two or more smaller clusters with an enforced ordering of the clusters. Collapsing steps uniting the two closest clusters into one cluster can be used to correct for errors made in the partitioning steps. We implement an automated HOPACH which increases average silhouette at each divisive and collapsing step until no more improvement is possible. We

[0]Mark J. van der Laan is Professor in Biostatistics and Statistics, University of California, Berkeley. Katherine S. Pollard is a doctoral candidate in Biostatistics in the School of Public Health, Division of Biostatistics, University of California, Berkeley. This research has been supported by a grant from the Life Sciences Informatics Program with industrial partner biotech company Chiron Corporation. Author for correspondence: Mark J. van der Laan, Div. of Biostatistics, Univ. of California, School of Public Health, Earl Warren Hall #7360, Berkeley, CA 94720-7360

propose to visualize the clusters at any level of the tree by plotting the distance matrix corresponding with an ordering of the clusters and an ordering of genes within the clusters. An important benefit of a hierarchical tree is that one can look at clusters at increasing levels of detail. A final ordered list of genes is obtained by running down the tree completely, possibly intervening with collapsing steps. The bootstrap can be used to establish the reproducibility of the clusters and the overall variability of the followed procedure. The power of the methodology compared to current algorithms is illustrated with simulated and publicly available cancer data sets.

Key words: Gene expression, bootstrap, cluster analysis, hierarchical tree.
Abbreviated title: HOPACH clustering of gene expression data
AMS classification: primary 62H30, secondary 62H10

1 Introduction

New technologies are allowing researchers to monitor the expression of thousands of genes simultaneously. By comparing gene expression profiles across cells that are at different stages in some process, in distinct pathological states, or under different experimental conditions, we gain insight into the roles and interactions of various genes. For example, one can compare healthy cells to cancerous cells within subjects in order to learn which genes tend to be over (or under) expressed in the diseased cells; regulation of such differentially expressed genes could produce effective cancer treatment or prophylaxis (DeRisi *et al.* (1996)). Groups of differentially expressed genes which are *significantly correlated with each other* are particularly interesting, since such genes might be part of the same causal mechanism. In addition to identifying interesting clusters of genes, researchers often want to find subgroups of samples which share a common gene expression profile. Ross *et al.*, for example, use microarray data to classify sixty human cancer cell lines.

A typical gene expression experiment results in an observed data matrix X whose columns are n copies of a p-dimensional vector of gene expression measurements. Con-

sider, for example, a population of cancer patients from which we take a random sample of n patients, each of whom contributes p gene expression measurements. For microarrays, each measurement is a ratio, calculated from the intensities of two flourescently labeled mRNA samples (*e.g.*: tumor and healthy tissues) cohybridized to arrays spotted with known cDNA sequences. Gene chips produce similar data, except each element is a quantitative expression level rather than a ratio. The methods presented can be applied to both types of data, but we will focus on microarrays. Data preprossessing may include background subtraction, combining data from replicated spots representing the same cDNA sequence, normalization, log transformation, and truncation.

Data analysis methods appropriate for microarray data are presented and surveyed by Claverie (1999), Tibshirani *et al.* (1999), Eisen *et al.* (1998), and Herwig *et al.* (1999). Approaches to gene expression data analysis rely heavily on results from cluster analysis (*e.g.*: agglomerative hierarchical clustering, k-means, self-organizing maps). Most standard clustering routines, including the methods proposed in this paper, fit in the general statistical framework for gene clustering presented in van der Laan & Bryan (2001). In this formal framework, the underlying target subset (with cluster labels) is defined as a deterministic function $S(\mu, \Sigma)$ of the population mean and covariance matrix (or any other smooth function of the true data generating distribution). The bootstrap is used to estimate the distribution of the estimated target subset $S(\mu_n, \Sigma_n)$, obtained by substituting the empirical mean and covariance. In particular, the authors propose to run the bootstrap with fixed cluster centers and to report for each gene the cluster-specific proportion of times it falls in that cluster. A corresponding cluster probability plot allows one to inspect the cluster reproducibility visually and numerically. The authors also prove consistency of the subset estimates and asymptotic validity of the bootstrap under the assumption that the sample size converges faster to infinity than the logarithm of the number of genes.

Hierarchical clustering of genes has been popularized by Eisen *et al.*, who apply an agglomerative hierarchical clustering algorithm to the empirical correlation matrix. Kaufman & Rousseeuw describe other hierarchical clustering algorithms and implement several of these in FORTRAN and Splus functions, which can be used to cluster gene

expression data. In addition to the resulting hierarchical tree, these routines all produce a (non-unique) corresponding ordered list of the genes. Such a list, in which nearby genes are supposed to be similarly expressed, is more helpful to the subject-matter scientist than a collection of large, unordered groups of genes. Some drawbacks of these methods are that the ordering of the genes has a random component and that the hierarchical tree forces binary splits at each node, while there is no biological reason for doing so. In addition, the variability and reproducibility of the clustering results need to be addressed by the bootstrap. The bootstrap results could also be used to obtain a sensible ordering of genes within clusters so that weakly clustered genes can be filtered out.

In addition to illustrating how hierarchical clustering can be applied to gene expression data, Eisen *et al.* provide a visual plot of the ordered p by n data matrix X. They apply their **Cluster** algorithm separately to both genes and samples to reorder the rows and columns of the data matrix for the plot. Each log ratio is represented by a spot on the red/green color scale. This plot allows one, to a certain extent, to visually inspect the strength of clusters. We feel that the visualization of clusters is an important contribution, but that plotting the data matrix might not show clustering patterns with respect to distances such as absolute correlation. We propose that one should, in particular, visualize the corresponding distance matrix.

In this paper, we aim to build on the strengths of these currently employed methods and to address some of their drawbacks. In section 2, we describe our clustering method, Hierarchical Ordered Partitioning and Collapsing Hybrid (HOPACH), which creates a hierarchical tree of clusters with an enforced ordering of the clusters. Important components of this method are 1) running down the tree to obtain an ordered list of genes and plotting the corresponding ordered distance matrix to visually and judge the clustering output, 2) identifying the main clusters using an automated HOPACH, and 3) given a particular level of the tree, running the bootstrap to establish the reproducibility of the clusters and to provide a corresponding ordering of genes within the clusters with respect to the cluster-specific bootstrap probabilities. In section 3, we apply the method to a simulated data set in order to illustrate its performance. In

section 4, we analyze a publicly available cancer data set consisting of a variety of cell lines corresponding with different types of tumors.

2 Hierarchical Ordered Partitioning and Collapsing Hybrid clustering and the bootstrap.

We present a new clustering method, Hierarchical Ordered Partitioning and Collapsing Hybrid (HOPACH), which applies a partitioning algorithm iteratively to create a hierarchical tree whose final level is an ordered list of the elements. The ordering of elements at any level of the tree can be used to visualize the clustering structure in a colored plot of the reordered data or distance matrix. Ordered distance matrix plots, and to a certain degree ordered data matrix plots, help to determine the main clustering structures in the data set and provide information about the clusters such as their strength and their similarity to each other. A collapsing step can be applied at any level of the tree to unite similar clusters. By combining the strengths of two celebrated approaches to clustering, partitioning and agglomerative methods, we create a more flexible algorithm for finding patterns in data.

It is important to note that our methodology is a general approach which could be applied with any choice of partitioning algorithm. Commonly employed partitioning algorithms include k-means and Self-Organizing Maps (SOMs). To be concrete, however, we will henceforth assume that Partitioning Around Medoids (PAM) is being used. We refer to our particular implementation with PAM as the partitioning algorithm as HOPACH-PAM. Other possible algorithms would be HOPACH-KMEANS and HOPACH-SOM.

2.1 Partitioning Around Medoids (PAM).

Our HOPACH-PAM algorithm is based on the output of the clustering procedure PAM (Kaufman & Rousseeuw, 1990, chap. 2), which takes as input a dissimilarity matrix $\mathbf{D}$ based on any distance metric. Let D_{ij} denote the dissimilarity between genes i and j

where each gene is represented by an n dimensional vector. Possible dissimilarities are:

$$\begin{aligned} D_{ij} &= 1 - \rho_{ij} \text{ correlation} \\ D_{ij} &= 1 - \mid \rho_{ij} \mid \text{ absolute correlation} \\ D_{ij} &= 1 - \rho^0_{ij} \text{ cosine-angle} \\ D_{ij} &= 1 - \mid \rho^0_{ij} \mid \text{ absolute-cosine-angle} \\ D_{ij} &= \sum_{l=1}^{n} (Y_{il} - Y_{jl})^2 \text{ euclidean,} \end{aligned}$$

where

$$\rho^0_{ij} \equiv \frac{\sum_{l=1}^{n} Y_{il} Y_{jl}}{\sqrt{\sum_{l=1}^{n} Y_{il}^2} \sqrt{\sum_{l=1}^{n} Y_{jl}^2}}.$$

It is of interest to note that the $1 - \rho^0_{ij}$ equals 0.5 times the squared euclidean distance of the two vectors standardized to have euclidean norm 1. This distance was used in Eisen *et al.* (1998), and it has been our experience that it is a sensible choice in many applications.

Let K be the number of clusters (*i.e.*: the number of causal mechanisms we believe to be operating). Given K, PAM selects K potential medoids, calculates for each gene its distance to the closest of these potential medoids and minimizes over the vector of K potential medoids the sum of these distances over all genes. The solution of this minimization problem is a vector of K medoids. Each medoid identifies a cluster, defined as the genes which are closer to this medoid than to any of the other $K - 1$ medoids. One can consider K as given or it can be data-adaptively selected, for example, by maximizing the average silhouette as recommended by Kaufman & Rousseeuw. The silhouette for a gene is calculated as follows. For each gene j, calculate a_j which is the average dissimilarity of gene j with each other member of gene j's cluster. For each gene j and each cluster k that is not gene j's cluster, calculate b_{jk}, which is defined as the average dissimilarity of gene j with the members of cluster k. Let $b_j \equiv \min_k b_{jk}$, where the minimum is taken over all clusters k that are not gene j's cluster. Finally, the silhouette of gene j is defined by the formula:

$$silhouette_j = \frac{b_j - a_j}{\max(a_j, b_j)}.$$

Note that the largest this can be is 1, which occurs only if there is no dissimilarity within gene j's cluster (*i.e.*: $a_j = 0$). The other extreme is -1. Heuristically, the silhouette measures how well matched an object is to the other objects in its own cluster versus how well matched it would be if it were moved to another cluster.

The (minimal) output of PAM consists of two vectors: (1) a p-dimensional vector $\mathbf{c}$, where $c_j = k$ indicates that gene j belongs to cluster k, and (2) a K-dimensional vector $\mathbf{m}$, where $m_k = j$ indicates that the medoid of cluster k is gene j, where $j \in \{1, \ldots, p\}$ and $k \in \{1, \ldots, K\}$. An attractive property of PAM is that the clusters are identified by the medoids, which are genes themselves, and it has been our experience that the medoids are stable representations of the clusters.

2.2 HOPACH-PAM.

Firstly, we describe an interactive algorithmic explanation of the proposed HOPACH-PAM algorithm for building a hierarchical tree of clusters at any level of detail (with the extreme being an ordered list of genes), as we have implemented it in an Splus program. Subsequently, we describe an automated version which finds the main clusters by optimizing average silhouette over the possible partitioning and collapsing steps. Let the distance metric d and integers "klow", "khigh" be given. We will use the notation $PAM(data, k, d)$ for PAM applied to the data "data" with k being the number of clusters and d the distance.

Subsetting/Pre-screening A typical first step is to select a subset of all p genes. For example, this can be based on cut-off values for mean expression (*e.g.*: select all genes which are on average 2-fold differentially expressed).

Initial level of tree Let subdata be the $r \times n$ data-frame of the r remaining genes. We apply now $PAM(subdata, k1, d)$ with a user-supplied number of clusters $k1$ (e.g. selected by maximizing average silhouette) to the subdata and we decide on an ordering of the $k1$-clusters. If $k1 = 2$, then the ordering does not matter. It is not necessary to use PAM here to decide on an initial split representing the main clusters. For example, we also allow to set cluster1 equal to all genes which are

suppressed on average and cluster2 equal to all genes which are over-expressed on average, or one can use the the automated HOPACH function rundownconverge() proposed below to determine the initial level of clusters. If $k1 > 2$, then we have two proposals for ordering the $k1$ clusters. Firstly, one needs to define a distance between any pair of clusters. Here we define the distance between a pair of clusters as the distance between the corresponding medoids, but other distances are options. The first method consists of building a hierarchical tree from the medoids with PAM as follows. Let "medoids.data" be the $k1$ by n matrix containing the medoids. Initially, we apply $PAM(medoids.data, 2, d)$ and label the two clusters with clust1 and clust2. For each of the two clusters we can now define the neighboring cluster "clust-next". Subsequently, at each node we apply PAM again with say $k = 2$ and we now order the k new clusters by their distance with respect to medoid of "clust-next" going from maximal distance to smallest distance if "clust-next" is to the right and from smallest distance to maximal distance if "clust-next" is to the left. In this way each level of the tree has an ordered list or clusters. By running down the tree until each cluster is of size one, we obtain a unique ordering of the $k1$ medoids.

We also implemented a function correlationordering() which maps an initial ordering of the $k1$ medoids and corresponding $k1 \times k1$ distance matrix into an ordered list of medoids which maximizes the empirical correlation between distance $j - i$ in the list and the corresponding distance $d(i, j)$ across all pairs (i, j) with $i < j$. The second method for ordering the clusters is to apply this function to any initial ordering of medoids or the one provided by the Hierarchical-PAM procedure described above. In our data examples this function did not change the ordering obtained by the method described above.

One now computes a list "hclust1" containing as elements subdata, an r-dimensional vector "labels" (assigning a cluster label to each gene), the total number of clusters "k", the row-numbers of the k medoids of the clusters "medoids", and the "cluster-sizes". One sets "hclust.previous" equal to "hclust1".

Step I: Next level of tree For each cluster "clust" in the previous level "hclust.previous" of the tree, one carries out the following procedure. Let "clust-next" be the next cluster in the previous level of the tree and if the current cluster is already the last cluster in the previous level, then we set "clust-next" equal to the previous cluster. Firstly, one applies $PAM(clust, k, d)$ with k running between "klow" and "khigh" and one selects the $k = k^*$ maximizing the average silhouette. Subsequently, one orders the k^*-clusters by their distance with respect to the medoid of "clust-next" going from maximal distance to smallest distance if "clust-next" is to the right and from smallest distance to maximal distance if "clust-next" is to the left. One can allow $k^* = 1$, i.e. stop splitting a cluster, by using some criteria. In particular, we allow not splitting a cluster if the "cluster-size" is below a cut-off value or if it results in a silhouette below a cut-off value. Again, the ordering of the clusters can be based on other pairwise distances between clusters than distances between medoids.

One now computes a list hclust.next containing as elements 1) "subdata", 2) an r-dimensional vector "labels" which extends the previous labels with another digit being the new cluster label, 3) the total number of clusters "k" at this level of the tree, 4) the row numbers "medoids" in "subdata" of the corresponding medoids and 5) the "cluster-sizes". Thus, a typical label of a gene at level 4 in the tree looks like "01.03.02.04" describing the path it walked through the tree. If a cluster is not split (*i.e.* $k^* = 1$), then the label of each of the genes in that cluster is extended with the digit "00" so such a label will look like "01.03.02.04.00".

Step II: Visualization of an ordered distance matrix. We can visualize the ordered distance matrix "distance"(l) corresponding with the l-th level of the tree. In this case, groups (clusters) of genes have the same label and will thus need to be ordered or left as in the original data matrix (*i.e.* randomly). We choose to order the genes within each of the clusters of this level l of the tree by either (i) their distance with respect to the medoid of that cluster so that the badly clustered genes end up at the edge of these clusters or (ii) their distance with

respect to the medoid of the neighboring cluster.

Step III: Possibly collapse similar clusters Due to the nature of hierarchical trees, it can happen that two or more clusters not next to each other are very similar. Visual inspection of the distance matrix "distance"(l) will often make it possible to identify medoids of different clusters, say i, j with $i < j$, which are very similar so that one might want to collapse the two corresponding clusters into one cluster i, with a medoid being (for example) the nearest neighbor of the average of the two corresponding medoids. Rather than identifying clusters to collapse based on distance between their medoids, one could also use other distances between clusters. The labels of one cluster are changed to those of the other cluster, so that the tree structure is preserved. The choice of labels could be based on similarity of the old medoids to the new neighboring cluster or some other criteria. After collapsing, one can visualize the distance matrix of the new ordered set of clusters as in Step II and decide if more collapsing steps are needed.

In addition to visual inspection, we also allow the decision to collapse to be based on comparing the average silhouette before and after the collapsing step. In particular, we implement a function which collapses until there is no pair of clusters for which a collapse improves average silhouette.

Iterate One can iterate this process by setting "hclust.previous" equal to "hclust.next" and carry out Step I "Next level of tree", Step II "Visualization", and Step III "Collapsing" again.

2.3 An automated version selecting the main clusters.

We also implement a completely automated HOPACH-PAM which has as its only goal to find the main clusters. This is done by 1) creating an initial level of the tree as above, 2) collapsing pairs of clusters until there is no pair of clusters for which a collapse improves average silhouette and we refer to this as the initial level of the tree, 3) creating a next level of the tree as above, 4) collapsing pairs of clusters in this next level of the tree until there is no pair of clusters for which a collapse improves

average silhouette, and then 5) accepting this collapsed next level of the tree if its average silhouette improves on the average silhouette of the previous (i.e. initial) level of the tree and accepting the previous level as final clustering output otherwise. If the collapsed next level is accepted, then one sets the current level equal to this collapsed next level and one repeats steps 2, 3 and 4 until one stops.

2.4 Final ordered list of genes and selecting the main clusters.

If we iterate the hierarchical PAM tree and run it down completely until every gene in "subdata" has a unique label, then we can order the genes from smallest to largest label which corresponds with the ordering of the clusters in the final level of the tree. It has been our experience that the collapsing step is most important at the initial levels of the tree, so that one can run down the tree automatically without further visual inspections after several initial iterations. Let "subdata.ord" be the $r \times n$-data matrix obtained by ordering "subdata". As in Eisen *et al.* (1998), we can visualize this data matrix "subdata.ord" by mapping the gene expressions into a color scheme. In addition, we can now compute the corresponding ordered $r \times r$-distance matrix "distance.final".

The ordered list of genes has two important applications. The list itself is useful since genes close in the list will be close in distance to each other. Given any gene of biological interest, we can find similarly expressed genes nearby in the list. Visualization of the distance matrix corresponding with the final ordering of all genes is also of interest, since it shows clusters at all levels of detail. Inspection of the distance matrices "distance"(l) at various levels l will often make it possible to select a level l^* of the tree which already closely resembles the important clustering structures in the final distance matrix "distance.final".

One form of output is now the ordered list of genes as in "subdata.ord" with an additional column giving their main cluster label. Additional output of interest is obtained by ordering the genes within each of the clusters of this level l^* of the tree by their distance with respect to the medoids so that the badly clustered genes end up at the edge of these clusters. It can also be of interest to add to each gene the vector

of relative distances (*i.e.* the sum of these add up to one) with respect to each of the medoids at level l^* of the tree so that the user can directly see how well a gene belongs to one or more of the main clusters. The latter is known as "fuzzy clustering".

2.5 The bootstrap.

Let μ_n, Σ_n be the $p \times 1$ empirical mean and $p \times p$ covariance matrix. Since (for the previously mentioned dissimilarities) the dissimilarity matrix is a function of μ_n, Σ_n, we can view each particular HOPACH-PAM algorithm, as defined by the various choices such as the level l^* of the tree, as a particular functional $(\mu, \Sigma) \rightarrow S(\mu, \Sigma)$ applied to (μ_n, Σ_n). Therefore we can consider the HOPACH-PAM clustering result $S(\mu_n, \Sigma_n)$ as an estimator of a true clustering parameter $S(\mu, \Sigma)$. In order to establish the variability and reproducibility of the clustering output $S(\mu_n, \Sigma_n)$ (*e.g.* the clusters in level l^* of the tree), we propose to run the parametric or nonparametric bootstrap. This involves repeatedly sampling n observations $Y_1^\#, \ldots, Y_n^\#$ from a multivariate normal distribution $N(\mu_n, \Sigma_n)$ (van der Laan & Bryan (2001)) or from the empirical distribution which puts mass $1/n$ on each of the original observations $Y_1, \ldots, Y_n$. One estimates the distribution (and, in particular, the variance) of the clustering output $S(\mu_n, \Sigma_n)$, with the empirical distribution of $S(\mu_n^\#, \Sigma_n^\#)$. One useful application of the bootstrap is to estimate the variability of the clusters at different levels of the tree and choose as main clusters the lowest level which is reproducible at a certain level of stringency. We will now describe some approaches to the bootstrap for estimating components of the distribution of $S(\mu_n, \Sigma_n)$.

Bootstrap for a deterministic algorithm with fixed number of clusters at each node. Useful summary measures of the bootstrap can be obtained by treating $S(\mu_n, \Sigma_n)$ as a deterministic function of (μ_n, Σ_n) for which the number of clusters at each each node is fixed, even when the number of clusters were in fact determined by maximizing average silhouette at each partitioning step. This guarantees that the number of clusters of $S(\mu_n^\#, \Sigma_n^\#)$ is fixed. In this case, the variability in number of clusters at each node is not addressed by the bootstrap. If the actual clustering was based on an interactive process with the user, then the steps defining $S(\mu_n, \Sigma_n)$ need to

be treated as fixed in the bootstrap. This approach views the data adaptive selection of the number of clusters or the interactive process used as a way to decide on a particular clustering parameter $S(\mu, \Sigma)$ and we use the bootstrap to estimate the variability of this particular parameter (but not variability in choosing the parameter). For example, the user might have split the original data into 2 clusters in the first level of the tree and then split these into 2 and 4 clusters, respectively, before collapsing the closest two clusters in this second level of the tree. Even if these splits and collapses were made data-adaptively, they are treated as fixed and applied exactly to each bootstrap sample. In this case, the bootstrapped $S(\mu_n^\#, \Sigma_n^\#)$ has the same number of clusters as $S(\mu_n, \Sigma_n)$.

In order to infer a correspondence between each of the bootstrap clusters with one of the original clusters in the original data, we propose to align the clusters in each split of the tree by examining the matrix of pair-wise distances between all bootstrap and original clusters in that split and consecutively matching the closest pairs. The distance between clusters could be based on the distance between medoids or a measure of the overlap in membership, such as $(A \cap B)/(A \cup B)$ or $P(A|B)/2 + P(B|A)/2$. Since there is now a correspondence between each cluster of $S(\mu_n, \Sigma_n)$ with a cluster in the bootstrapped $S(\mu_n^\#, \Sigma_n^\#)$, for each gene one can keep track of the proportion of times among the bootstrap samples that gene fell into each of the clusters $S(\mu_n, \Sigma_n)$. van der Laan & Bryan propose a cluster-probability plot to summarize these statistics which provides a visual way to inspect the cluster reproducibility, where we order the clusters as in $S(\mu_n, \Sigma_n)$. In addition, van der Laan & Bryan propose a sensitivity and positive predictive value measure for each cluster measuring proportions of correct genes and proportions of false positives.

Bootstrap for fixed medoids. A particular clustering output is to apply PAM with fixed medoids being the medoids of the clusters in $S(\mu_n, \Sigma_n)$. The clusters are now defined by the medoids and, in our data examples, they closely resembled the clusters in $S(\mu_n, \Sigma_n)$. In order to establish the cluster variability when fixing the medoids, one fixes the medoids in the bootstrap. Note that this bootstrap avoids estimating the variability in the selection of the medoids. Nonetheless, it is a sensible approach

to understanding the variability of the specific clusters obtained in the data analysis. Since the medoids are the same in each bootstrap sample, as in van der Laan & Bryan (2001), for each gene one can keep track of the proportion of times among the bootstrap samples that gene fell into each of the clusters and summarized these statistics in a cluster-probability plot. These bootstrap cluster-specific probabilities can also be used to order the genes within the clusters so that the badly clustered genes can be removed or end up at the edge of the clusters. It is particularly interesting to carry out this fixed medoids bootstrap to estimate the variability of the clustering output $S_{l^*}(\mu_n, \Sigma_n)$ at a level l^* of the tree for $l^* == 1, 2, 3, \ldots$. In this manner one can use the bootstrap to establish the levels of detail which cannot be distinguished anymore from the noise level.

Bootstrap for the automated HOPACH-PAM. One can also use the bootstrap to establish the variability and reproducibility of the main clustering results found by the automated HOPACH-PAM. Since the number of clusters is now also variable it does not necessarily make sense to align the bootstrap clusters with the original clusters and define cluster-specific variability measures, but one can still estimate the variability of the number of clusters and measures such as average silhouette. It helps to at least enforce an ordering of the bootstrapped clusters corresponding as close as possible to the ordering in $S(\mu_n, \Sigma_n)$ by comparing their medoids. In that case, visualizing a number of bootstrapped $S(\mu_n^{\#}, \Sigma_n^{\#})$ provides a very good sense of the variability of $S(\mu_n, \Sigma_n)$. One can also treat the tree produced by the automated HOPACH-PAM as fixed and perform the bootstrap on this deterministic proceedure as described above. In this case, the variability in number of clusters is no longer addressed by the bootstrap.

2.6 Clustering samples.

Researchers are not only interested in clustering genes, but also in clustering or classify samples based on similarity of gene expression patterns. Thus, while gene clustering results are of biological interest themselves, they serve an additional purpose if we can use them to aid in the task of clustering samples. We have found that different subsets of genes often cluster samples in different ways. Consider, for example, that different

gene clusters represent different biological mechanisms or states, so that there may be clusters of genes which are very good for distinguishing different types of samples. In addition to an overall clustering label, sub-groups of samples could be more accurately characterized by their expression pattern for each of these gene clusters. The simulation and data analysis illustrate, in particular, that a sensible strategy for clustering the samples is to first cluster genes and then cluster samples for each cluster of genes separately. These results can again be visualized in a reordered data matrix, where the ordering of samples is produced by applying HOPACH-PAM to the transposed data matrix. This approach can reveal underlying structure in the data which may not be apparent when the samples are clustered using all genes.

For a more formal treatment of this subject we refer the reader to Pollard & van der Laan (2001), where we extend the statistical framework of van der Laan & Bryan (2001) to include clustering of both genes and samples by defining a simultaneous clustering parameter which is a composition of a mapping for clustering genes and a mapping for clustering samples.

3 A simulated example.

In order to investigate the clustering performance of HOPACH-PAM, we have conducted simulations comparing the automated algorithm to the existing clustering routines PAM and KMEANS. We chose to use the Euclidean distance so that KMEANS (which allows only this distance metric in its usual implementation) could be compared to the other algorithms in the context where it performs best. We fixed the number of clusters to be the correct number in PAM and KMEANS.

3.1 Data generation

Consider a sample of $n = 60$ relative gene expression profiles of dimension $p = 500$, corresponding to cancer patients. Suppose that in truth there are three groups of 20 patients corresponding with three distinct types of cancer, but this is unknown to the data analyst. To generate such data, we sampled three groups of 20 subjects from three

multivariate normal distributions with diagonal covariance matrices, which differed only in their mean vector. All genes had common standard deviation log(1.6), which corresponds with a 0.75-quantile of all standard deviations in an actual data set. For the first subpopulation, the first 25 genes had $\mu_j = \log(3)$, genes 25-50 had $\mu_j = -\log(3)$, and the other 350 genes had mean zero. Then for the second subpopulation, genes 51-75 had $\mu_j = \log(3)$, genes 76-100 had $\mu_j = -\log(3)$ and the other 350 genes had mean zero. For the third subpopulation, genes 101-125 had $\mu_j = \log(3)$, genes 126-150 had $\mu_j = -\log(3)$ and the other 350 genes had mean zero. In other words, the cause of each of the three types of cancer is related to 50 genes of which 25 are suppressed (tumor-suppressor genes) and 25 are over-expressed (the onco-genes). All logs in the simulation are base 10.

We generated $B = 100$ such data sets. Note that this data generating distribution is such that a clustering routine is needed to identify the underlying structure. For example, when we simply ordered the genes by mean expression and made a picture of the corresponding Euclidean distance matrix, we saw no obvious pattern within the over-expressed genes and suppressed genes. We applied PAM, KMEANS and automated HOPACH-PAM with "klow"=2, "khigh"=9 to each of the $B = 100$ data sets. We used the Euclidean distance metric in all three algorithms and $K = 7$ clusters in PAM and KMEANS.

3.2 Identifying the clustering parameter

Given this data generating distribution, each of the algorithms defines a clustering parameter. These parameters were identified by running the algorithms on the true Euclidean distance matrix, which is a function of the mean and covariance of the data generating distribution. In this case, all three algorithms have the same clustering parameter which is six clusters of 25 genes and one large cluster of 350 non-differentially expressed genes. The true cluster labels correspond with a true average silhouette of 0.27. Since the algorithms are estimating the same clustering parameter, the distance of average silhouette from this shared true average silhouette can be used as a measure of clustering performance. For example, if average silhouette for an algorithm is far from

the true average silhouette then the cluster labels can not be correct. The converse is not necessarily true, so that when average silhouette is close to the true average silhouette, it is useful to also visualize the clustering result in order determine how close it is to the true seven clusters.

3.3 Average silhouette.

Table 1 shows the mean and standard error of average silhouette across the $B = 100$ data sets for each algorithm. According to this criteria, HOPACH-PAM performed better than the existing algorithms since it was less variable and produced higher average silhouettes. Both KMEANS and PAM were quite variable, sometimes producing average silhouettes as high as HOPACH-PAM, but also many average silhouettes below 0.1. On average, KMEANS performed better than PAM in terms of maximizing average silhouette, but it was almost twice as variable.

Algorithm	Mean (SE) Average Silhouette
KMEANS	0.16 (0.080)
PAM	0.094 (0.045)
HOPACH-PAM	0.24 (0.0059)
TRUE VALUE	0.27

Table 1: Average silhouettes from Euclidean distance simulation with $K = 7$ clusters in PAM and KMEANS. The mean and standard error across the $B = 100$ simulated data sets is reported for each clustering algorithm.

3.4 Visualization.

In order to investigate how well each algorithm is able to identify the seven clusters and the degree to which average silhouette measures this success, we plotted the reordered distance matrix for each data set according to the cluster labels from each algorithm (so that genes clustered together would appear consecutively, but the clusters were not ordered). The performance of the algorithms varied greatly, although all of the

algorithms identified the clusters perfectly at least once. KMEANS often split one or more of the six small clusters or combined (parts of) two of these together, sometimes with mean zero genes also. This result may be a consequence of the lack of robustness of cluster means. These erroneous clustering results for KMEANS had relatively high average silhouettes (> 0.15) because the errors only affect the average silhouettes of a small group of genes. PAM did not tend to split or combine the six small clusters, but often split the mean zero genes into two clusters, combining about half of them with one of the six small clusters. This makes sense since PAM minimizes the sum of the distances to the closest medoid and splitting the mean zero genes reduces the distance to the closest medoid for many genes. The erroneous clustering results of PAM always resulted in an average silhouette less than 0.10, since splitting of the ≈ 350 mean zero genes creates two clusters whose neighbors are very close, making the silhouettes of the genes in these two clusters very small. HOPACH-PAM selects the number of clusters in addition to identifying the clusters. The automated algorithm usually chose $k = 7$ (see below), and when it did so, the clusters were the correct ones. These results had high average silhouettes. Occasionally HOPACH-PAM chose $k = 6$ or 9 clusters. These results corresponded with combining a small cluster with the mean zero genes and dividing a small cluster into many clusters, for $k = 6$ and 9 respectively.

3.5 Choosing the number of clusters.

The number of clusters k can be given or it can be data-adaptively selected, for example, by maximizing the average silhouette as recommended by Kaufman & Rousseeuw. For each data set (with 500 genes and 60 samples), we applied PAM and KMEANS with Euclidean distance and $k = 2, \ldots, 10$ clusters. We observed that KMEANS produced clustering results with maximum average silhouette frequently at $k = 6$ and PAM produced clustering results with average silhouettes increasing in k up to $k = 8$ or 9, so that choosing the number of clusters by maximizing average silhouette, as suggested by Kaufman & Rousseeuw, would usually not result in the correct number of clusters ($k = 7$) with KMEANS or PAM. The automated HOPACH-PAM simultaneously chooses the number of clusters (based on average silhouette) and performs the

clustering. In contrast to KMEANS and PAM, HOPACH-PAM frequently identified the correct number of clusters, choosing $k = 7$ in 83%, $k = 6$ in 14%, and $k = 9$ in 3% of the simulated data sets.

3.6 Variations in the simulation.

Number of genes. We repeated the simulations with many more genes, but cluster sizes of the same proportions. The trends observed were the same as those reported for 500 genes.

Number of samples. Since the clustering algorithms estimate the same clustering parameter, their differing performance was the consequence of their having different efficiencies for a fixed, relatively small number of samples ($n = 60$). In order to investigate the asymptotic consistency, we repeated the simulation for $n = 1200$ samples. We generated $B = 100$ data sets, and fixed $k = 7$ in KMEANS and PAM. We observed that KMEANS was very sensitive to its starting values, only some times finding the true clusters. In contrast, PAM always found the true clusters. Next, we looked at the KMEANS and PAM average silhouette for $k = 2, \ldots, 10$. PAM always had a maximum average silhouette at $k = 7$, but KMEANS had a maximum average silhouette at either $k = 6$ or 7. Thus, PAM was better able to identify the correct number of clusters and to correctly assign genes to these clusters. HOPACH-PAM always identified the true seven clusters. Hence, we see that for n large enough the outputs of PAM and HOPACH-PAM consistently estimate the same clustering parameter. For a fixed size data set with $n << p$, however, HOPACH-PAM is more efficient at estimating this parameter.

Binary splits. We also ran the simulations with the automated HOPACH-PAM allowing only binary splits at each node. In this case, the algorithm only identified the true seven clusters about half of the time, frequently finding six or fewer clusters. This result illustrates the importance of allowing a data-adaptive number of clusters in each split of a hierarchical tree.

Using visualization. Rather than applying the automated HOPACH-PAM, it is also possible to cluster the simulated data sets manually by visualizing the ordered distance

matrix at each level of the tree. When this method is used, it is clear which clusters need to be collapsed since the ordered distance matrix contains off-diagonal blocks of solid color corresponding with the correlated genes which have been put in different clusters in the first level of the tree. By collapsing such pairs of clusters, the true seven clusters are easily identified in the second level of the tree.

Distance metric. We repeated the simulations with the cosine-angle distance metric, which is commonly employed in gene expression clustering analysis. With this distance metric, the mean zero genes are no longer close to each other so that they do not form a cluster. They each belong weakly to one of the six smaller clusters and can be referred to as "noisy" genes. We often find genes like these in real data analyses and have noted that existing clustering algorithms have trouble identifying the true underlying clustering pattern in the presence of much noise. We applied each clustering algorithm to $B = 100$ samples with $K = 6$ clusters in PAM and KMEANS. For the PAM and HOPACH-PAM algorithms we used the cosine-angle distance. The results of these cosine-angle distance simulations were similar to those reported above for Euclidean distance except that, as expected, KMEANS performed very poorly since it is not able to cluster with respect to cosine-angle distance. PAM often finds the correct clustering pattern, and HOPACH-PAM nearly always does so. It was also observed that the PAM family of algorithms has the property that the medoids are a good representation of the true clustering patterns even in the presence of noise.

4 Data analysis.

We extracted a publicly available data set from the data base accompanying Ross *et al.* (2000). The authors performed microarray experiments on 60 human cancer cell lines (the NCI60) derived from tumors from a variety of tissues and organs by researchers from the National Cancer Institute's Developmental Therapeutics Program. The data set includes gene expression measurements for 9,996 cDNAs representing approximately 8,000 unique transcripts. Each tumor sample was cohybridized with a reference sample consisting of an equal mixture of twelve of the cell lines chosen to maximize diversity.

We used the normalized tumor:reference ratios, as in Ross *et al.* (2000). These were transformed to a log base 10 scale and truncated above and below, so that any ratio representing greater than 20-fold over- or under-expression was set to $\log_{10}(20)$. We applied HOPACH-PAM, always using the cosinus-angle distance and ordering relative to the neighboring cluster medoid.

4.1 Gene clustering.

In order to create a real example that might represent a similar problem to that explored in the simulation, we selected three very different types of cancer from those included in the NCI60: leukemia, colon and melanoma. We created a data set with all samples from these three types of cancer, which included six leukemia, seven colon, and eight melanoma cell lines. Next, we applied a subset rule in order to select all genes differentially expressed in a significant proportion of samples. We retained those genes where at least 25% of cell lines had a ratio corresponding with greater than 2-fold over- or under-expression. The 25% cut-off was chosen so that if a gene was differentially expressed in one type of cancer and not the other two, it would still be included. HOPACH-PAM was used to cluster the resulting data set of 3445 genes.

In the first level of the hierarchical tree, average silhouette suggested strongly a split into $k = 2$ clusters. These clusters corresponded closely (but not exactly) with a split into the genes with negative and positive mean log ratios across the cell lines (*i.e.:* genes over- and under-expressed on average).

In the second level of the tree, each of the two clusters from level 1 was split into two clusters. When these were reordered, some structure began to appear in plots of the data and distance matrices, but the clusters still seemed heterogeneous and no collapsing appeared to be necessary. So, the third level of the tree was examined. In this level, two of the clusters from the previous level were split into four clusters each and two into two clusters each. The resulting twelve clusters were reordered and the distance matrix was examined. Correlation between several clusters which had been united in a large cluster in the previous level indicated that some collapsing might be necessary. The clusters requiring collapsing were over-expressed genes, which we have

found tend to have more homogeneous expression profiles. After two collapsing steps, the reordered distance matrix had the desired block diagonal structure. The resulting ten medoids and corresponding clusters of genes were identified as the main clustering result. The tree was run down completely from this level. The final reordered distance matrix shows the clustering structure.

As discussed above, there are several ways to use the bootstrap to assess the variability of HOPACH-PAM clustering results. As an illustration, we applied PAM with $k = 10$, fixing the ten medoids identified in level three after two collapsing steps in each bootstrap sample. As a final step, we could perform post-screening and remove genes with lowest probability of belonging to their cluster or greatest distance from their cluster medoid.

Gene clustering was also performed with the automated HOPACH-PAM. As we have seen in other data sets, if allowed to do so, the automated HOPACH-PAM will converge to the first level of the tree with two clusters containing essentially the over- and under-expressed genes. If we want a more detailed clustering result, we can require that collapsing not occur until a lower level of the tree. In this case, the automated HOPACH-PAM clustering result is similar to the one identified manually. There were 8 clusters identified in the third level of the tree, and collapsing was mostly applied to the over-expressed genes.

4.2 Cell line clustering.

HOPACH-PAM can be applied to a transposed data matrix to obtain a final ordering of the samples. This ordering can be used to make an improved version of the reordered data matrix, where both genes and samples are ordered. The order obtained in this way for the cell lines in the leukemia, colon and melanoma data set was perfect in the sense that the three types of cancer were totally separated from each other (See Table 2, column 1). This result is explored further in the comparison section.

It is also of interest to cluster the cell lines using only the genes from each of the 10 gene clusters. For most gene clusters, the cell line clustering labels corresponded extremely well with distinctions between the different types of cancer. In many cases,

there were two clusters with one type of cancer clustering by itself (*e.g.*: gene cluster 4). In several cases, there were three clusters containing each of the types of cancer (*e.g.*: gene cluster 6). We could use these gene cluster specific cell line cluster labels to more accurately characterize the three types of cancer. For example, a gene cluster where two of the types cluster together might represent a biological pathway which is activated (or not activated) in two of these cancers, but not the other. A few gene clusters produced a cluster result which did not correspond with the distinction of the three types of cancer (*e.g.*: gene cluster 8). These gene clusters are also of interest as they may include genes which differentiate cell lines on some other biological basis such as tumor severity. In experiments with patient samples, this sort of clustering result could be correlated with clinical outcomes such as survival or treatment history.

4.3 Comparison with other hierarchical clustering methods.

We compared the final ordering from HOPACH-PAM to orderings produced by other hierarchical clustering routines and our proposed correlationordering() function, which maximizes the empirical correlation between distance $j - i$ in the list and the corresponding distance $d(i, j)$ across all pairs of cell lines (i, j) with $i < j$. We used HOPACH-PAM without collapsing because we were interested in a final ordering of the cell lines rather than the main cluster results. Our knowledge of the cell line labels was not used to supervise the clustering of cell lines, but rather to check (after clustering) if the various orderings were able to separate different types of cancer. We considered:

1. Correlationordering() function
2. HOPACH-PAM (without collapsing) with data adaptive numbers of clusters between 2 and 6 at each split (based on silhouettes)
3. HOPACH-PAM (without collapsing) with binary splits only
4. DIANA algorithm for divisive hierarchical clustering as implemented in Splus Kaufman & Rousseeuw (1990)

5. AGNES algorithm for average linkage agglomerative clustering as implemented in Splus Kaufman & Rousseeuw (1990)

6. AGNES algorithm for single linkage agglomerative clustering as implemented in Splus Kaufman & Rousseeuw (1990)

7. Average linkage agglomerative clustering as implemented in the publicly available packages **Cluster** and **TreeView** Eisen *et al.* (1998)

8. Single linkage agglomerative clustering as implemented in the publicly available packages **Cluster** and **TreeView** Eisen *et al.* (1998)

The correlationordering() function is not a clustering routine, but simply a proposed method for ordering elements using any distance metric. The other methods are all hierarchical clustering routines that produce an ordering of the clustered elements. HOPACH-PAM without collapsing and DIANA are divisive (or partitioning) methods, which start with the entire data set and divide it into clusters. The other algorithms are agglomerative methods, which start with every element in its own cluster and combine the closest clusters. In average linkage methods, the distance between two clusters is the average of the dissimilarities between the points in one cluster and the points in the other cluster. In single linkage methods, the dissimilarity between two clusters is the smallest dissimilarity between a point in the first cluster and a point in the second cluster (nearest neighbor method). We used the cosinus-angle distance in all of the algorithms. The data sets were selected *a priori*, not based on the performance of the algorithms. We repeated the analysis with randomly permuted data matrices in order to account for any effect of the original ordering of the cell lines on the final orderings produced.

These algorithms were applied to the leukemia, colon and melanoma data set from the data analysis above (See Table 2). The correlationordering() algorithm separated the three types of cancer and placed colon between leukemia and melanoma. The two versions of HOPACH-PAM produced the same ordering as each other, which also placed the three types of cancer totally separately with colon lying between leukemia and melanoma regardless of the original ordering of the data set. The other algorithms,

in contrast, placed the groups of different types of cancer in different orders depending on the original ordering of the data set. The DIANA algorithm placed one melanoma cell line with the leukemias. The AGNES algorithms placed this same melanoma cell line with the colon cell lines. The **Cluster** algorithm (single and average linkage) kept the colon cell lines together and distinct from melanoma, but spread the leukemia and melanoma cell lines throughout the ordering.

Correlationordering()	HOPACH-PAM $k \in [2, 6]$	HOPACH-PAM $k = 2$	DIANA	AGNES average	AGNES single	**Cluster** average	**Cluster** single
MEL10008	MEL10001	MEL10001	MEL10001	MEL10001	MEL10001	MEL10001	MEL10001
MEL10005	MEL10020	MEL10005	LEUK7010	COL4003	COL4003	LEUK7010	LEUK7010
MEL10002	MEL10014	MEL10008	LEUK7003	COL4010	COL4010	LEUK7019	MEL10002
MEL10014	MEL10021	MEL10007	LEUK7006	COL4015	COL4015	LEUK7006	MEL10021
MEL10020	MEL10007	MEL10021	LEUK7008	COL4002	COL4002	LEUK7003	MEL10007
MEL10021	MEL10008	MEL10020	LEUK7005	COL4017	COL4017	MEL10005	MEL10020
MEL10007	MEL10002	MEL10014	LEUK7019	COL4009	COL4009	MEL10002	MEL10008
MEL10001	MEL10005	MEL10002	COL4003	COL4001	COL4001	MEL10007	MEL10014
COL4003	COL4009	COL4001	COL4010	LEUK7019	LEUK7019	MEL10021	MEL10005
COL4002	COL4015	COL4003	COL4015	LEUK7010	LEUK7003	MEL10014	LEUK7006
COL4015	COL4010	COL4017	COL4002	LEUK7003	LEUK7006	MEL10020	LEUK7003
COL4017	COL4017	COL4010	COL4017	LEUK7006	LEUK7008	MEL10008	LEUK7008
COL4009	COL4002	COL4002	COL4009	LEUK7008	LEUK7005	COL4001	LEUK7005
COL4010	COL4003	COL4015	COL4001	LEUK7005	MEL10005	COL4003	COL4001
COL4001	COL4001	COL4009	MEL10005	MEL10005	MEL10008	COL4009	COL4003
LEUK7019	LEUK7006	LEUK7019	MEL10008	MEL10008	MEL10014	COL4017	COL4009
LEUK7006	LEUK7003	LEUK7010	MEL10021	MEL10014	MEL10020	COL4002	COL4017
LEUK7003	LEUK7008	LEUK7003	MEL10020	MEL10020	MEL10021	COL4015	COL4002
LEUK7008	LEUK7005	LEUK7006	MEL10007	MEL10021	MEL10002	COL4010	COL4015
LEUK7005	LEUK7019	LEUK7008	MEL10014	MEL10007	MEL10007	LEUK7008	COL4010
LEUK7010	LEUK7010	LEUK7005	MEL10002	MEL10002	LEUK7010	LEUK7005	LEUK7019

Table 2: Orderings of the cell lines in the NCI60 leukemia, colon, and melanoma data set produced by different hierarchical clustering methods.

This analysis suggests that HOPACH-PAM can improve upon the orderings produced by currently employed hierarchical clustering algorithms. Although the binary split and data-adaptive number of splits versions of HOPACH-PAM performed similarly here, we have seen evidence in similar data analyses that it is beneficial to allow greater than two clusters at each node of a HOPACH-PAM tree. The correlationordering() funtion also produced an ordering which separated the cell lines by cancer type. In fact, in data sets where clustering is not so clear, we have seen that this function performs better at creating an ordered list than any of the hierarchical clustering routines. Since correlationordering() allows any distance metric, it is a flexible tool for

creating an ordered list when the number of elements to order is not too large, such as the medoids of gene clustering (but not all genes) or the samples themselves, as illustrated here with cell lines.

5 Conclusions.

The application of the HOPACH methodology to a simulated and a publicly available data set illustrated some of its strengths. One advantage of this algorithm over other hierarchical methods is that the final ordering is unique. Also, partitioning algorithms allow for splits into two or more clusters so that the nodes of the hierarchical tree need not be binary. As we saw in the data analysis, lifting the binary split restriction can improve the final ordering produced, even when the algorithm is otherwise identical. We specifically like employing the PAM algorithm in HOPACH-PAM for two reasons. First, we have found that the medoids of clusters, which are themselves elements in the cluster, are much more robust profiles of a cluster than the cluster means. Secondly, the clustering in HOPACH-PAM can be based on any choice of distance metric, such as absolute correlation, which we have found is of interest to biologists since it clusters together anti-correlated over- and under-expressed genes that may be part of the same biochemical pathway. It is also intuitively appealing that this same distance metric is then used for the partitioning, the ordering of clusters, and the ordering of the elements within clusters.

A collapsing step can be applied at any level of the tree to unite similar clusters. The automated version of HOPACH-PAM identifies the main clusters by performing such collapsing steps until there is no more improvement in average silhouette. The simulation illustrated the importance of collapsing for identifying clustering structure. Collapsing serves two different purposes. First, collapsing can correct the number of clusters k in a split. Usually, we choose k by maximizing average silhouette over a range of values. Since PAM requires that the number of clusters be at least two, collapsing can be used to reverse a $k = 2$ split when the cluster is homogeneous enough to stop splitting. Also, using the average silhouette to choose k is an imperfect method, so that

we might want to make k smaller than the value indicated by the average silhouette by reuniting one or more clusters. In fact, one might even choose to make k too high on purpose and then collapse similar clusters. Secondly, collapsing can combine clusters across branches in the tree. For example, a split of a heterogeneous parent cluster may produce a cluster of genes which is more similar to clusters from another part of the tree than to the clusters with which it shares a parent. This group represents an error made at a higher level of the tree, which has only become apparent as the clusters got smaller and more homogeneous. Collapsing can unite this cluster with the most similar other cluster elsewhere in the level. By simply changing the cluster's labels to those of the other cluster (or vica versa), the structure of the tree is preserved. Thus, we do not just use collapsing to select the number of clusters in PAM at each node of the tree, but rather we use PAM to separate the elements into groups and then collapse to decide on the final clusters at that level.

Employed with collapsing, HOPACH-PAM is a "hybrid" clustering method which uses both partitioning and agglomerative steps. Agglomerative algorithms, such as single and average linkage as implemented in **Cluster** and AGNES, are equivalent to applying collapsing steps beginning at the bottom of the tree with each element initially in its own cluster. Our hybrid algorithm is able to provide a data-adaptive, sensible ordering of the clusters at each level of the tree. Hence, in particular, it produces a better final ordering of the elements. For **Cluster** and AGNES, the ordering of the clusters at every level of the resulting tree is determined by the initial ordering of the data. Because they start at the bottom of tree, there is no data-adaptive way to order the clusters at every level of the tree. On the other hand, HOPACH-PAM begins as a partitioning method so that it produces an ordering of the clusters (based on the distance metric) from the first level of the tree. This ordering is maintained after collapsing. To obtain such an ordering is particularly important for producing a final ordered list of genes which is unique and not dependent on the ordering in the original data set.

Contrary to the other orderings examined in the data analysis, the ordering produced by HOPACH-PAM is a unique function of the distance metric, assuming that

$PAM(data, k, d)$ at each node does not depend on the ordering of the data (which it should not because it aims to minimize a sum of distances). Like many clustering routines optimizing a criteria, however, PAM solves a minimization problem which has many local minima. In this sense, HOPACH-PAM shares (through the imperfect convergence of PAM) the dependence on the order of the original data matrix that we have noted in other algorithms. We could regain full uniqueness by rerunning PAM on a large number of permuted data sets (*i.e.*: starting values) and selecting the medoids which give the smallest sum of distances. We have noted, however, that this is not usually necessary since the cluster labels are often very stable so that the impact of the original ordering of the data on the final ordering is much less for HOPACH-PAM than for DIANA, AGNES or **Cluster**.

The HOPACH methodology as we have presented it is intentionally general enough to allow for adaptations in specific parts of the algorithm. For example, a range of different partitioning algorithms and distance metrics can be used. We have also suggested multiple ways to perform the ordering step and to make the decision to collapse. HOPACH algorithms need to combine partitioning and collapsing steps to produce an ordered hierarchical tree and corresponding list of elements. Visualization of the ordered distance matrix is an innovative approach to identifying the clustering structure. In addition, we like methods which fit into in the general statistical framework of van der Laan & Bryan (2001), so that we can treat the clustering output as an estimate of a true underlying parameter and perform inference with the bootstrap. Most commonly used algorithms fit into this framework, so the HOPACH approach remains very flexible. We see this flexibility, along with the hybridizing of partitioning and agglomerative algorithms to produce a sensible ordering, as HOPACH's greatest strengths.

References

Claverie, J.-M. (1999). *Human Molecular Genetics,* **8** (10), 1821–1832.

DeRisi, J., Penland, L., Brown, P., Bittner, M., Meltzer, P., Ray, M., Chen, Y., Su,

Y., & Trent, J. (1996). *Nature Genetics,* **14**, 457–460.

Eisen, M., Spellman, P., Brown, P., & Botstein, D. (1998). *Proc. Natl. Acad. Sci.* **95**, 14863–14868.

Herwig, R., Poustka, A., Moller, C., Bull, C., Lehrach, H., & O'Brien, J. (1999). *Genome Research,* **9**, 1093–1105.

Kaufman, L. & Rousseeuw, P. (1990). *Finding Groups in Data: An Introduction to Cluster Analysis*. New York: John Wiley & Sons.

Pollard, K. & van der Laan, M. (2001). Technical Report 96 Group in Biostatistics, University of California.

Ross, D., Scherf, U., Eisen, M., Perou, C., Rees, C., Spellman, P., Iyer, V., Jeffrey, S., Van de Rijn, M., Waltham, M., Pergamenschikov, A., Lee, J., Lashkari, D., Shalon, D., Myers, T., Weinstein, J., Botstein, D., & Brown, P. (2000). *Nature Genetics,* **24**, 227–235.

Tibshirani, R., Hastie, T., Eisen, M., Ross, D., Botstein, D., & Brown, P. (1999). Technical report Department of Statistics, Stanford University.

van der Laan, M. & Bryan, J. (2001). *Biostatistics,* **2**, 445–461.