# DEVELOPING AND VALIDATING A TOOL FOR COMPARING CLUSTERING RESULTS ON MICROARRAY DATA

by
Ted Laderas

A THESIS

Presented to the Department of Medical Informatics
and Clinical Epidemiology
and the Oregon Health Sciences University
School of Medicine
in partial fulfillment of
the requirements for the degree of
Master of Science
June 2004

School of Medicine
Oregon Health Sciences University

CERTIFICATE OF APPROVAL

This is certify that the M.S. thesis of
Ted Laderas
has been approved

Professor in charge of thesis

Member

Member

Member

**Table of Contents**

## *Acknowledgements*

Thanks to my thesis committee, especially Dr. Shannon McWeeney, without whom I would have never made it this far. Thanks for the encouragement, the guidance, and the urge to push my thesis and my work beyond my own meager expectations.

Thanks to my partner Jim, for encouragement and for helping me deal with my occasional frustrations about graduate school.

Thanks to Andrea Ilg, for guiding me through all of various departmental hurdles I needed to jump.

Special thanks to Dr. Laird Sheldahl, for assisting me to make sense of the functional annotations of the Cho dataset.

## *Abstract*

Microarray experiments offer the user the potential to monitor gene expression across thousands of genes at once. However, researchers are often left with a dimensionality problem - too few technological and biological replicates, and thousands of genes to monitor for differential expression. [1] The problem of finding interesting and novel genes within the thousands of genes on a microarray can seem akin looking for a needle in a haystack of needles. One highly popular approach to finding interesting genes for further study is finding similar patterns of expression within the data. A common hypothesis-generating approach, clustering, has been shown to have much potential in finding genes with similar function. However, there are a variety of clustering methods, and each has different strengths and weaknesses in finding patterns within microarray data.

In this thesis, I will first discuss some issues with acquiring and normalizing microarray data, which will be useful in discussing clustering methods. I will then discuss three types of clustering methods, namely hierarchical, partitional, and model-based clustering methods, highlighting the strengths and weaknesses of each approach. For microarray data that has subtle changes in expression across samples, different methods may give different answers, a point that is often overlooked. The underlying research question is to determine an effective way of comparing results across methods. This leads to my project goal, which is to develop and validate an evaluation framework for comparing clustering methods. Clustering methods used to develop this framework will be discussed, including program design and a review of the metrics used to evaluate

the clusterings. Validation of this evaluation framework will also be discussed, utilizing both simulated and real data. Interpretation of the results indicate that the tool has potential in finding consensus between clustering methods. The tool allows users to take appropriate caution interpreting a cluster if a gene of interest is only clustering with other known genes in one method. Finally, some possible future directions outside of the scope of the present study will be discussed.

# *Chapter 1: Background and Research Question*

## 1.1 Acquisition and Low-Level Analysis of Gene Expression Data

As a hypothesis-generating tool, microarray experiments offer the potential of monitoring the expression of thousands of genes simultaneously.

At its simplest, a microarray is a library of sequences printed to a glass slide. Each sequence is representative of a gene. There may be several sequences that represent a particular gene. There are two platforms available for doing microarray experiments, spotted microarrays, and oligonucleotide based microarrays. The spotted microarray platform is based on complementary DNA (cDNA) sequences, which are derived from libraries of cloned genes. cDNA chips can be made commercially or inhouse, and are printed using a robotic printing machine which individually spots each sequence to the glass slide. Sequences from these chips are usually about 100-200 base pairs long. In contrast, oligonucleotide based chips are manufactured using a photolithographic process, which attaches RNA bases to sites that have not been masked on the chip. Oligonucleotide sequences are also much shorter, on the order of 25-50 base pairs. [2]

After the chips are manufactured, labeled gene transcripts are hybridized to the slides. The gene transcripts are labeled with a fluorescent dye. For oligo-based arrays, a single sample is hybridized to the slide. When scanned by a laser, the fluorescence intensities obtained from oligo-based slides represent the absolute abundance of gene transcripts in the sample. In contrast, spotted arrays hybridize two samples to the slides:

a reference sample and a test sample. Each sample is labeled with a different fluorescent dye. These two samples competitively hybridize to the slide, and so the fluorescent intensities obtained from spotted slides only represent the relative abundances in the sample.

After the hybridized chips have been scanned with the laser, the spot intensities are then converted to numbers, and the raw data is obtained. However, it should be emphasized that the potential for systematic variation exists at each step of the data acquisition process. This unwanted variation contributes noise to the data, which may interfere with further data exploration. [1] Therefore, a key objective is to remove this systematic variation in order to study the interesting, or biological sources of variation. This process is known as preprocessing and normalization. We will discuss this process for both spotted and oligo- based arrays. Within oligo-based arrays, three of the most common methods will be discussed: Affymetrix's own Microarray Analysis Suite (MAS 5.0), and two model-based methods: the dChip method of Li and Wong, and Irizarry's own Robust Multiarray Average (RMA) method. [3-6]

Often, the first step in this process is log-transform the intensity values. This is done for two reasons. The first reason is that log-transforming makes effects additive, which makes normalization easier. [1] However, in the RMA process, the log transformation is performed after background subtraction. The second is that microarray data is highly skewed, and log transformation helps to even out these highly skewed distributions.

After log-transforming the intensity values, an estimate of the background noise due to nonspecific binding and optical noise is obtained, either by direct measurement, or

by mathematical modeling. [3-6] Because only a small percentage of genes are differentially expressed on a microarray, background subtraction will help to eliminate variation due to noise. However, improper background signal calculation can add noise to the data so care must be taken in determination of background levels.

In spotted arrays, a direct estimate of the background noise is found by estimating the signal level of the immediate area surrounding the spot. This approach is not possible with oligo-based arrays due to the fact that the sequences are so tightly packed on the chip. Affymetrix's Microarray Analysis Suite (MAS) divides the chip into smaller regions. A low percentile of the distributions of each region (2% for MAS) is found, and used as an estimate of the background noise for this region. [1] One caveat to this approach that should be noted is that this method of background subtraction is local. If one region has a completely different signal distribution than another region, some signal information can actually be lost through this method. The Robust Multiarray Average (RMA) method takes a different approach by fitting the raw measured signal to an additive model of noise plus signal. Noise is modeled as a Gaussian distribution and the true signal as an exponential distribution. The modeled noise can then be mathematically subtracted from the raw measured intensity. [3]

Another potential source of noise is cross-hybridization. Many oligo-based methods utilize the fact that the oligo sequences are represented by both a Perfect Match (PM) sequence, which matches the sequence exactly, and a Mismatch (MM) sequence, whose central base differs from the sequences by one base. The MM sequence was intended to be a probe-specific measure of cross-hybridization effects. Both MAS and dChip use the quantity PM-MM in their summarization processes. It has been shown that

on average, 30 percent of the MM values are larger than the PM values, which suggests that the MM values contain some signal. [1] The newer version of MAS, MAS 5.0, uses what is called the imperfect mismatch (IMM), which is a quantity derived from MM that is never larger than the PM values. [4]

dChip (also known as MBEI, Model Based Expression Index), in contrast, attempts to adjust for cross-hybridization and other effects through the iterative fitting of probe sets to its multiplicative model, which will be explained in the summarization section. [5, 6]

Additionally, it has been shown in spike-in studies, where known amounts of control cRNAs have been added to a sample, that MM intensities also increase along with the PM intensities as more cRNA is added. [3] This suggests that the MM intensities contain some signal information and that PM-MM may not be a biologically meaningful expression measure. This was the motivation behind Irizarry et al's use of just PM in calculating expression indexes. [3] A newer version of dChip utilizes only the PM values.

The next step in the preprocessing process is normalization. Many potential sources for systematic variation exist, and these should be corrected in order to allow for the comparison of the expression indices from one slide to the next. It should be emphasized that correcting for these variations is an iterative process.

One source of systematic variation that is most easily seen in spotted microarray data is the variation due to the print-tips (or pins) used to spot the sequence to the slide, as the pins can become bent or contaminated and impact an entire group of spots. When the same sample is used as both the reference and test sample (known as self-self

hybridization), a notable difference in the single distributions is detected when the genes are separated by print-tip group prior to normalization (see figure 1.1.1). Other sources of variation include variations in experimental technique, differing efficiencies of the dyes, spatial effects, and sample quality.

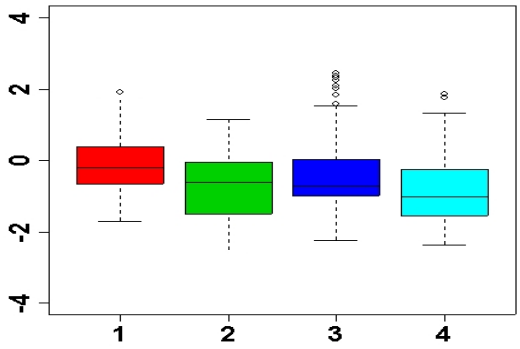



**Figure 1.1.1**. Self-self hybridization of a cDNA microarray showing variations due to four print tips. [7]

There have been varying approaches suggested to correct for these sources of variation in spotted microarrays. Perhaps the most popular and successful approach has been the use of locally weighted regression (lowess) to correct for intensity-dependent biases. Lowess correction has been shown to reduce systematic variation between print-tip groups, dye channels, and other sources of variation. [7, 8] The key assumption in lowess normalization is that the distribution of the dataset is symmetric, which may not be the case if the array is biased towards highly expressive sequences.

There are similar sources of variation with oligo-based arrays that must be removed. Each method takes a different strategy towards normalizing arrays.

MAS's approach to normalization is the simplest approach. In order to normalize arrays for comparison, the array with the median average intensity is chosen to be the baseline array. The intensity of each array is globally scaled such that average intensities

of that array match the average intensity of the array. This is the crudest form of normalization. [4]

MBEI's approach also utilizes a baseline approach. However, the normalization process is more sophisticated in that the normalization is based on a set of genes that is non-differentially expressed across arrays, known as an invariant set. Determining which genes belong to the invariant set is a nontrivial process and is described further in [5]. The baseline array values of this invariant set are plotted against the values of the to-be-normalized array and a smoothing spline is fitted to the data in order to determine the relation between this array and the baseline array. This process is done for each array.

RMA uses quantile normalization in order to normalize across arrays. In quantile normalization, the distributions of each array are forced to fit a chosen distribution. This distribution is chosen by averaging the distributions across all arrays (hence the term Multiarray Average). [3]

The point is that the application of these normalization methods takes human judgment in order to find the most effective normalization for the data. Data may need to be iteratively normalized within groups to effectively remove systematic variation, such as within print-tip group, and thus this is one processing step that cannot be easily automated.

Oligo-based microarrays add the complication that multiple oligonucleotide sequences represent a single gene (which Affymetrix calls a probe set), and an summarization process needs to be applied to the intensity values over all the sequences in that probe set. If there are multiple transcripts that represent a single gene on a cDNA array, this summarization step is also necessary for that array.

MAS originally used a summarization known as AvDiff that took the average of the differences between all probe pairs within the probe set as the expression index for that gene sequence. This has not had the problems of the distribution of MM values, but was not robust to outlier pairs within the probe set, which may be less informative than the other pairs within the probe set. [4] However, the latest version of MAS uses the Tukey's Biweight of the PM-IMM pairs for summarization. Tukey's biweight can be thought of as a mean that is robust to outliers. MBEI uses as its expression indices the maximum likelihood estimates (MLE) of its fitted model. [6]

The dChip model attempts to account for probe-specific effects through a multiplicative model of the PM-MM differences for each probe. This model is constrained by forcing the sum of squares to be equal to the number of probe pairs in this model. Implicit in the model is an estimate of the standard errors of the expression indexes. [6] The model is iteratively fit to the data by first identifying and excluding arrays with high errors in the expression indexes, then excluding probe sets with high errors, and finally excluding those probes with high errors. Thus, the iterative nature of fitting the MBEI model allows for the detection and exclusion of outliers at every level in the dataset. Li and Wong have suggested that this approach is most successful when more than ten arrays are used in the analysis. [5, 6]

RMA also uses a model to summarize expression values across a probe set. However, RMA first log transforms the normalized values so that the effects are additive. It then fits an additive model to the data using median polish. Median polish is a fitting method that is robust to outliers. The expression indices can then be estimated from the fitted model. [3]

After the data has been preprocessed, normalized, and summarized, a filtering process may be applied. Filtering is a data-reduction step. Even after normalization, very small fold changes (i.e., the ratio of intensity values between two different experimental conditions) among samples may be due to noise, and so removing those genes that are not differentially expressed will improve the analysis of results. [1] Two common criteria used for filtering include: choosing an arbitrary threshold for fold-change, or filtering based on whether the differences in expression across samples are statistically significant. Another option, sometimes used for oligo-based arrays, is to filter out genes based on whether the gene is present or absent. This presence/absence call is often made by comparing the expression of a gene with that of the distribution of intensities from negative control genes (e.g., genes that are not expressed under that experimental condition).

After the normalization and filtering processes, a working data set is obtained, which is known as a gene expression matrix. In a gene expression matrix, each column represents a different sample, and each row represents a different gene. In time studies, each column may correspond to an individual time point. If there are $p$ genes and $n$ samples, this matrix M is a [$p$ x $n$] matrix. Each entry $M_{ij}$ in M represents the abundance of gene transcript $i$ for sample $j$. [9]

## 1.2 High-Level Analysis of Gene Expression Data

The next step in utilizing the gene expression matrix is dependent on the type of microarray experiment. In biomarker studies, the interest is finding genes that are

differentially expressed across cases and controls. These genes can be found through the filtering process described above. However, in time series experiments, the focus is on tracking how large groups of genes are expressed over a time period. Finding genes that have similar temporal expression profiles is an important goal of these studies. [2] In genetic pathway experiments, the focus is on finding genes that are expressed similarly across a number of situations. In these cases, filtering becomes inadequate.

Regardless of the type of experiment, the researcher is limited by which statistical tools he can use on the data because of the dimensionality problem associated with the gene expression matrix. Most statistical tools for analyzing such data sets rely on the assumption that $p$ and $n$ are of similar magnitude, which is not the case with gene expression matrices. Gene expression matrices have a large set of genes, but a much smaller range of samples. This is especially problematic with time series experiments, where the time points are discrete and sparse. Normal methods for analyzing time series will not work on such data. Thus, there are few statistically rigorous testing frameworks available to explore the data set.

Given these problems, many investigators utilize clustering techniques to aid in mining and visualizing the data. Clustering is a hypothesis-generating tool that partitions the data into smaller chunks, or clusters. The data may be partitioned either by gene (clustering on rows) or by sample (clustering on columns). The data is partitioned based on the similarity of the expression profiles. Thus, for any clustering, a measure of similarity, also known as a similarity metric, is needed. [9] The similarity metric can be magnitude-based (Euclidean) or based on similar patterns of ups and downs in the profile (correlation). In successful clusterings, similarity in expression profiles can lead to

similarity of function. It should be noted that any results from clustering must be confirmed by some external method, whether through combining clustering with functional annotations or follow up experiments.

One visualization technique associated with clustering is the Eisengram, or heatmap. Developed by Eisen, [10] the heatmap provides an easy way to visualize patterns of expression. Heatmaps utilize a false color representation; the most common color scheme is of downregulated genes are represented by red, and upregulated genes are represented by green. [10] The relative expression level of the genes is represented by the intensity of these colors. Most importantly, heatmaps provide the user with a way of comparing gene expression profiles visually.

## 1.3 Early Success Stories with Clustering

One of the earliest and most successful applications of clustering microarray data was that of Eisen. [10] He applied a variation of average-link hierarchical clustering to two microarray data sets. Average-link hierarchical clustering is a data exploration method that produces a dendrogram representation that relates the gene expression profiles by a similarity measure. The first data set was a time series of the growth response of human cells, and the second was a combination of data sets on yeast cells, which included mitotic cell cycle data, sporulation, as well as others. [10] He clustered all the genes in the data set with known functional annotations. With both data sets, he showed two important results. The first is that redundant sequences representing a single gene clustered together. The second, and more important result, is that genes of similar function clustered together. As can be seen from figure 1.2.1 below, there is a strong

similarity in the expression profiles across any cluster. [10]

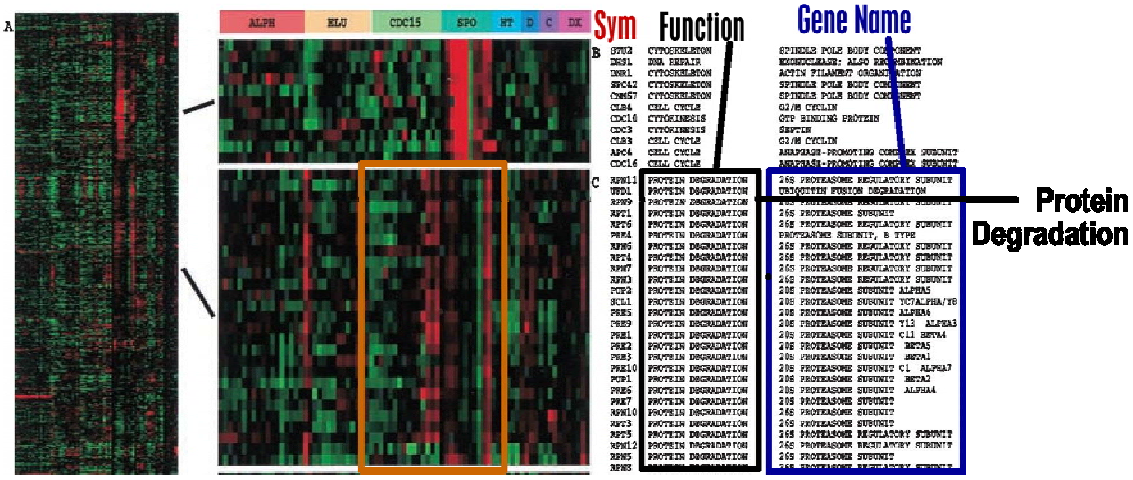



**Figure 1.2.1** A portion of Eisen's clustered data in heatmap form. Note the similiarities in expression of genes within a cluster (especially within the orange box), and the functional annotations of the bottom cluster are the same (that of protein degradation). Reproduced from Eisen, et al [4]

However, there are some properties of the datasets that led to the success of this study. Both of the domains Eisen studied exhibited large fold-changes in expression. This is especially true with the yeast data set, as cell cycle data generally has very large fold-changes in expression, on the order of 64-fold. In addition, the geometry of both datasets was well known. As a result of these properties, Eisen's clusterings were relatively stable across a variety of clustering methods.

In contrast, the majority of microarray data sets may exhibit more subtle differences between conditions (e.g., not have such large fold-changes in expression), which means that the patterns of expression that clustering relies on may be obscured by noise in the data. Because clustering methods (especially correlation-based methods) are sensitive to noise, clusterings may not be reproducible across a variety of methods. [1] Clearly, some way of comparing clusterings from different methods is needed, which will

be discussed in section 3.

## 1.4 A Motivating Example for Comparing Clusterings

Microarrays often contain sequences that have no functional annotation, either unannotated gene sequences or sequences derived from sequenced tagged sites (STS). Part of what makes microarray studies exciting is the possible discovery of new genes with unknown functions. As the Eisen study shows, genes clustered based on gene expression levels may also end up clustering by function.

Suppose a biologist uses a clustering method on her microarray data and finds such an unannotated sequence clustering with genes of known function. Such a discovery is potentially exciting. However, suppose another lab uses the same microarray data but a different clustering method on the data, and that gene no longer clusters with the same genes as before. Is that gene sequence still worth studying?

## 1.5 Research Question

In order to help answer this question, we propose the development and validation of a evaluation framework for comparing clusterings obtained from different clustering methods. The approach we use in this study is that of obtaining consensus among clustering results. This will allow a researcher to weigh all of the evidence and determine how reliant the results are on the method used.

## 1.6 Project Goals

The end goal of this project is to develop an evaluation framework that will allow for the comparisons of different clustering results. The framework will have two aspects: a numerical report with metrics comparing aspects of the various results, presented in a table format, and a visualization piece that will allow for visual comparison of clustering results.

The framework will then be validated using both a "gold-standard" data set, that of the Eisen Yeast Cycle data set, and a simulated data set. The Eisen Yeast Cycle data set is a good candidate for validation as it has been extensively analyzed and has an obvious geometry. Simulated data sets offer opportunity for validation in that they allow for comparison of clustering results with the known true clustering of the data.

## 1.7 Use Cases

There are two possible use cases to consider in the design of the evaluation framework. The simpler case to consider is that of an informatics researcher wishing to compare the performance of different clustering methods on a dataset where the answer is already known. In this case, the program asks which methods the user wants to compare. The program will then ask for appropriate parameters for each method. The program then runs through all clustering methods and produces a report comparing those methods. This framework will also allow new methods to be added to compare performance.

The second use case would be use by a biologist. In this use case, a biologist, wishing to find new sequences to pursue, would input her expression matrix into the program.

# *Chapter 2: Review of Clustering Methods*

## 2.1 Cluster Analysis

Picking a clustering method involves three choices:

1) The similarity metric,
2) The clustering type,
3) A particular implementation of the clustering type.

The first step in picking a clustering method is picking the similarity metric. The choice is important to the clustering method because it is dependent on what qualities of the data one wishes to capture in the data exploration. [11] Two metrics that are commonly used are Euclidean distance and correlation distance. Euclidean distance measures the absolute similarity of two expression profiles. It is calculated using the following equation:

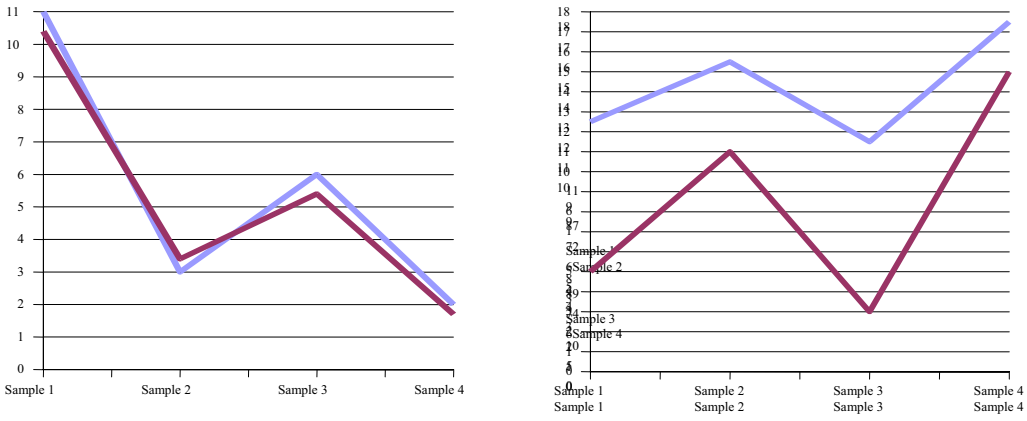
$$\|x - y\| = \sqrt{\sum_{i=1}^{p} (x_i - y_i)^2} \,. \qquad \text{(eqn 2.1.1)}$$

Where $x$ and $y$ are genes, and $x_i$ and $y_i$ represent the value for that gene under the ith experimental condition. Figure 2.1.1a shows two expression profiles that have a small Euclidean distance.

Correlation distance, on the other hand, measures how much of a linear relationship the two profiles have. Correlation distance is calculated by the following equation:

$$r(\vec{x},\vec{y}) = \frac{1}{m}\sum_{i=1}^{m}\left(\frac{x_i - \bar{x}}{\sigma_x} \times \frac{y_i - \bar{y}}{\sigma_y}\right) \qquad \text{(eqn 2.1.2)}$$

Where x and y are genes, $x_i$ and $y_i$ represent the value for that gene under the ith experimental condition , $\bar{y}$ represents the average value of y over all experimental conditions, and $\sigma_y$ represents the standard deviation of y over all experimental conditions. Correlation distance is useful in capturing patterns of up and down expression even when two profiles are not absolutely similar. [11] Figure 2.1.1b shows two profiles that have a small correlation distance.




(a) (b)

**Figure 2.1.1** Two different gene expression profiles. In figure 2.1.1 (a), there are two expression profiles that are very close in a Euclidean sense. In figure 2.1.1 (b), there are two expression profiles that are close in a correlation sense.

After the distance metric is picked, a distance matrix D must be calculated. This is a *p* x *p* matrix that contains the distances between every gene expression profile. An entry $D_{ij}$ represents the distance between gene profile *i* and gene profile *j* (Note that this matrix is symmetric, because $D_{ij} = D_{ji}$ and $D_{ii} = 0$). This matrix is the data format that the particular clustering algorithms act upon.

## 2.2 Clustering Methods: An Overview

After the distance matrix is generated, a type of clustering must be chosen. Although there are numerous approaches, we will discuss two of the most common types of clustering: hierarchical and partitional clustering.

Within the type of clustering, a particular implementation must be chosen. Each method differs by the assumptions it makes about the data, the structure it imposes on the data, and the output from that method. A summary table comparing the two methods discussed is provided at the end of this section.

The first type of method we will discuss is hierarchical clustering. Hierarchical clustering is a relatively popular method because it requires very few assumptions about the data, and it imposes very little structure upon the data. Hierarchical methods produce a dendrogram, a tree structure whose height represents how distant cluster elements are from each other. Clusters can be chosen at a fixed level of similarity by "cutting" the tree at a particular height. [1] More commonly, however, clusters are hand-picked from the dendrogram. Thus, a dendrogram can be thought of as a family of related clusterings.

The dendrogram can be produced by two methods. The first method is known as bottom-up, or agglomerative clustering, which starts with each element as a singleton which is then clustered with other singletons, until a single cluster is formed. [1] Examples of agglomerative clustering include complete-link, single link, and average link clustering. The second method is known as top-down, or divisive clustering. Divisive clustering starts with all elements as members of a single cluster which is then divided into smaller clusters, until the singletons form their own clusters. Examples of divisive

clustering include 2-means (also known as Tree Structured Vector Quantization) and Diana. [1, 6, 7]

Two hierarchical methods were chosen to be studied, one agglomerative and one divisive. The agglomerative method chosen was that of Unweighted Pair Group Method with Arithmetic Mean (UPGMA), also known as average link clustering. This method is one of the most common agglomerative clustering algorithms used in cluster analysis. This is also the method that was used by Eisen in his original paper. In UPGMA, one starts with the original p x p distance matrix, where p is the number of genes. The matrix is scanned for those two elements that are closest to each other. These two genes are combined into a single cluster. A new distance matrix with dimensions (p-1) x (p-1) is then calculated. The distance between a single gene i and a cluster of genes is calculated by averaging the distances from that single element i to all the members in the cluster. Likewise, when the distance between two clusters C1 and C2 is calculated, it corresponds to the mean distance of all the elements of C1 to all the elements of C2.

The divisive method chosen was a variant of the MacNaughton-Smith Algorithm known as Diana. Diana starts with the entire population and starts to form a "splinter group" by picking the gene with the largest mean dissimilarity to all the other points. The remaining genes are then swapped to this splinter group until no gene in the original group is closer (in mean dissimilarity) to the splinter group. Similarly, these two groups are further divided until the singletons are reached. [12]

There are caveats to hierarchical clustering that must be noted. If the clustered data is noisy, the dendrogram can have very little biological meaning. Another caveat specific to agglomerative clustering is that higher level clusters (those clusters that are

near the top of the dendrogram) tend to be less reliable than the lower level clusters (those clusters near the bottom of the dendrogram). [13]

The second type of method we will discuss is partitional clustering. Whereas hierarchical clustering requires no parameters to produce the clustering, in partitional clustering, the number of clusters K must be specified in these methods. Thus, a certain amount of information must be known for these clustering methods to be used. Partitional clustering forces the data into the K partitions specified. Thus, if K is not known or obvious from the geometry of the data, this method can give nonsensical results, forcing the data to partition into nonsensical partitions. [11, 14]

Partitional methods are iterative, meaning that cluster assignments can change from iteration to iteration. Typically, these methods are iterated until the cluster assignments do not change. Examples of partitional algorithms include k-means and Self Organized Maps (SOM). [1, 15]

The two partitional methods that were chosen in this study are K-means and Self Organized Maps (SOM). K-means is an iterative algorithm that has only two steps. The algorithm can be started with either a set of k-cluster centers randomly assigned or specified from previously known information. Now the algorithm can start. In step one, those genes that are closest to the cluster centers are found and made members of that cluster. In step two, the cluster centers are recalculated by calculating the vector mean of each member of the cluster. These steps are repeated until there are no changes in assignment.

Self-Organized Maps are similar to k-means. However, the cluster centers (referred to as prototypes) are linked together in either a one or two dimensional

geometry, such as a one-dimensional grid or a two-dimensional grid. This grid of prototypes is randomly assigned, much like the cluster centers of k-means. For each iteration, a gene is selected. The closest prototype Np is moved towards that gene the most. The other prototypes are moved as well, but with a weight dependent on their "connectedness" (i.e., their relative position in the network with respect to another prototype) to the closest prototype Np. The algorithm is iterated for a set number of iterations, usually on the order of 20,000 to 50,000 iterations. Thus, the geometry of the SOM imposes an additional structure to the clusterings. [1, 15]

Table 2.2.1 provides a summary of the salient points about each type of clustering.

| | **Hierarchical** | **Partitional** |
|---|---|---|
| *Output* | Dendrogram | Clusters with cluster centers |
| *Assumptions* | Very few assumptions made about data | Must know number of clusters (K) beforehand. |
| *Structure imposed on Data* | Very little structure imposed on data | Impose # of structures of data |
| *Notes* | Good for when very little is known about data. | Good for when there is an obvious pattern of expression or geometry of elements is obvious, or have other previous knowledge. |
| *Caveats* | Dendrogram may have very little biological meaning | If K is wrong, clusters may be nonsensical. |
| *Examples* | Agglomerative/Top-Down (UPGMA) and Divisive/Bottom-up (Diana) | K-means and Self-Organized Maps (SOM) |

**Table 2.2.1** Summary of clustering types.

# *Chapter 3: Implementation of program*

## 3.1 Implementation of Framework

Both the visualization and numerical components will be implemented within the R/Bioconductor framework. R (http://www.r-project.org) is an open-source statistical programming language that is widely used among statisticians. [16] Bioconductor (http://www.bioconductor.org) is a bioinformatics analysis package for R that includes gene annotation and microarray analysis tools. [16] This framework was chosen for a variety of reasons. First of all, many of the clustering methods we wish to investigate are already implemented and validated in R, easing the development time of such a tool. Additionally, Bioconductor offers some visualization capablities of microarray data that can be easily extended for the visualization component. Finally, R is easy to web-enable, which means that the final implementation of the evaluation framework can be implemented as a web-based tool.

## 3.2 Program Workflow

The program workflow can be seen in figure 3.2.1. A list of clustering methods to be compared is supplied to the program. This list is currently supplied through the use of a simple graphical user interface (GUI). The appropriate output functions for each clustering method must also be supplied. These output functions are responsible for performing the clustering method on the expression matrix and returning the cluster

assignments in a standardized data format. The output functions themselves have a generic interface that accepts two inputs a list of appropriate parameters for that clustering method and the expression matrix itself. The generic interface design enables new clustering methods to be added to the framework relatively easily.

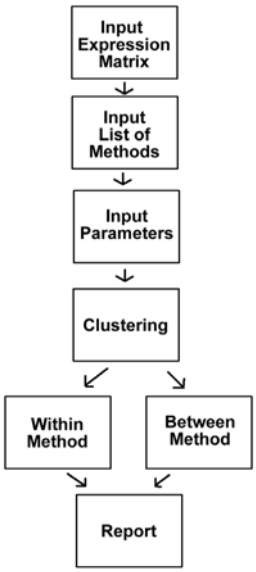



**Figure 3.2.1** Simplified workflow of program.

The program then asks for the appropriate parameters for each clustering method, and binds these parameters to the appropriate output function in the form of a list. Each slot in this list contains two objects: the output function for a clustering method, and a list of the associated parameters. The list format, along with the generic interface of the output functions, allows the program to easily execute any number of clustering methods to be compared.

After the clustering methods are executed, a list of the cluster assignments is obtained. At this point, this list of clustering assignments is submitted to the metrics portion of the program. This portion of the program is separated into two sections within method metrics and between-method metrics. The separation is necessary because each

kind of metric accepts differing inputs. Specifically, the within-method metrics need the cluster assignments as well as the calculated cluster centers.

Between-method metrics, on the other hand, need two sets of cluster assignments, as well as a calculated contingency table. The contingency table compares the individual k clusters from a clustering C to the individual clusters k' from a second clustering C'. The contingency table counts the number of members in common in a cluster in C with a cluster in C' in all possible combinations.

After the metrics have been calculated, the program produces a report of both kinds of metrics, and also returns a list of cluster assignments for each method. This list of cluster assignments can be used to analyze the performance of the clustering methods if the true assignments are known.

## 3.3 Standard Data Format

In order to reduce the amount of work dedicated to data conversion, a standard data format was for the expression matrix was decided upon. The format for the expression matrix is simply a matrix that contains the expression data, with the row labels corresponding to the gene or sequence being studied and the column labels corresponding to the experimental condition. One requirement is that the row labels must be unique, as they are used in calculating the Variation of Information and the Jaccard Index. R has many facilities for reading a dataset in a number of formats and parsing it into this standard data format.

## 3.4 List of Methods/Parameters

After the program accepts the expression matrix as input, the program queries the user for a list of clustering methods to study. Currently, the program asks for parameters for all six methods through a simple graphical user interface (GUI).

## 3.5 Clustering

After the methods and parameters are decided upon, they are bundled into a list format along with the appropriate output function. The output function performs the actual clustering and returns the results in a standard output format. The clusterings are returned in the form of a numeric vector which represents the cluster assignments for each gene in the expression matrix.

The strength of the list format for executing the clusterings is that it is easy to implement new clustering methods and extend the capabilities of the framework. In order to implement a new method, the appropriate parameters need to be implemented in the framework, and a new output function needs to be written.

The final clusterings are also returned in the form of a list. Each slot in the list contains the parameters for that clustering method, along with the clustering results. The list format allows for easy computation of the metrics. This is especially important in the case of the between-method metrics, which compare the clusterings from one method to the other. Figure 3.5.1 shows an example of how the clusterings are saved as numerical assignments for each gene.

| ProbeSetID | UPGMAEUC | UPGMACOR | DIANAEUC | DIANACOR | SOM1 | KMEANS1 |
|---|---|---|---|---|---|---|
| 203741_s_at | 1 | 2 | 2 | 1 | 3 | 3 |
| 200782_at | 1 | 2 | 2 | 1 | 2 | 3 |
| 209430_at | 1 | 2 | 2 | 1 | 3 | 3 |
| 212586_at | 1 | 2 | 2 | 1 | 3 | 3 |
| 210346_s_at | 1 | 2 | 2 | 1 | 2 | 3 |
| 214683_s_at | 1 | 1 | 2 | 1 | 2 | 3 |
| 208091_s_at | 1 | 2 | 2 | 2 | 3 | 3 |
| 208873_s_at | 1 | 2 | 2 | 2 | 3 | 3 |
| 221740_x_at | 1 | 2 | 2 | 1 | 2 | 3 |
| 216733_s_at | 1 | 2 | 2 | 1 | 2 | 2 |
| 204057_at | 1 | 2 | 2 | 1 | 2 | 2 |
| 212150_at | 1 | 2 | 2 | 1 | 3 | 3 |
| 221899_at | 1 | 2 | 2 | 1 | 2 | 2 |
| 217779_s_at | 1 | 2 | 2 | 1 | 3 | 3 |
| 217779_s_at | 1 | 2 | 2 | 1 | 2 | 2 |
| 204020_at | 1 | 2 | 2 | 1 | 2 | 3 |
| 201811_x_at | 1 | 2 | 2 | 2 | 2 | 3 |
| 203221_at | 1 | 2 | 2 | 1 | 3 | 3 |
| 218396_at | 1 | 2 | 2 | 1 | 2 | 3 |
| 207655_s_at | 2 | 2 | 1 | 1 | 1 | 1 |
| 219471_at | 2 | 2 | 1 | 2 | 2 | 2 |
| 44790_s_at | 2 | 2 | 1 | 1 | 2 | 2 |
| 212681_at | 2 | 2 | 1 | 1 | 1 | 1 |
| 221958_s_at | 2 | 2 | 1 | 2 | 1 | 2 |
| 204681_s_at | 2 | 2 | 1 | 2 | 2 | 2 |
| 212956_at | 2 | 2 | 1 | 2 | 1 | 1 |
| 219737_s_at | 2 | 3 | 1 | 2 | 2 | 2 |
| 213624_at | 2 | 2 | 1 | 2 | 2 | 2 |
| 39318_at | 2 | 2 | 1 | 1 | 1 | 1 |
| 209995_s_at | 2 | 2 | 1 | 1 | 1 | 1 |

**Figure 3.5.1.** Example of storage of clusterings. Columns represent different clustering methods and rows represent different genes. Clusterings consist of numerical assignments of a gene to a cluster in a method.

## 3.6 Properties of a Good Metric

Before the metric portions of the tool are discussed in detail, properties of a good metric need to be discussed. For the purposes of this study, a good metric must have three properties:

1) *A good metric should be comparable across results*. The metrics used in this study need to be normalized with respect to the size of p and the number of clusters.
2) *A metric can be bounded or unbounded.* Bounded metrics (those metrics that range between a certain numerical range, such as [0,1]) are useful, but unbounded metrics (metrics that may range to infinity) are also useful. One example of a bounded metric is this study is the Jaccard Index, which has a range of [0,1]. One unbounded metric used in this study is the Variation of Information, which has a range of $[0, \infty)$
3) *A good metric needs to have some physical meaning*. Obviously, the metric has to relate some useful property of the clustering to be useful.

Note that it may not make sense to compare metrics across types of methods (such as comparing hierarchical and partitional) if the actual answer is not known.

## 3.7 Within-Method Metrics

Within-method metrics calculate a single property on one clustering result. Three types of within metrics are *Homogeneity*, *Separation*, and *Average Silhouette Width*.

### Homogeneity

*Homogeneity* is a metric that measures the amount of variation within clusters. [17] Homogeneity is defined as the average distance of an element to its cluster center over all data points.

$$H_{ave} = \frac{1}{p}\sum_{i} D(g_i, F(g_i)) \qquad \text{(eqn 3.7.1)}$$

Where p = # of genes, D is the distance function, $g_i$ is a gene, and $F(g_i)$ is its cluster center. Homogeneity is an unbounded metric, but because it relies on distance measures, the homogeneity of one clustering can be directly compared to another. Homogeneity measures how tightly each cluster is defined. Figure 3.7.1 provides a visual depiction of homogeneity.

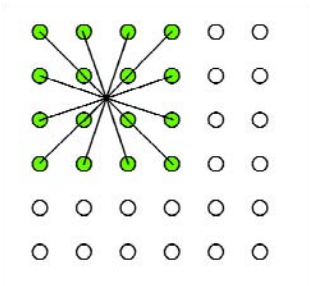


**Figure 3.7.1** The homogeneity of one cluster measures how tightly bound its elements are to its cluster center.

## Separation

*Separation*, on the other hand, is a measure of the amount of variation between clusters, as depicted in figure 3.7.2. [17]

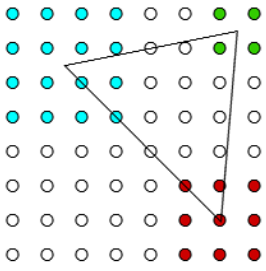


**Figure 3.7.2** Separation measures how far apart clusters are.

The distances between each cluster centers are calculated, and normalized according to the membership size of the clusters. This quantity is averaged for all distances between cluster centers.

$$S_{ave} = \frac{1}{\sum_{i \neq j} |C_i||C_j|} \sum_{i \neq j} |C_i||C_j| D(F(C_i), F(C_j)) \qquad \text{(eqn 3.7.2)}$$

where $|C_i|$ and $|C_j|$ are the # of points in Cluster *i* and Cluster *j*, D is the distance function, and $F(C_i)$ is the center of Cluster *i.*

Both Homogeneity and Separation require cluster centers as an input. Code to calculate the cluster centers was implemented in R, as was code to calculate the Homogeneity and Separation metrics.

## Average Silhouette Width

*Average Silhouette Width* is a metric proposed by Kaufmann and Rouseeuuw. Basically, a silhouette measures the degree of uncertainty of assigning an element to two clusters: 1) the cluster with which it is assigned, and 2) the cluster with the next closest cluster center. [18, 19] For each data point *i*, two quantities are calculated: the average distance of that point to its own cluster, denoted $a_i$. The average distance of that point to the next closest cluster center is denoted by $b_i$. The silhouette value for a single data point *i* is then:

$$sil_i = \frac{b_i - a_i}{\max(b_i, a_i)}. \qquad \text{(eqn 3.7.3)}$$

The average sihouette width $\overline{sil}$ is the silhouette width calculated for every data point:

$$\overline{sil} = \sum_i \frac{sil_i}{n} \qquad \text{(eqn 3.7.4)}$$

If the clusters are well defined within a clustering, the average silhouette width will be closer to 1. Clusters that are ill-defined will have an average silhouette width closer to 0. Code for average silhouette width, `silhouette`, previously implemented in R as part of the `cluster` package, was used to implement this metric within the framework. The distance metric we use in this study for the implementation of the silhouette function was correlation distance. Thus, this metric as currently implemented detects how strong the linear relationship a profile is to its cluster and the neighboring cluster.

## 3.8 Between-method Metrics

We will now discuss methods that compare agreement between two clusterings. The two metrics we will discuss here are the Jaccard index and Variation of Information.

### Jaccard Index

The *Jaccard Index* measures how frequently pairs of genes stay in the same cluster across clusterings. [13] For example, in one clustering C, genes a and b may be in a cluster together, but in another clustering C', they may be in two different clusters.

Many similar comparison metrics exist, which all rely on counting the following pairs of genes or elements:

$N_{11}$ = number of pairs of genes that cluster together in both C and C'
$N_{00}$ = number of pairs of genes that are in separate clusters in both C and C
$N_{10}$ = number of pairs of genes that cluster together in C, but not C'
$N_{01}$ = number of pairs of genes that cluster together in C', but not C

The Jaccard index is expressed simply as

$$J(C,C') = \frac{N_{11}}{N_{11} + N_{01} + N_{10}} \qquad \text{(eqn 3.8.1)}$$

The Jaccard index offers a sense of the overall agreement of clusterings in terms of pair membership. The Jaccard index reports the fraction of pairs that cluster together in both C and C' relative to those pairs that cluster together in at least one clustering. Therefore, if the clusterings have a relatively similar pair membership, the Jaccard index should be close to 1, because $N_{01}$ and $N_{10}$ should be small compared to $N_{11}$. Those clusterings that have relatively few pairs in common ($N_{11}$ is small compared to $N_{10}$ and $N_{01}$), will have a Jaccard Index close to 0.

The Jaccard index gives the user a sense of how robust the clusters are across two different methods. The Jaccard index can help to address whether the unknown gene is worth studying if it clusters differently across methods. If the majority of the gene pairs belong to $N_{11}$, the clusterings are relatively robust across the two methods.

Code to implement the Jaccard Index was written and implemented for the framework.

## Variation of Information

Another method for comparing two clusterings relies on information theoretic notions. This is the metric of *variation of information*. [20] Intuitively, two clusterings C and C' may have information in common; one artificial example is presented in figure 3.8.1. Two different clusterings both contain two clusters. There is a cluster that overlaps in both of them (the cluster in the center), but the other cluster in both clusterings has no overlap. The cluster in common can be thought of as where the two clusterings overlap informationally, also known as the *mutual information* between the two clusterings. If we subtract this quantity from the original clusters, we have what is informationally different between the two clusterings. This is known as the *variation of information*.

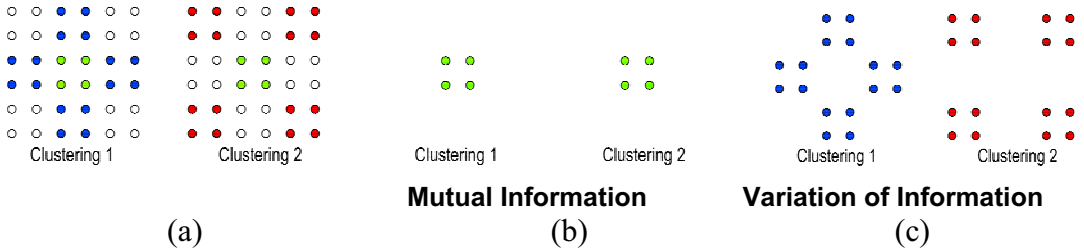



**Figure 3.8.1** Illustration of variation of information. 2.3 a) shows two sets of two clusters, one of which is the same for both sets. b) This common cluster is the *mutual information* between the two clusterings. c) When the mutual information is subtracted from the two clusterings, what is left is the variation of information.

To exactly explain what the variation of information measures, a little background in information theory is needed. First, consider information available in a clustering C, with K clusters. This can be estimated by calculating the Entropy associated with that clustering, H(C)

$$H(C) = -\sum_{k=1}^{K} P(k) \log P(k), \qquad \text{(eqn 3.8.2)}$$

where $P(k)$ is the probability of an element being in cluster $C_k$.

Secondly, consider two clusterings, $C$ and $C'$, as a joint distribution. The probability $P(k,k')$ of a point being in cluster $C_k$ in $C$ and in cluster $C'_{k'}$ in $C'$ is the overlap of these clusters divided by the total number of elements:

$$P(k,k') = \frac{\left|C_k \cap C'_{k'}\right|}{n}. \qquad \text{(eqn 3.8.3)}$$

Thus, the mutual *information* $I(C,C')$ between these two clusterings can be calculated by

$$I(C,C') = \sum_{k=1}^{K}\sum_{k'=1}^{K'} P(k,k') \log \frac{P(k,k')}{P(k)P(k')}. \qquad \text{(eqn 3.8.4)}$$

Finally, the variation of information is basically what is left over after we subtract the mutual information (twice because it is an overlap) from the total entropy of the two clusterings:

$$VI(C,C') = H(C) + H(C') - 2I(C,C'), \qquad \text{(eqn 3.8.5)}$$

Which may be rearranged as

$$VI(C,C') = \underbrace{H(C) - I(C,C')}_{\text{information lost from clustering C'}} + \underbrace{H(C') - I(C,C')}_{\text{information gained from clustering C'}}, \qquad \text{(eqn 3.8.6)}$$

from which it may be seen that the variance of information represents the information lost and gained when picking clustering $C'$ over clustering $C$.

Code for the Variation of Information was written and implemented in R for the framework.

## 3.9 Reporting/Final Clusterings

After the metrics have been calculated, the program combines the output of both the within-method metrics and between-method metrics and produces a report. Parameters that have been used in the clustering methods as well as the clustering results are also available.

Within-method metrics are reported in a tabular format, which can be seen in table 3.9.1 below. The rows list the various methods, and the columns the values of each metric. The first column reports the number of clusters. This is important to report as hierarchical methods may not be able to cut the clusters into a specified number. The second column reported is the homogeneity, then separation, and average silhouette width.

| | n-clusters | homogeneity | separation | silhouette |
|---|---|---|---|---|
| **UPGMAEUC** | 5 | 91.121 | 4.9606 | 0.4828648 |
| **UPGMACOR** | 5 | 90.612 | 4.945 | 0.4823129 |
| **DIANAEUC** | 5 | 91.158 | 5.0078 | 0.4835716 |
| **DIANACOR** | 5 | 90.094 | 4.7385 | 0.4052473 |
| **SOM1** | 5 | 104.87 | 3.6936 | 0.2206755 |
| **KMEANS1** | 5 | 88.16 | 4.3722 | 0.4530941 |

**Table 3.9.1.** Sample within-method table. Rows represent different clustering methods. Columns represent different metrics. In order from left to right: number of clusters, homogeneity (within cluster variation), separation (average distance between cluster centers), and silhouette (how well defined the clusters are).

Between-method metrics were initially reported in an (n-1) x (n-1) matrix format for each metric, where n is the number of methods studied. This representation is pictured in figure 3.9.1a. This format is symmetrical about the diagonal and the value of the matrix at [i,j] corresponds to the value of the metric comparing method i and method j. However, it soon became apparent that this is a difficult format to read. Thus, a dendrogram that corresponds to the "metaclustering" of the clusterings was implemented. Basically, the metaclustering corresponds to an average-link clustering of the matrix. For the Jaccard Index, since 1.0 signifies agreement, it was necessary to use 1-Jaccard in the metaclustering. Dendrograms represent the between-method metrics is an easy-to-grasp visual format that allows the user to group methods visually, as can be seen in figure 3.9.1b.

| | UPGMAEUC | UPGMACOR | DIANAEUC | DIANACOR | SOM |
|---|---|---|---|---|---|
| **UPGMACOR** | 0.6631373 | | | | |
| **DIANAEUC** | 0.6631124 | 0.9841910 | | | |
| **DIANACOR** | 0.6631124 | 0.9920323 | 0.9841905 | | |
| **SOM1** | 0.7401632 | 0.4980161 | 0.4978157 | 0.4978157 | |
| **KMEANS1** | 0.6631124 | 0.9841910 | 1.0000000 | 0.9841905 | 0.4978157 |

(a)

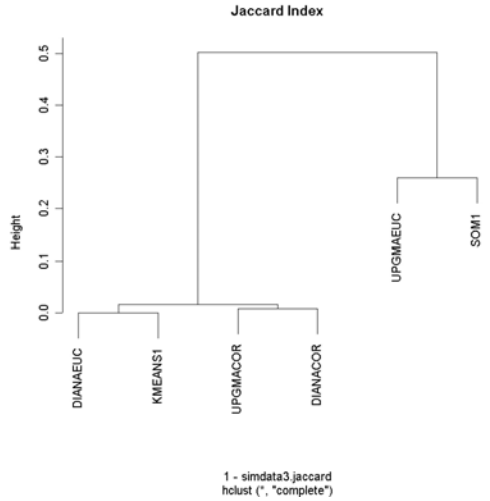



**Figure 3.9.1** Two different representations of the Jaccard Index.
a) Shows the Jaccard Index in a tabular format.
b) Dendrogram "meta-clustering" format. Note

that the metaclustering uses 1-jaccard.

The program also returns parameters as well as the actual cluster results for further study. Clustering results are returned as a numeric vector of integer number that correspond to the clustering assignments of the inputted expression matrix.

# *Chapter 4: Validation of Tool*

## 4.1 Why Validate?

It is one thing to build a tool. It is another thing altogether to evaluate its usefulness to others. Thus, it is necessary to validate the tool on the appropriate microarray data. We will discuss validating the tool on two different kinds of data. The first kind of dataset are three simulated datasets. The importance of validating on simulated data are that the correct clusterings are known for the simulated data. Thus, accuracy rates can be assessed for each method, and we can trace exactly what each clustering method is doing. The data will be evaluated in a fashion similar to the first use case, that of the informatics researcher wishing to compare the performance of the various clustering methods on a dataset with known geometry.

The second kind of dataset that it is important to validate this tool on is real-world microarray data. Even when perturbed with lots of noise, simulated data represents an ideal state for the data. Real-world data is much messier, possibly containing noisy expression profiles that can throw off clustering methods. In the section following, I will discuss the results of running the tool on a smaller subset of the yeast cell-cycle data that Eisen initially used to show the viability of average link clustering as a useful clustering method. The tool will be used in a similar fashion to the second use case, that of the biologist wishing to discover how consistent the clusterings are with an unknown method.

## 4.2 Methods Studied

The six methods studied in the validation are given below in table 4.2.1. These methods were previously discussed in Chapter 2. All six methods were given the correct geometry, and so could be considered on equal footing. For the hierarchical clusterings, R allows for the specification of the number of clusters k cut from the dendrogram.

| ID | Name | Distance | Type |
|---|---|---|---|
| **UPGMAEUC** | UPGMA | Euclidean | Hierarchical – Agglomerative |
| **UPGMACOR** | UPGMA | Correlation | Hierarchical – Agglomerative |
| **DIANAEUC** | Diana | Euclidean | Hierarchical – Divisive |
| **DIANACOR** | Diana | Correlation | Hierarchical – Divisive |
| **KMEANS** | K-means | Euclidean | Partitional |
| **SOM** | Self-organizing maps | Euclidean | Partitional |

**Table 4.2.1**. List of methods studied.

## 4.3 Generation of Simulated Data

The purpose of this simulation study was to provide an initial evaluation of this tool. This should not be considered a traditional simulation study but rather an attempt to create a benchmark data set for initial tool evaluation. One dataset consist of 1000 gene profiles total. These 1000 profiles are divided into four populations of 250 genes each. Each population of 250 genes is derived from a different expression profile pattern, which can be seen in figure 4.3.1. The four different kinds of expression profiles are an upward pattern, a downward pattern, an up-then-down pattern, and a down-then-up pattern. The data was assumed to be log2-transformed, so that a difference in expression level of 1.0 corresponded to a two-fold difference in expression of the raw data.

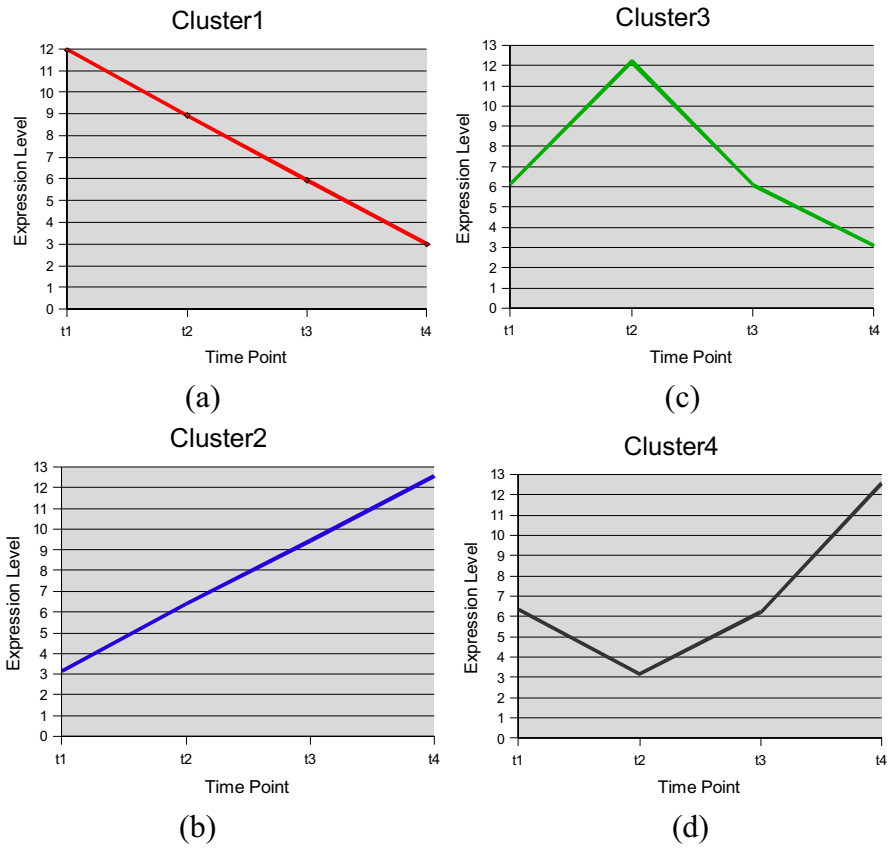



**Figure 4.3.1**. The four gene expression profiles used in the simulation of data.
a) down, b) up, c) up-then-down, and d) down-then-up.

Piecewise linear functions which correspond to each profile were constructed. A random number x from a Gaussian distribution with mean 3.0 and whose standard deviation was known was inputted into one of the piecewise linear functions to generate a gene expression profile. For example, if the random number x was 1.2 and the piecewise linear function was [1x 2x 4x 2x], the expression profile calculated would be [1.2 2.4 4.8 2.4].

Once each population of 250 simulated genes was generated, they were shuffled together into the final expression matrix of 1000 genes. This was necessary because some clustering methods may possibly take advantage of the ordering of the genes.

As previously noted, there are many undesirable sources of variation that can obscure the underlying patterns of expression. This situation is simulated by adding a noise component to each time point in an expression profile. A random number from a Gaussian distribution with known standard deviation and mean 0 was added to each value in the expression matrix. A rough estimate of the Signal to Noise ratio was estimated by taking the mean of x and dividing by the standard deviation of the noise distribution.

Thus, there are two parameters that can be altered to generate new data sets. The first is the standard deviation of the random "signal" input x. This corresponds to the amount of absolute variation in gene profiles across a population in our model. The second parameter is the standard deviation of the noise signal.

Three datasets were generated. The first dataset had low amount of variation in values (signal sd = 0.25), and a low amount of noise (S/N = 20). It was expected that the clustering methods would perform the best on this dataset.

The second dataset had a high amount of variation in values (signal sd = 1.0), but a low amount of noise (S/N = 20). Because of the large amounts of absolute variation, it was expected that the correlation-based methods would do better than the Euclidean based methods on this dataset.

The final dataset had a low amount of variation (signal sd = 0.25), and a high amount of noise (S/N = 4). We would expect clustering methods that are potentially sensitive to noise (such as correlation-based methods) to perform poorly on this dataset.

## 4.4 Notes on the Cho Dataset

A small subset of data derived from the Cho yeast cell cycle dataset [21] by Yeung [22] was also run through the evaluation framework. The Cho dataset consists of 17 time points sampled every 10 minutes from yeast cells undergoing mitotic division. The yeast cells were synchronized to start in the late G1 phase by raising their temperature to 37ºC, which halted them in the late G1 phase. Shifting their temperature to 25ºC reinstated the cell cycle. [21] The 17 time points roughly correspond to two full cell cycles, as can be seen in Figure 4.4.1 The Cho dataset was part of Eisen's original dataset for his paper showing the success of clustering. [10] Therefore, we would expect that the clustering methods should pick up relatively similar clustering patterns.

However, it should be noted that the dataset here used is a subset of Eisen's dataset, both in the number of genes clustered and in the number of experimental conditions. Eisen included data from diauxial shifts in yeast, many of which showed large fold-changes in expression. Thus, it is also possible that our results may differ. It is also worth noting that Eisen's clustering method is slightly different from the standard UPGMA method in the way missing values are treated. However, there are no missing values in this dataset, unlike the number of missing values in Eisen's full dataset.

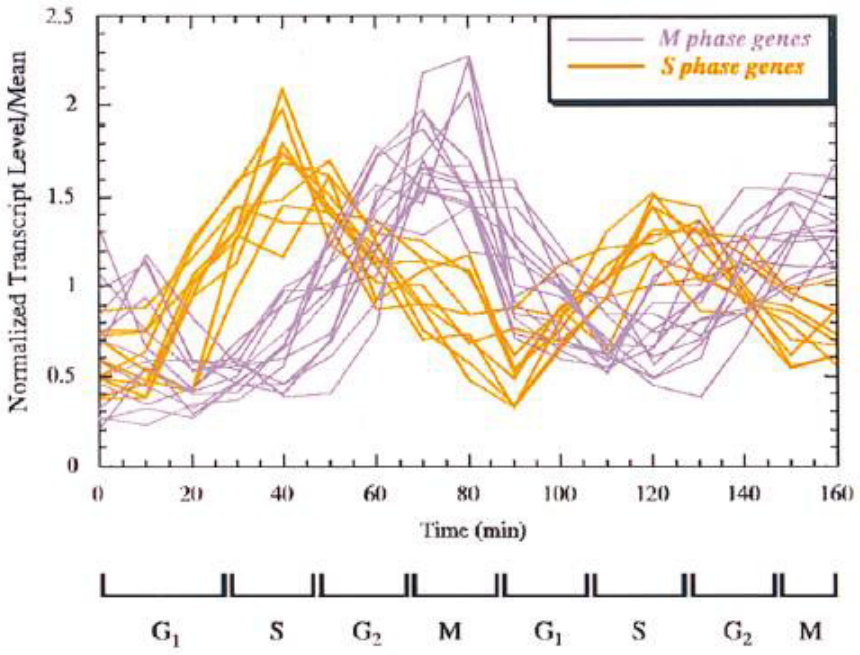



**Figure 4.4.1**. A sampling of genes from the Cho Mitotic Cell-cycle dataset. M-phase genes are those that peak during the M-phase, and S-phase genes correspond to those that peak in the S-phase. Note that the above represents two full cell cycles. Reproduced from Cho, et al. [21]

In Cho's original paper, a subset of 420 genes was reported that had been separated into six functional categories. These categories corresponded to genes with peak expression in the Early G1, Late G1, S, G2, M, phases, and those genes that were expressed in multiple phases. Figure 4.4.1 shows examples of both M-phase and S-phase genes. This subsetting was done largely by visual examination of the genetic data, which was somewhat problematic in our study. Thus, it should be noted the annotations given by Cho were not definitive and it was necessary to obtain the Gene Ontology Molecular Function annotations for each gene to see if the methods were clustering genes by function.

This subset of these original 420 genes was constructed using a filtering process described in Yeung. [22] The first step involved filtering out those genes that were

expressed in multiple phases. Such genes could possibly have a different pattern of expression than those expressed in a single phase and thus could make correlation-based hierarchical methods produce bad clusters, as was seen with the high-variation, low-noise dataset. Those genes containing negative values were also filtered out. This is because without extensive notes about the dataset, negative values are uninterpretable. The remaining genes were then log-transformed and then normalized using the same procedure described in Tamayo. [15]

The Cho dataset has been extensively analyzed with an obvious geometry, and so we expected clustering methods to perform reasonably well on this smaller subset. However, there are some questions as to the validity of this dataset; there have been concerns that two of the timepoints may be suspect. [15, 23] Several papers have noted scaling problems with the time point at 90 minutes, and so remove this point from their analysis. [15, 24] This was not done in our validation.

## 4.5 Analysis of Results

In order to gauge performance of the clustering methods, the true assignments of the genes were compared with assignments from the clustering methods. A confusion matrix was calculated for each method, which compares the clustering label to the true label. The confusion matrix is a useful tool for understanding clustering results because it can show overlap between the method's cluster assignments and the true clustering assignments. This is especially true with the hierarchical clustering methods, where one cluster can contain two or more of the true clusters. Because the cluster assignment labels did not correspond to the true assignments, it was necessary to permute the

columns of the confusion matrix in order to maximize the diagonal. This was done using R code available in the `e1071` library (more information available at http://cran.r-project.org/src/contrib/Descriptions/e1071.html). An accuracy rate for a particular clustering method was calculated by taking the total of the maximized diagonal and dividing by the total number of genes.

Confusion matrices for each simulated dataset can be seen in Appendix B. Reports including accuracy rate are given below for each dataset.

For the Cho dataset, accuracy rates were first assessed using the original functional annotation given by Cho. It became clear that the functional annotations were assigned by visual inspection and so the accuracy rates were not used. In order to aid with biological interpretation, GO Molecular Function annotations were obtained from the appropriate Bioconductor annotation packages (YEAST and GO) for each open reading frame (ORF, defined as an identified stretch of DNA that begins with a start codon and ends with a stop codon, not necessarily tied to any function).

Clusters from each method were initially analyzed as to how many genes contained which phase annotation. It became clear that three clusters had a high degree of overlap, and so the overlap with analogous clusters across methods was calculated. This was done by taking the intersection of the three datasets and comparing the remaining genes by annotation. The overlapping clusters were then analyzed by individual genes description and GO Molecular Function annotation in order to gauge the biological significance of these clusters.

## 4.6 Parameters used on Simulated Data Sets

All three simulated datasets were run through the evaluation framework. The six clustering methods shown in Table 4.2.1 were run on each dataset. K-means was given a maximum iterations parameter of 10,000. R has functions for cutting a dendrogram into a specified number of clusters, and these were used to cut the dendrograms for the hierarchical method into 4 clusters. k-means received an input k of 4, and SOM received an input of a 1x4 map. Cutting the dendrogram was done in order to put all six methods on equal ground. However, it could also be argued that doing so forces the hierarchical methods into unrealistic clusters. A better comparison might be to give k-means and SOM the necessary geometry and to cut the dendrogram by visual inspection of the clusters. This is one potential limitation of this validation and it will become apparent especially in examining the high-variation, low-noise dataset.

## 4.7 Results: Low-noise, low-variation dataset

The true clusterings for the low-noise, low-variation dataset can be seen in figure 4.7.1 below. Overall, this dataset is probably the least realistic simulation of all the datasets. Noise affects the correlation patterns minimally, which is not the case in real life. We would expect that both Euclidean-based and correlation-based datasets to do equally well on this dataset due to the fact that the profiles are very close and well defined in the sense of both correlation and Euclidean distance.

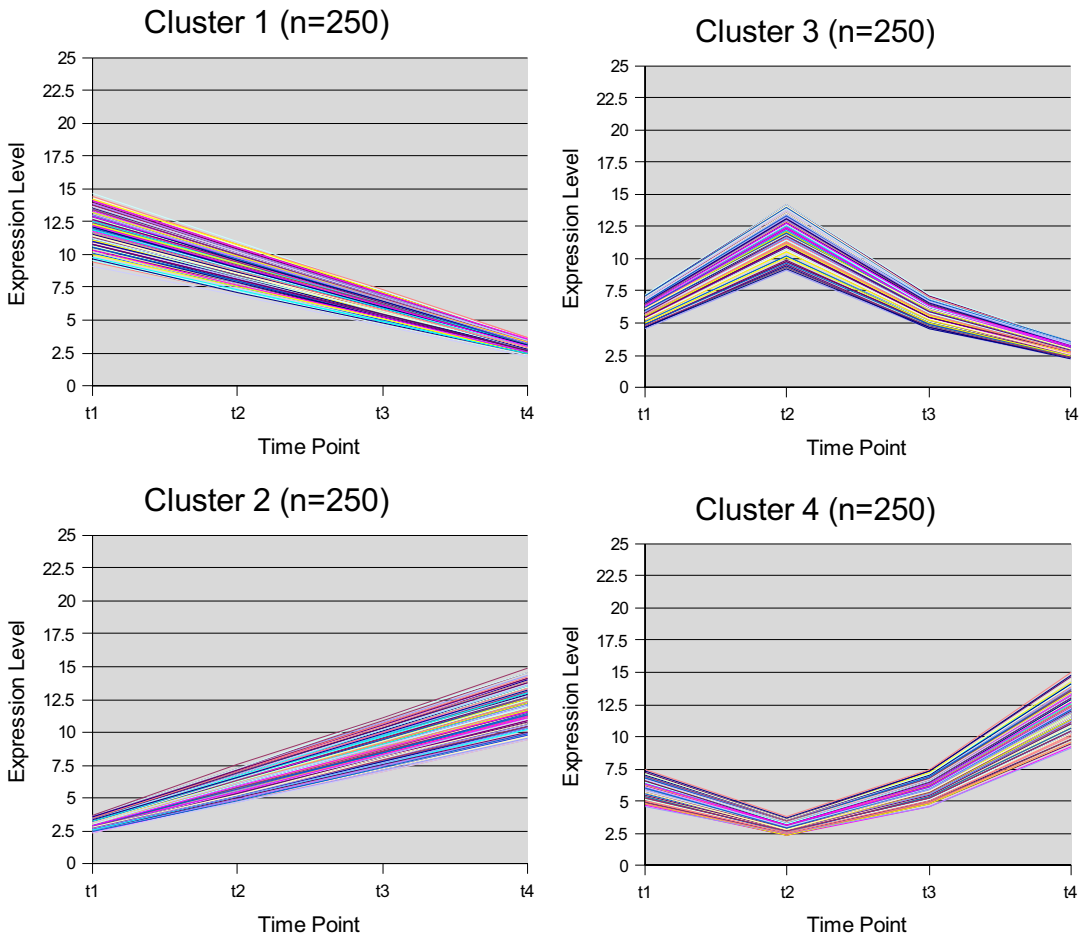



**Figure 4.7.1**. The true clusterings of the low-noise, low-variation dataset.

The performance of the six methods can be seen in table 4.7.1. Overall, the hierarchical methods perform best on this dataset, all having accuracy rates of 100%. The partitional methods do not fare as well, with SOM being only 75% accurate, and K-means even less accurate at 62.9%. Two Euclidean-based methods (UPGMAEUC and DIANAEUC) do very well, while the other two Euclidean-based methods (K-means and SOM) do not. It is necessary to examine the clusters assigned by both k-means and SOM in order to understand their lack of accuracy.

| Method | Accuracy Rate |
|---|---|
| UPGMAEUC | 100.00% |
| UPGMACOR | 100.00% |
| DIANAEUC | 100.00% |
| DIANACOR | 100.00% |
| SOM1X4 | 75.00% |
| KMEANS4 | 62.90% |

**Table 4.7.1.** Performance of the various clustering methods on the low-noise, low-variation dataset.

The cluster assignments of SOM can be seen in figure 4.7.2. (Note that the cluster numbers assigned by each method are arbitrary, so the clusters in this figure and all following are arranged visually to correspond as much as possible with the original clusterings.) As can be seen from the figure, SOM does not even utilize one of the available partitions, instead choosing to cluster the up cluster with the down and then up cluster. Because these two clusters have a relatively closer Euclidean distance than to the other two clusters, this is not an unreasonable result. However, SOM does cluster the other two clusters, the down cluster and the up-then-down cluster perfectly.

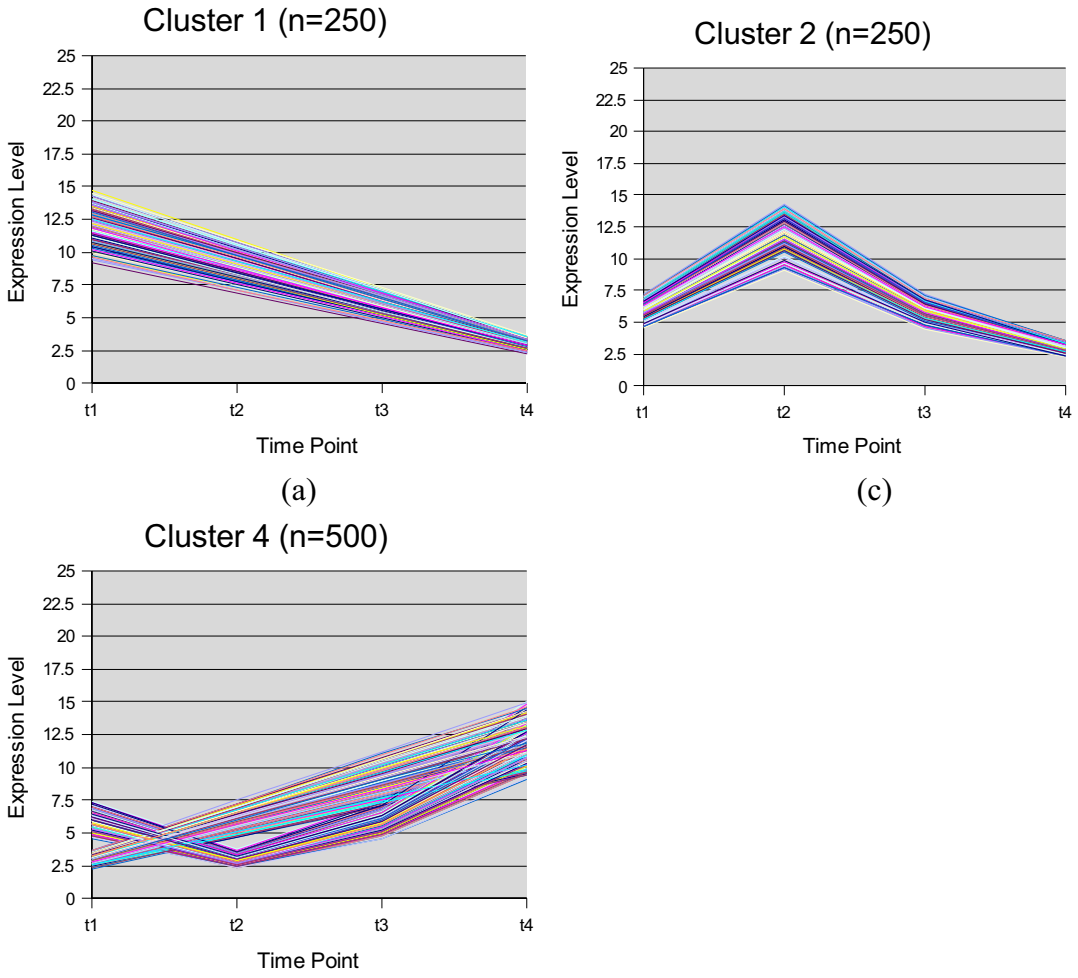



**Figure 4.7.2**. Clusterings for SOM on low-noise, low-variation data. In figure 4.7.2 b), SOM has combined both the up and the down-then up cluster, due to their close Euclidean distance. Note that cluster numbers assigned by methods are arbitrary, so the clusters are arranged visually to correspond as much as possible with the original clusterings.

The clusterings for k-means can be seen in figure 4.7.3. K-means partitions this data set similarly to SOM in that it clusters the up and the down-then-up into a single cluster. However, it also separates the up-then-down cluster into two separate clusters, even though their overall Euclidean distances are rather close. Thus, we expect that the average silhouette value for this clustering should be lower than the other clusterings.

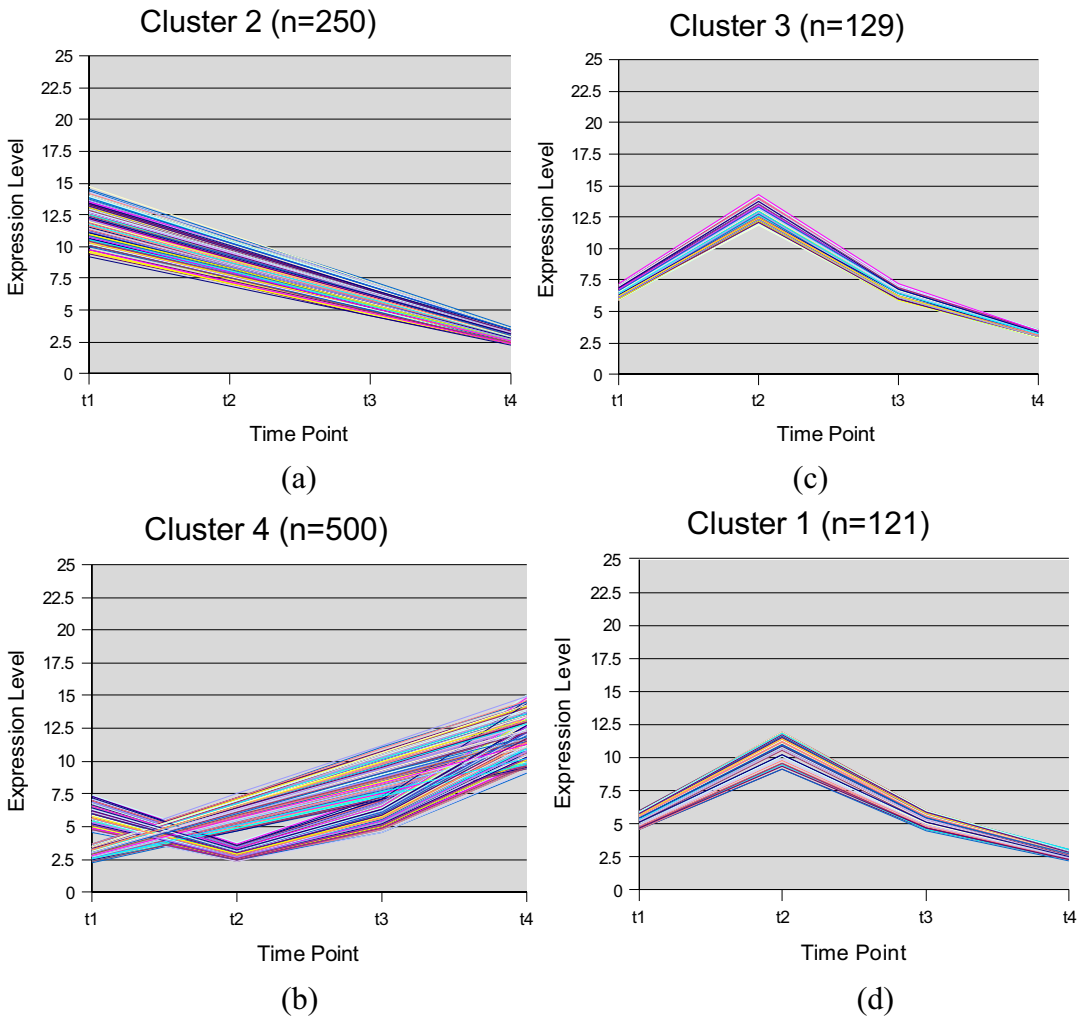



**Figure 4.7.3**. Performance of k-means on low-variation, low-noise dataset. Note that in figure 4.7.3 b) k-means combines the up cluster with the down-then-up cluster. In figure 4.7.3 c) and d), k-means has divided the up-then-down cluster into two distinct clusters by range.

Now we will examine the within-metrics and see what they add to the analysis of the clustering methods. Table 4.7.2 shows the within-method metrics in the report format. Note that the four best-performing methods have identical homogeneity, separation, and silhouette values. Also note that k-means has an overall lower silhouette value than the other five methods. This suggests that the tool should be used to look for aggregations of agreeing values, rather than using any of the metrics as an absolute

measure. The NA value for SOM's homogeneity has to do with the fact that the membership of one of the slots in the 1x4 map is zero.

| | n-clusters | homogeneity | separation | silhouette | %match |
|---|---|---|---|---|---|
| **UPGMAEUC** | 4 | 109.1011 | 10.498687 | 0.9996794 | 100% |
| **UPGMACOR** | 4 | 109.1011 | 10.498687 | 0.9996794 | 100% |
| **DIANAEUC** | 4 | 109.1011 | 10.498687 | 0.9996794 | 100% |
| **DIANACOR** | 4 | 109.1011 | 10.498687 | 0.9996794 | 100% |
| **SOM1** | 4 | 198.1462 | NA | 0.9559742 | 75.00% |
| **KMEANS1** | 4 | 189.0392 | 9.797287 | 0.7081339 | 57.60% |

**Figure 4.7.2.** Within-method metrics on low-noise, low-variation dataset. Note that the best performing methods, UPGMAEUC, UPGMACOR, DIANAEUC, and DIANACOR have identical values.

The metaclustering dendrograms for both between-method metrics can be seen below in figure 4.7.4. In general, the between-method metrics group the methods similarly as the within-method metrics. Note that both metrics group the best performing methods as a single group, separate from SOM and k-means. Note that the height of the joins for UPGMAEUC, UPGMACOR, DIANAEUC, and DIANACOR is zero, indicating that these clusterings are identical according to both methods.

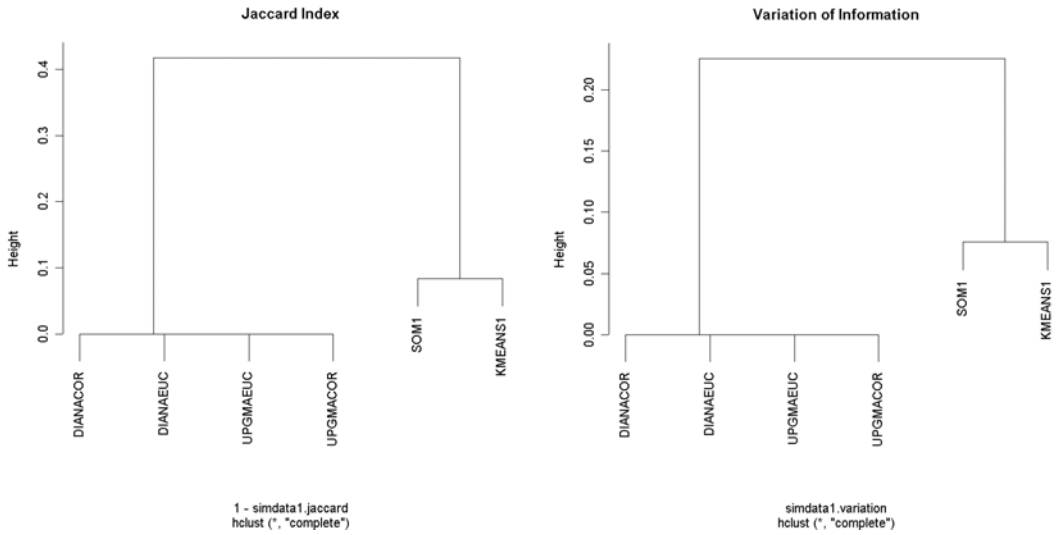



**Figure 4.7.4.** Metaclustering dendrograms for between-method metrics. Note that both the Jaccard Index and Variation of Information group the best performing methods into one group separate from the other methods.

There are a few lessons we can glean from running the tool on the low-variation, low-noise dataset. The first is that the tool seems to be good at showing consensus between clusterings that are identical. The second lesson is that Homogeneity and Separation should not be used as an absolute quantity – they are most helpful in pointing groupings of similar values. The third is that there is general agreement with the between-method metrics and the within-method metrics. Both show the same consensus information about the clusterings.

## 4.8 Results: High-variation, Low-Noise Dataset

The true clusterings for the high-variation, low-noise dataset can be seen in figure 4.8.1. below. Because of the large variation in gene profiles, we expect that Euclidean-based methods should do poorly on this dataset.

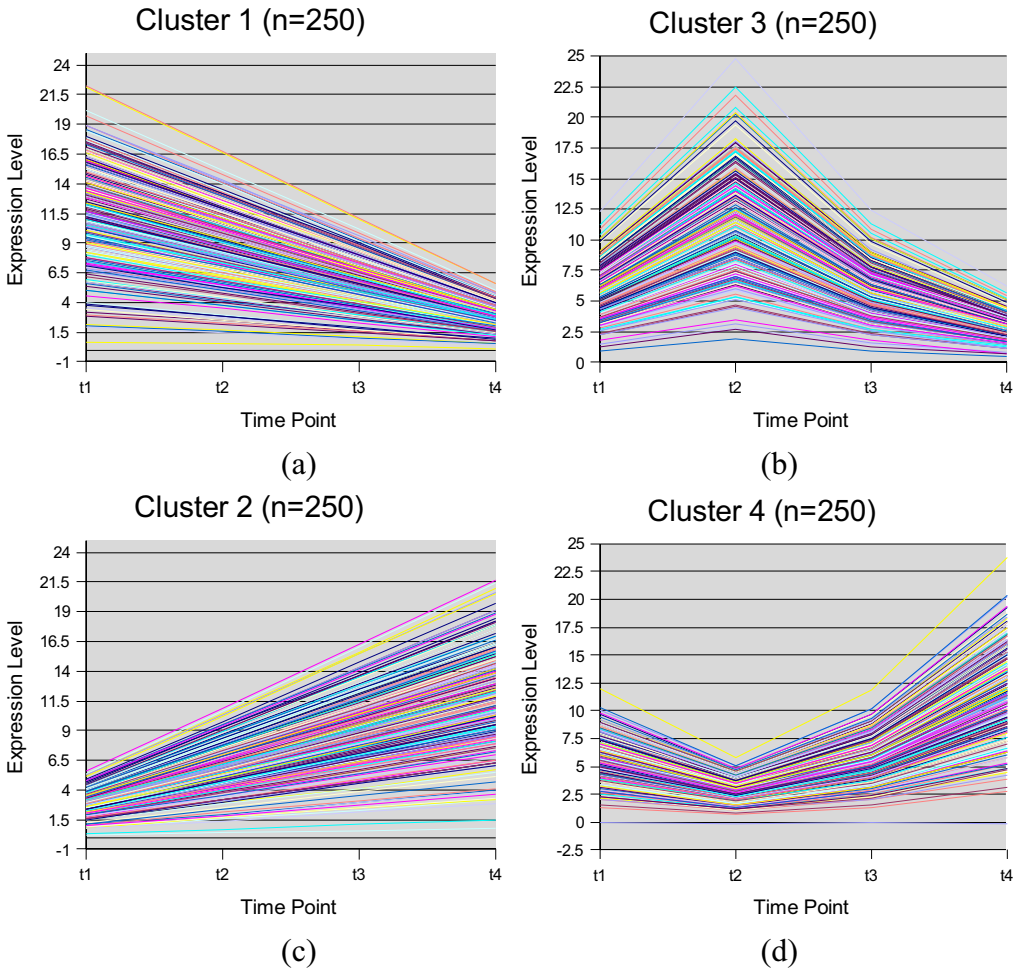



**Figure 4.8.1**. True clusterings for the high-variation, low-noise dataset.

The performance of the six methods on the high variation, low noise dataset can be seen in table 4.8.1. below. In summary, none of the methods completely clusters the data into the correct partitions. The two correlation-based hierarchical methods do the

best, but are still only 75% accurate. It is instructive to examine the clusterings of DIANACOR and SOM.

| Method | Accuracy Rate |
|---|---|
| UPGMAEUC | 46.00% |
| UPGMACOR | 75.10% |
| DIANAEUC | 61.50% |
| DIANACOR | 75.10% |
| SOM1X4 | 53.40% |
| KMEANS4 | 57.60% |

**Table 4.8.1.** Performance of Clustering Methods on High-variation, low-noise dataset.

The clusters assigned by DIANACOR can be seen in figure 4.8.2. DIANACOR clusters the up and the down-then-up cluster into the same cluster, much like what SOM did for the low-variation, low-noise dataset. The problem turns out to be that one noisy gene impacts the cluster, forcing a new, high-level, single cluster, as can be seen in figure 4.8.2.

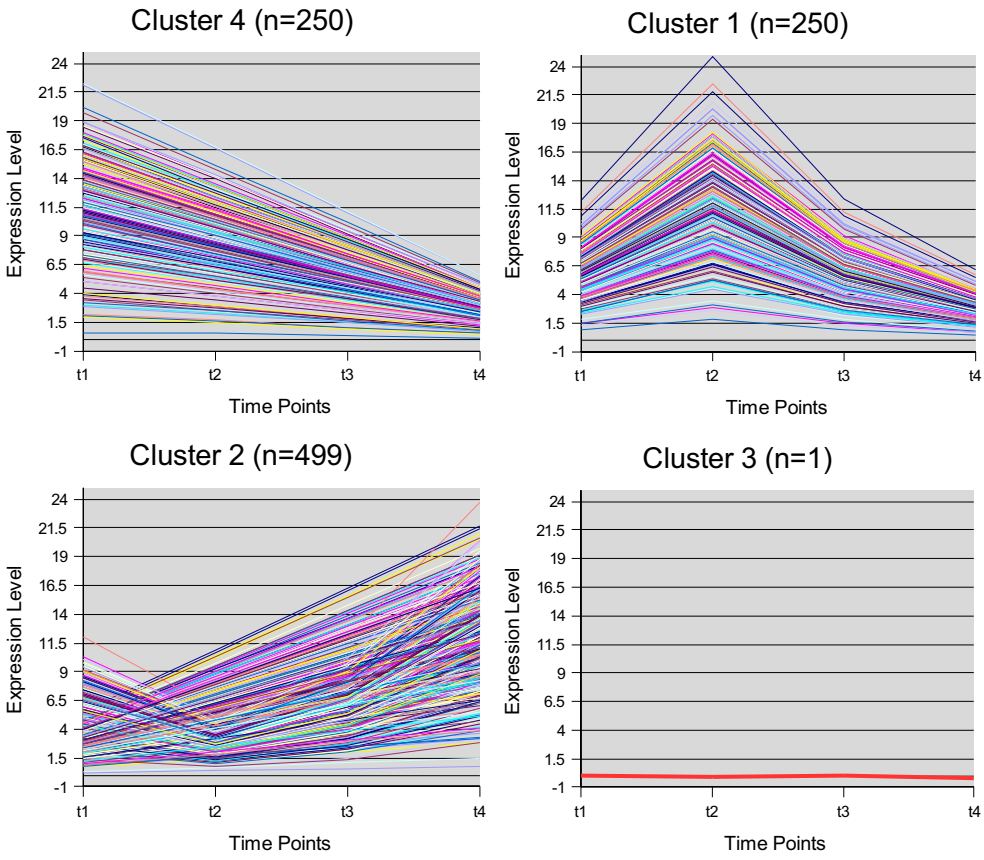



**Figure 4.8.2.** Performance of Correlation-based hierarchical clusterings on high-variation, low-noise data.

In figure 4.8.3 a zoom in of the single gene of cluster 3 can be seen. This is a gene that was a member of the down-then-up cluster. Even though it is essentially a gene with no large changes in expression, the correlation-based method picks up that this gene has a different correlational structure than all of the other population. This clustering emphasizes the importance of filtering out noisy genes that have no significant changes in expression across samples, as correlation-based methods can be misled by such genes.

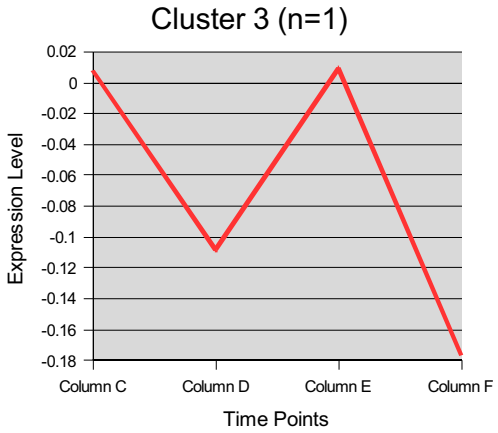



**Figure 4.8.3**. Zoom in of Cluster 3. Note that this noisy gene has a different correlational structure than all of the other genes. Note that expression scale is very small – essentially this expression profile is noise.

Part of the problem of why the correlation-based hierarchical methods are susceptible to noise has to do with the fact that the geometry (the number of clusters) is specified by the program. Thus, we are robbing these methods of one of their potential strengths – these methods do not need a prespecified geometry. This effect can be seen from the UPGMACOR dendrogram in figure 4.8.4. The single noisy gene is joined to the rightmost cluster at a height larger than the two leftmost (and larger) clusters. Thus, specifying the algorithm to cut the tree into four clusters results in the two leftmost clusters becoming combined, leaving the singleton as a noisy gene. This fact does point towards a limitation of our study.

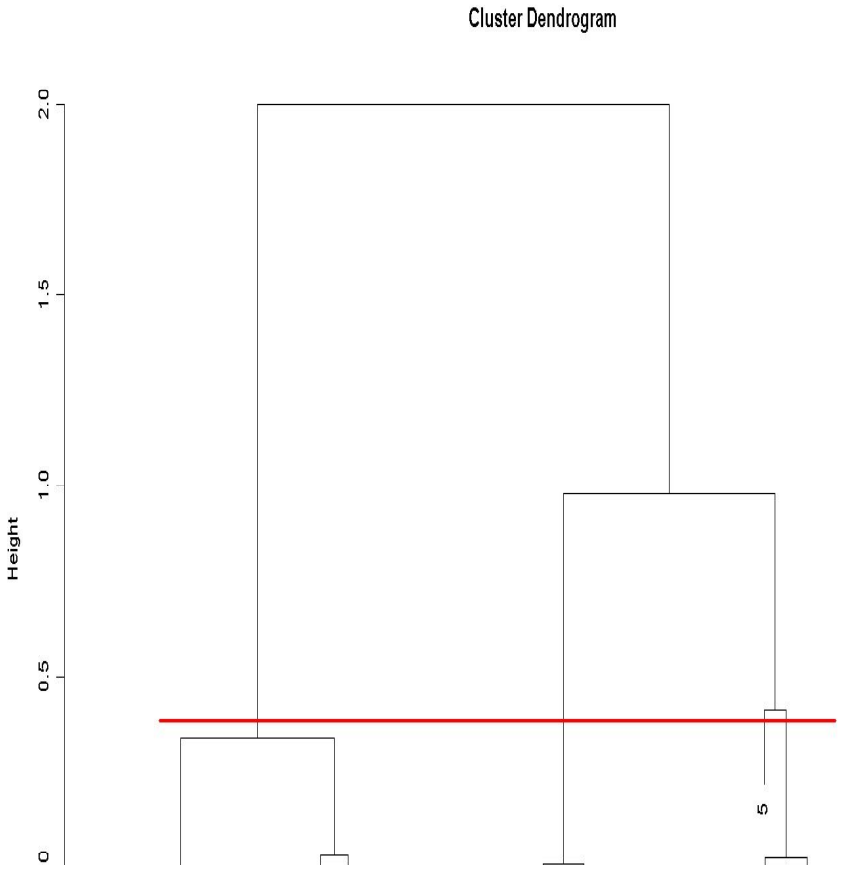



**Figure 4.8.4.** Zoom in of clustering dendrogram for UPGMACOR. Note that the single noisy gene is joined with the rightmost cluster higher than the two leftmost clusters.

The clusterings assigned by SOM can be seen in figure 4.8.5. As can be seen, the partitioning of SOM is strictly Euclidean, with SOM confusing the up-then-down cluster and down cluster, as well as the up cluster and the down-then-up cluster. Because these clusters are not well-defined, we expect that the silhouette value for this clustering to be very low.

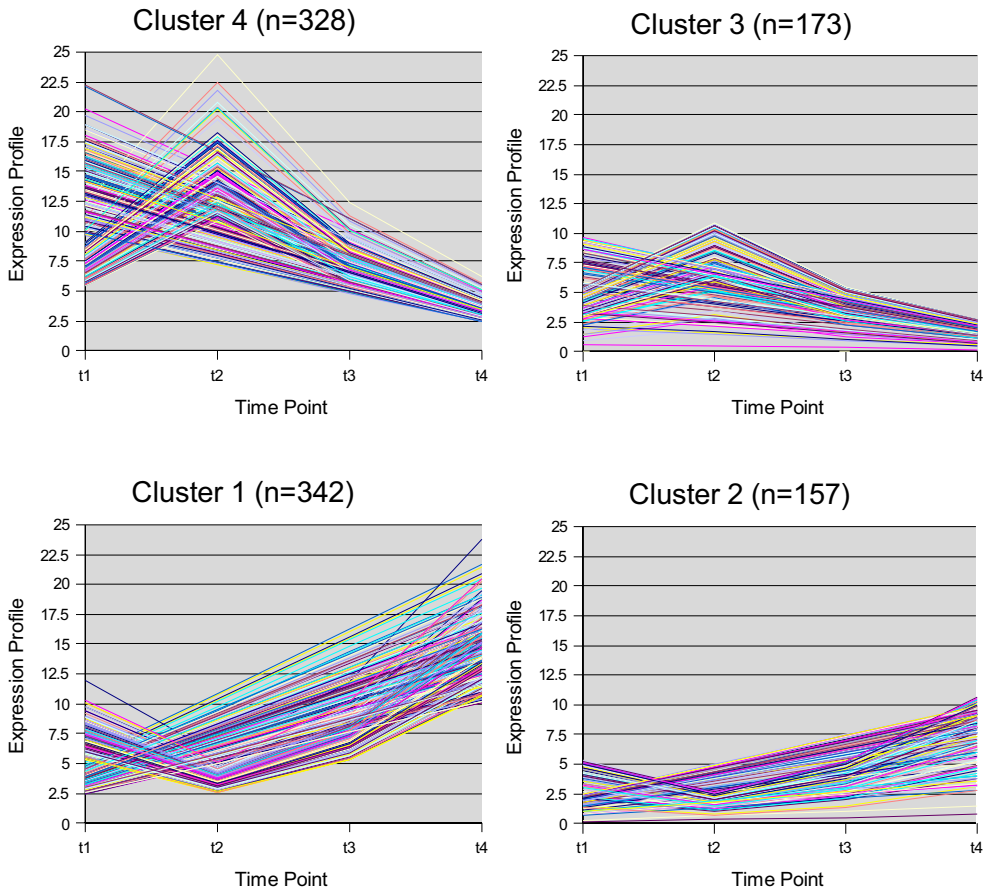



**Figure 4.8.5.** SOM clustering of high-variation, low-noise dataset.

At this point, it is instructive to examine the within-method metrics in table 4.8.2. As before, the two best-performing methods have identical homogeneities, separations, and average silhouette widths. Also note that SOM has a low silhouette value.

| | n-clusters | homogeneity | separation | silhouette | %match |
|---|---|---|---|---|---|
| **UPGMAEUC** | 4 | 566.7264 | 13.83335 | -0.5155965 | 46% |
| **UPGMACOR** | 4 | 454.754 | 13.95272 | 0.95298656 | 75.10% |
| **DIANAEUC** | 4 | 397.8658 | 10.3832 | 0.68264895 | 61.50% |
| **DIANACOR** | 4 | 454.754 | 13.95272 | 0.95298656 | 75.10% |
| **SOM1** | 4 | 407.9042 | 11.64011 | 0.00908861 | 53.40% |
| **KMEANS1** | 4 | 397.4189 | 10.85499 | 0.05120928 | 57.60% |

**Table 4.8.2.** Within-method metrics for high-variation, low-noise dataset. Note that the two best performing methods (UPGMACOR and DIANACOR) have identical homogeneities, separations, and average silhouette widths.

The between-method metrics agree with the within-method metrics. Note that the distance between UPGMACOR and DIANACOR is zero for both metrics, and they are separated the furthest from the other methods.

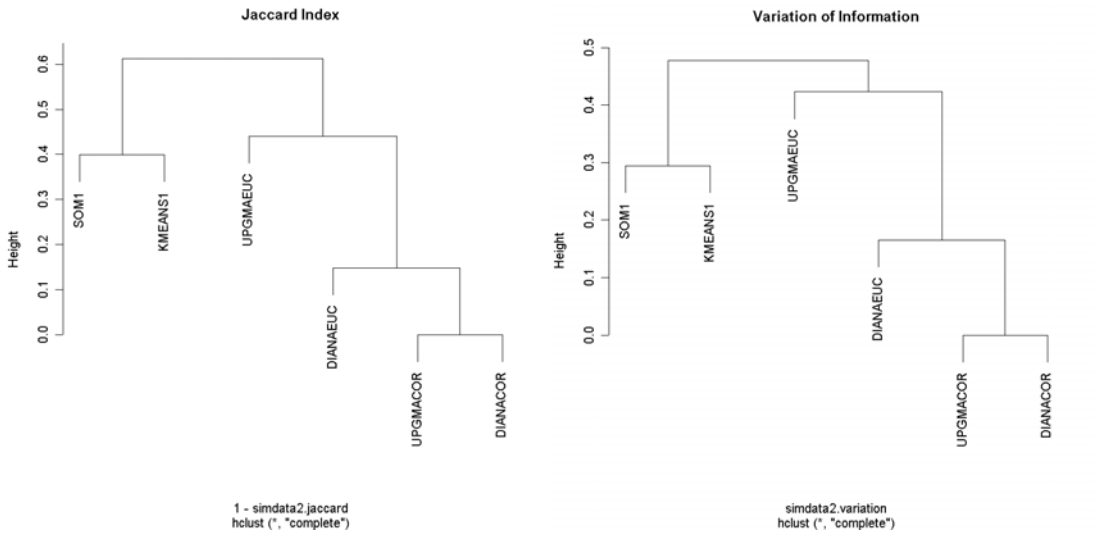



**Figure 4.8.6.** Metaclustering dendrograms of between-method metrics on high-variation, low-noise dataset.

There are some interesting lessons to be gleaned from running the tool on the high-variation, low-noise dataset. The first is that correlation-based methods are especially sensitive to noisy genes, and the importance of filtering out genes based on significant changes across samples is very important. The second lesson is that seeing the performance of the various methods on the dataset suggests that the dataset should be plotted and descriptives of the data should be examined before any blind clustering occurs.

## 4.9 Results: Low-variation, High-Noise dataset

The true clusterings for the low-variation, high-noise dataset can be seen in figure 4.9.1. We would expect that correlation-based methods should perform worse than Euclidean-based methods on this dataset. This is also the dataset that seems to be closest to the Cho dataset and thus probably the most realistic of the simulated datasets.

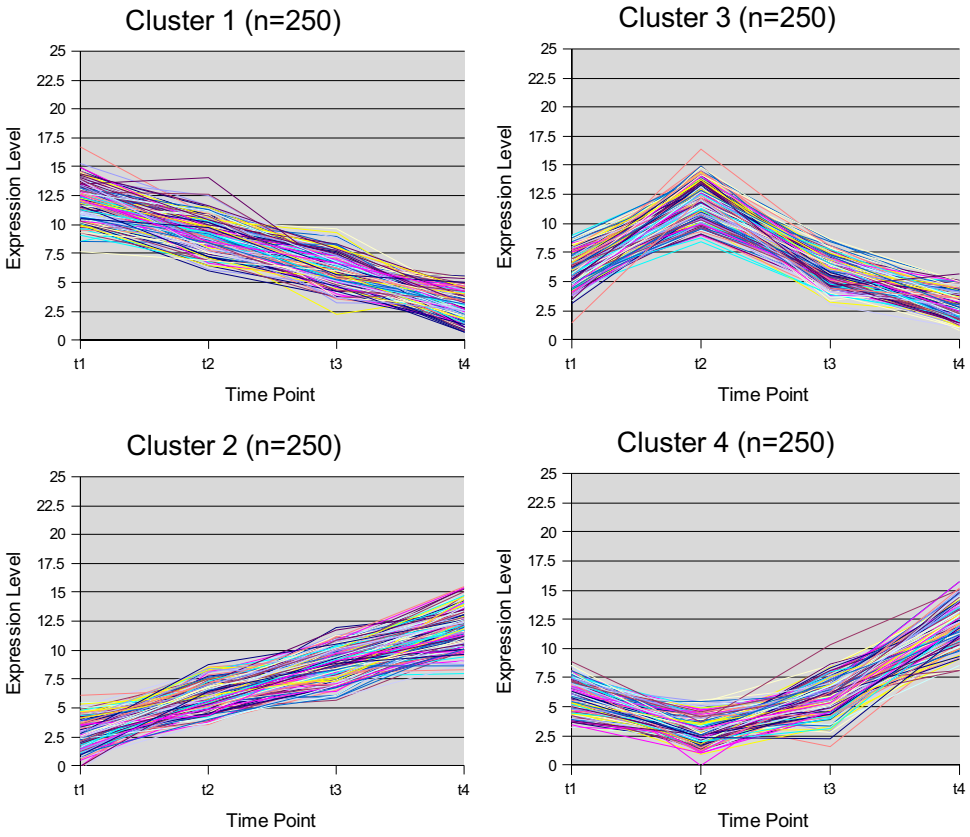



**Figure 4.9.1.** True clusterings of low-variation, high-noise dataset.

The performance of the various methods can be seen in Table 4.9.1. Unexpectedly, correlation-based methods do rather well on this dataset. Two Euclidean-based methods do well, KMEANS and DIANAEUC, but the other two methods, UPGMAEUC and SOM, do not. SOM behaves similarly as in the other two datasets. However, it is instructive to examine what is going on with UPGMAEUC.

| Method | Accuracy Rate |
|---|---|
| UPGMAEUC | 74.90% |
| UPGMACOR | 99.60% |
| DIANAEUC | 99.60% |
| DIANACOR | 99.80% |
| SOM1X4 | 66.10% |
| KMEANS4 | 99.60% |

**Table 4.9.1**. Performance of clustering methods on low-variation, high-noise dataset.

The clusters assigned by UPGMAEUC can be seen in figure 4.9.2. Oddly enough, UPGMAEUC assigns a gene from the down-then-up cluster into a cluster of its own. The range is slightly higher than the other members of its population. Agglomerative methods are sometimes known for undependable high-level clusters, and this appears to be of those cases. [1]

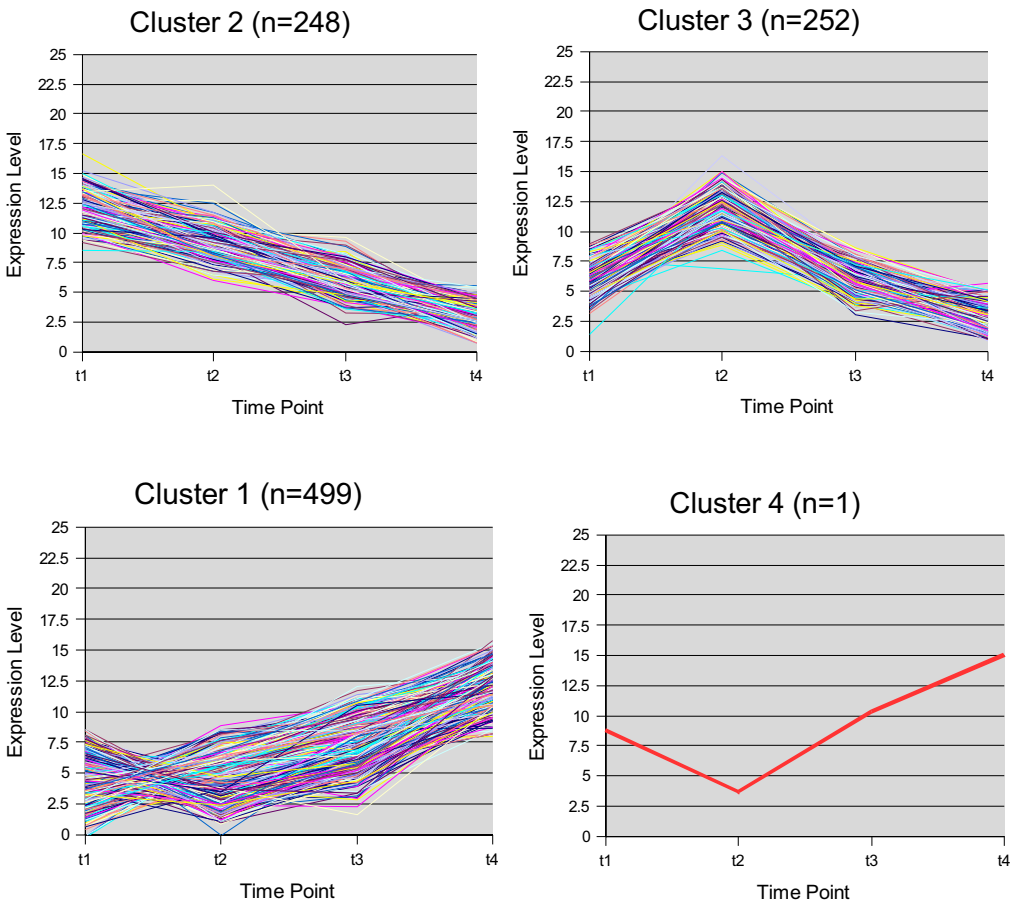



**Figure 4.9.2.** UPGMAEUC clusterings on low-variation, high-noise dataset.

Now we see what the tool has to say about these clusterings. Again, the four best-performing methods have similar metrics.

| | n-clusters | homogeneity | separation | silhouette | %match |
|---|---|---|---|---|---|
| **UPGMAEUC** | 4 | 281.1023 | 11.76735 | 0.3979998 | 74.90% |
| **UPGMACOR** | 4 | 219.8808 | 10.28038 | 0.8711627 | 99.60% |
| **DIANAEUC** | 4 | 219.6732 | 10.27096 | 0.869455 | 99.60% |
| **DIANACOR** | 4 | 219.805 | 10.27004 | 0.872493 | 99.80% |
| **SOM1** | 4 | 316.6078 | 11.38122 | 0.3621547 | 66.10% |
| **KMEANS1** | 4 | 219.6732 | 10.27243 | 0.869455 | 99.60% |

**Table 4.9.2.** Within-cluster metrics for low-variation, high-noise dataset.

The between-method metrics show agreement with the within-method metrics, grouping the four best performing methods apart from UPGMAEUC and SOM. Note

that the metrics are sufficiently sensitive to pick up even the small differences between the four best-performing clusterings.

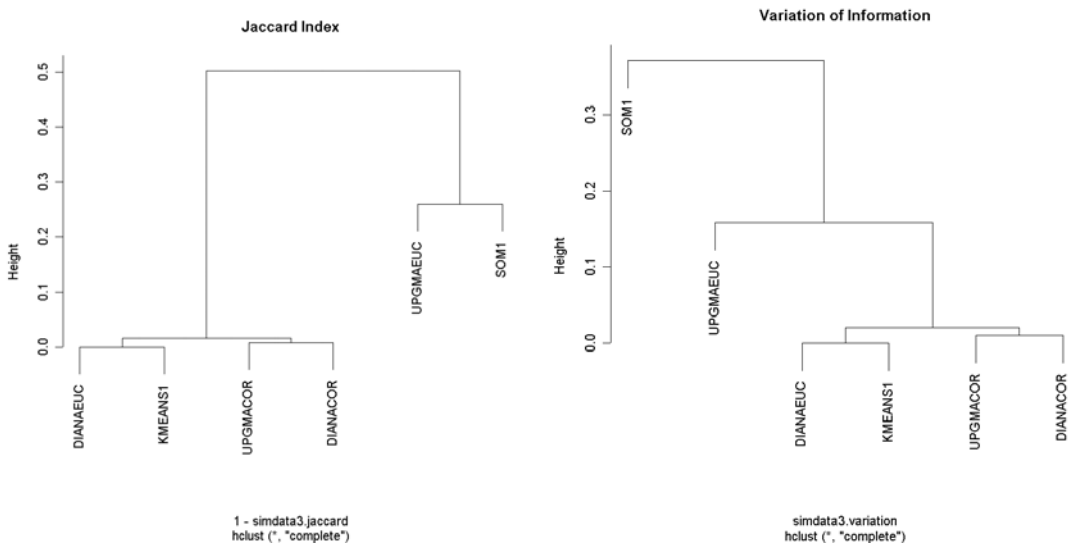



**Figure 4.9.3.** Metaclustering dendrograms for low-variation, high-noise dataset.

There is a lesson that can be gleaned from the low-variation, high-noise dataset. Correlation-based methods are able to tolerate a certain amount of noise before their performance suffers. However, it can be argued that not enough noise was added to this artificial dataset. Because there are only four time points, large changes in value were required to build any correlational structure in the data. Because there is a high amount of correlational structure in these four populations, it was difficult to add enough noise to completely destroy the correlational structures of each population. The correlational structure of each population was still largely intact after the noise component was added.

## 4.10 Results: Cho dataset

All six methods were run on the normalized form of the Cho dataset. K-means was given a parameter of 10,000 maximum iterations. Again, to put all methods on equal ground, 5 clusters were specified for each method. Hierarchical methods were instructed to cut the dendrogram into 5 clusters, SOM used a 1 x 5 map as input, and, kmeans was given a k of 5.

The genes grouped by functional annotation are shown in figure 4.10.1. Note that the annotations were done largely by visual inspection of each expression profile. Thus, we would not expect the clustering methods to reproduce these groupings perfectly. We did use the functional annotations as a quick way to compare across clusterings, but not to validate these clusterings. In order to do, it is necessary to examine the GO molecular function annotations.

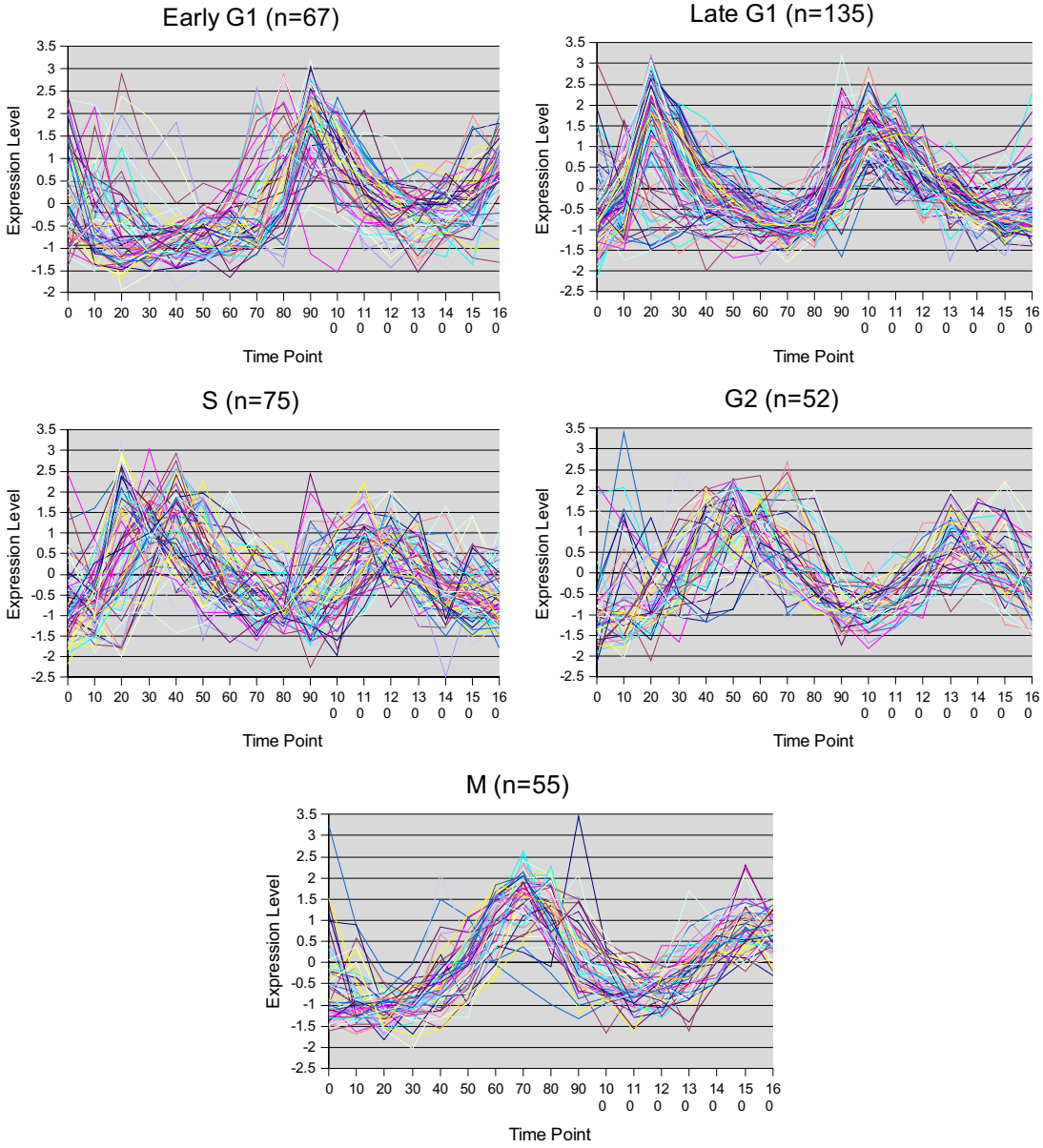



**Figure 4.10.1.** Genes grouped by the functional annotation given in Cho, et al. [21]

Upon examining these groupings, it is apparent that only three of the function groupings seem to be clusters, those of Early G1, Late G1, and M. S and G2 are not very well characterized patterns of expression.

It is a useful exercise to approach this dataset like a biologist searching for genes with similar patterns of expression. Examining the within-method metrics in table 4.10.1,

we note that there are three methods that have highly similar metrics: UPGMAEUC, UPGMACOR, and DIANAEUC. We have seen that in the simulated data that the best cases have very similar metrics. Hence this suggests that there is consensus between these three methods.

| | n-clusters | homogeneity | separation | silhouette |
|---|---|---|---|---|
| **UPGMAEUC** | 5 | 91.121 | 4.9606 | 0.4828648 |
| **UPGMACOR** | 5 | 90.612 | 4.945 | 0.4823129 |
| **DIANAEUC** | 5 | 91.158 | 5.0078 | 0.4835716 |
| **DIANACOR** | 5 | 90.094 | 4.7385 | 0.4052473 |
| **SOM1** | 5 | 104.87 | 3.6936 | 0.2206755 |
| **KMEANS1** | 5 | 88.16 | 4.3722 | 0.4530941 |

**Table 4.10.1.** Within-method metrics for the normalized Cho dataset.

One question that could be asked is why the silhouette value is so low for all three of these clusterings. Once we examine one of these clusterings, it will become apparent that there is at least one junk cluster whose profiles of expression differ from the other profiles enough to be put in their own cluster. However, we will see that there is very little correlation within this junk cluster.

Again, the report shows that homogeneity and separation should not be used as absolute magnitudes when judging a clustering. Homogeneity and Separation are metrics that give the user a feel for the overall shape and spread of the clusterings. As the true clusters can be highly spread out, the absolute magnitude of either quantity is not a useful judge of the clustering. Rather, these two methods should be used to look for consensus among the clusterings.

What do the between-method metrics say about these methods? As can be seen in figure 4.10.2 below, two between-method metrics group the methods very similarly. Both metrics group the SOM clustering as being much more different than the other five clusterings. UPGMACOR is consistently grouped with KMEANS in both, as are the two

Euclidean-based hierarchical methods. However, the placement of DIANACOR is slightly different. Variation of Information groups DIANACOR with UPGMAEUC and DIANAEUC. Jaccard groups DIANACOR with UPGMACOR and KMEANS. However, looking at the height of the joins of DIANACOR relative to these two groupings, either clustering is plausible.

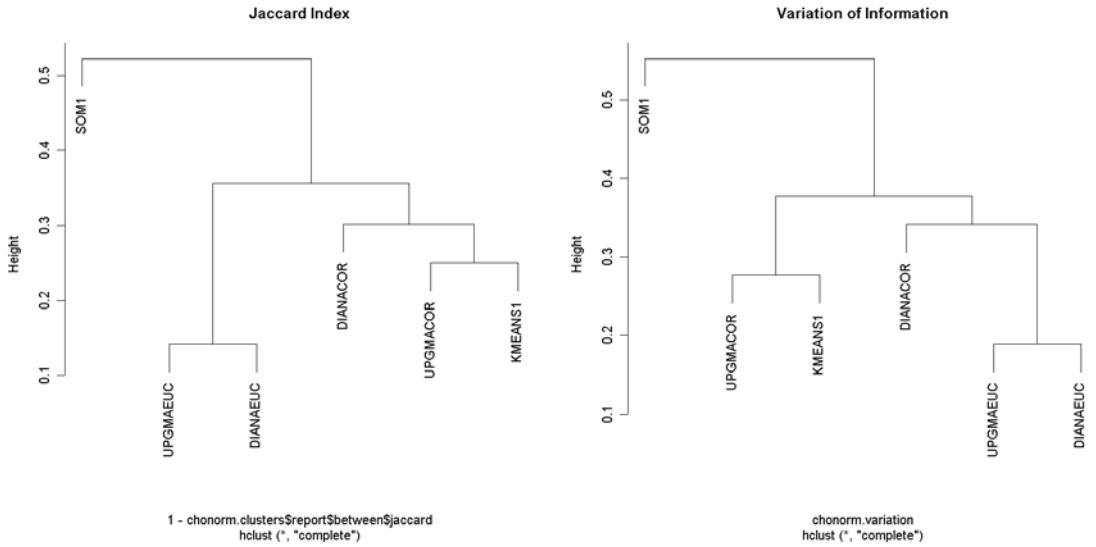



**Figure 4.10.2**. Metaclustering dendrograms of between-method metrics.
Note that the trees are generally the same, except for the relation of DIANACOR.

In order to see more of the story behind the clustering methods, it is necessary to view a clustering and to examine the GO functional annotations that make up a cluster. UPGMACOR is a good candidate to start with, because it is the same method that Eisen used to show the usefulness of clustering as a method. Below, in figure 4.10.3, the clusters of UPGMACOR can be seen. Note that cluster 5 is essentially a "junk" cluster and that the profiles contained in it correlate poorly. This is one of the reasons the silhouette value for this clustering is somewhat low.

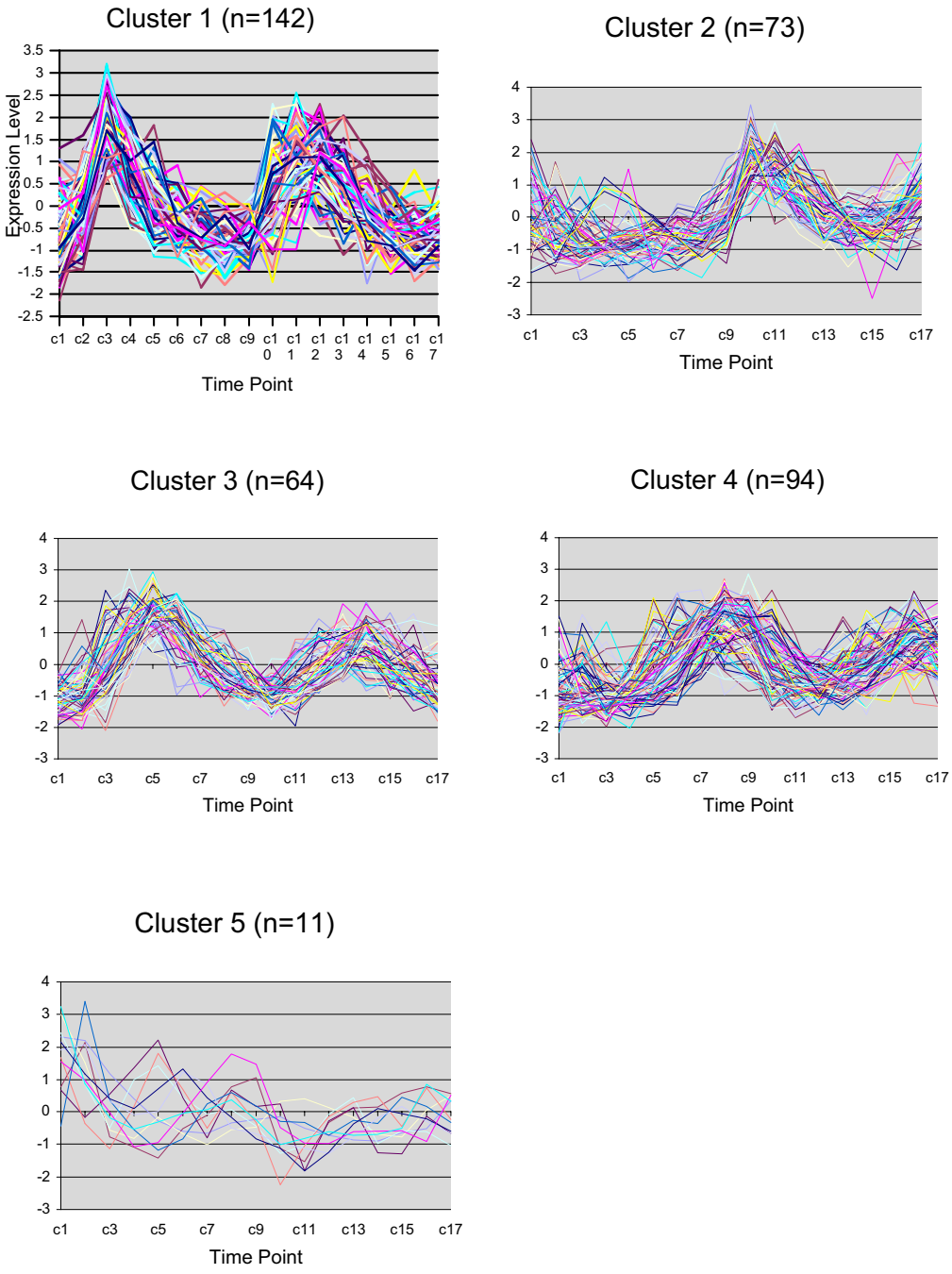



**Figure 4.10.3** Clusters as assigned by UPGMACOR.
Note that the members of cluster 5 are largely uncorrelated.

In table 4.10.2 below, the compositions of each cluster are shown, according to the Cho annotations. It becomes apparent that there are three clusters that each contain a

large number of genes with a specific annotation. Cluster 1 has mostly genes with late G1 annotations, cluster 2 has a large amount of genes with early G1 annotations, and cluster 4 has genes with mostly M phase annotations.

Cluster 1

| Phase | n-genes | Total in Phase |
|---|---|---|
| Early G1 | 4 | 67 |
| Late G1 | 113 | 135 |
| S | 25 | 75 |
| G2 | 0 | 52 |
| M | 0 | 55 |

(a)

cluster2

| Phase | n-genes | Total in Phase |
|---|---|---|
| Early G1 | 49 | 67 |
| Late G1 | 18 | 135 |
| S | 5 | 75 |
| G2 | 0 | 52 |
| M | 1 | 55 |

(b)

cluster 3

| Phase | n-genes | Total in Phase |
|---|---|---|
| Early G1 | 0 | 67 |
| Late G1 | 3 | 135 |
| S | 36 | 75 |
| G2 | 24 | 52 |
| M | 1 | 55 |

(c)

cluster 4

| Phase | n-genes | Total in Phase |
|---|---|---|
| Early G1 | 12 | 67 |
| Late G1 | 0 | 135 |
| S | 6 | 75 |
| G2 | 24 | 52 |
| M | 52 | 55 |

(d)

cluster 5

| Phase | n-genes | Total in Phase |
|---|---|---|
| Early G1 | 2 | 67 |
| Late G1 | 1 | 135 |
| S | 3 | 75 |
| G2 | 4 | 52 |
| M | 1 | 55 |

(e)

**Table 4.10.2.** Composition of clusters for UPGMACOR by annotation. Note that cluster 1 (a) is mostly composed of genes with Late G1 annotations, cluster 2 (b), genes with Early G1 annotations, and cluster 4 (d), genes with M annotations.

The question is whether the other clusterings with similar within-method metrics have analogous clusters with similar genes. On further examination, there are analogous

clusters to cluster 1, 2 and 4 in both UPGMAEUC and DIANAEUC. Figure 4.10.4 shows analogous clusters to cluster 2 across all three methods.

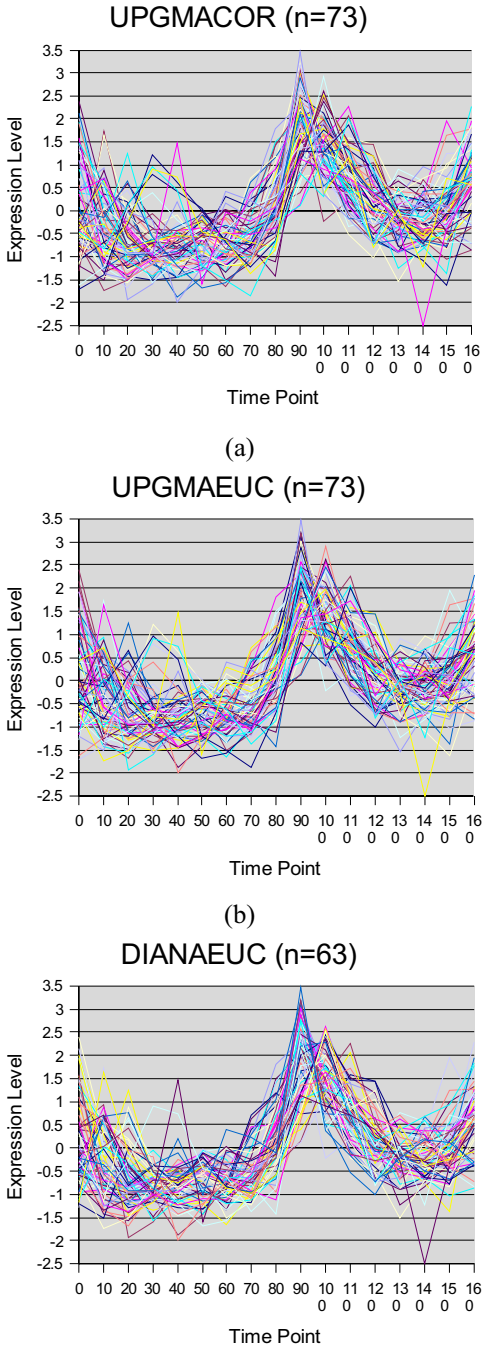



**Figure 4.10.4.** Visual comparison of analogous clusters to cluster 2. a) UPGMACOR, b) UPGMAEUC, and c) DIANAEUC.

A comparison of the composition of cluster 2 across all three methods can be seen in table 4.10.3 below. All three analogous clusters contain at least 45 genes with the Early G1 annotation.

| Phase | UPGMACOR | UPGMAEUC | DIANAEUC |
|---|---|---|---|
| Early G1 | 49 | 49 | 45 |
| Late G1 | 18 | 18 | 18 |
| S | 5 | 5 | 3 |
| G2 | 0 | 0 | 0 |
| M | 1 | 1 | 1 |
| Total # | 73 | 73 | 67 |

**Table 4.10.3.** Comparison of clusters analogous to UPGMACOR cluster 2. Note all three have a large number of genes with Early G1 annotations.

The overlap, or intersection, of genes between all three analogous clusters was calculated within R (see Appendix A for sample code to calculate overlaps). A summary can be seen in table 4.10.4. Note that the overlap captures 45 of the 67 total genes with early G1 annotations. Also note that 16 genes that have late G1 annotations are found in all three clusters. This suggests that the grouping of genes with G1 annotation is somewhat stable across all three methods.

| Phase | Overlap | # of Members |
|---|---|---|
| Early G1 | 45 | 67 |
| Late G1 | 16 | 135 |
| S | 3 | 75 |
| G2 | 0 | 52 |
| M | 1 | 55 |
| Total Genes | 65 | 384 |

**Table 4.10.4.** Overlap between the three analogous clusters. Note that 45 genes with Early G1 annotation remain.

A list of the genes that remain after the overlap can be seen in table 4.10.5. References have been given where possible in order to justify the actual phase the genes are active in. The biological interpretation of this cluster is a little difficult. First of all,

there are several genes within this cluster that are well associated with the Early G1 stage. Lee, et al. have noted that PCL9 complexes with PHO85 in the early stages of morphogenesis. [25] Han, et al. have noted that CLN3 helps to control initiation of cell division. [26] MF(ALPHA)1 and MF(ALPHA)2 are mating factors, and their presence suggests that the cells are arrested in G1 phase. [27] FAR1 is part of a cascade (which includes the mating factors) that keeps the cell in G1 cell arrest. [27, 28] CDC6 is a licensing factor that is necessary for the coating of DNA in order to start DNA replication and has been shown to accumulate in the cell during the G1 phase. [29] ASH1 is an interesting example. Earlier studies by Long, et al. and Takizawa et al. suggest that ASH1 is expressed in the M phase. [30, 31] However, a later study by McBride suggests that this gene is expressed in the Early G1 phase. [32]

The second grouping includes genes that function in different phases than the G1 phase. CTS1, CYK3, SCW11, and DSE4 are associated with cell cytokinesis, a function associated with the M phase. [33, 34] This presence of these genes in this cluster suggests that the biological interpretation of this cluster is somewhat suspect.

**Table 4.10.5.** Overlap of three methods for cluster 2.

| ORF | Gene Name | Description | GO Molecular Function | GO Biological Process | Cho Annotation | Evidence |
|---|---|---|---|---|---|---|
| *Genes regulating G1 cycle* | | | | | | |
| YLR274W | CDC46 | Component of the hexameric MCM complex, which is important for priming origins of DNA replication in G1 and becomes an active ATP-dependent helicase that promotes DNA melting and elongation when activated by Cdc7p-Dbf4p in S-phase | chromatin binding, ATP dependent DNA helicase activity | pre-replicative complex formation and maintenance, DNA replication initiation, DNA unwinding, establishment of chromatin silencing | Early G1 | Late G1/S |
| YDL179W | PCL9 | PHO85 cyclin | cyclin-dependent protein kinase regulator activity | cell cycle | Early G1 | G1 (Lee, et al. [25]) |
| YBR158W | AMN1 | Antagonist of MEN (Mitotic Exit Network); Chromosome Stability | protein binding | negative regulation of exit from mitosis, mitotic checkpoint | Early G1 | G1 (Guertin, et al. [35], Wang, et al. [36]) |
| YLR079W | SIC1 | P40 inhibitor of Cdc28p-Clb5 protein kinase complex | kinase inhibitor activity, protein binding | G1/S transition of mitotic cell cycle, regulation of CDK activity | Early G1 | Late G1/S |
| YAL040C | CLN3 | G1 cyclin | cyclin-dependent protein kinase regulator activity | regulation of CDK activity, G1/S transition of mitotic cell cycle | M | Late G1/S (Han, et al. [26]) |
| YJL194W | CDC6 | pre-initiation complex component | protein binding, DNA clamp loader activity, ATPase activity | pre-replicative complex formation and maintenance | Early G1 | Late G1/S (Piatti, et al. [29]) |
| YJL157C | FAR1 | Cdc28p kinase inhibitor | cyclin-dependent protein kinase inhibitor activity | signal transduction during conjugation with cellular fusion, cell cycle arrest | Early G1 | G1 (McKinney, et al. [27]) |
| YPL187W | MF(ALPHA)1 | mating factor alpha | pheromone activity | response to pheromone during conjugation with cellular fusion | Late G1 | G1 (McKinney, et al. [27]) |
| YGL089C | MF(ALPHA)2 | alpha mating factor | pheromone activity | response to pheromone during conjugation with cellular fusion | Late G1 | G1 (McKinney, et al. [27]) |
| *Genes possibly misplaced* | | | | | | |
| YDL117W | CYK3 | SH3-domain protein located in the mother-bud neck and the cytokinetic actin ring; mutant phenotype and genetic interactions suggest a role in cytokinesis | molecular_function unknown | cytokinesis | Early G1 | M |
| YGL028C | SCW11 | glucanase | glucan 1,3-beta-glucosidase activity | cytokinesis, completion of separation | Late G1 | M |
| YNR067C | DSE4 | Daughter cell-specific secreted protein with similarity to glucanases, degrades cell wall from the daughter side causing daughter to separate from mother | glucan 1,3-beta-glucosidase activity | cytokinesis, completion of separation | Early G1 | M |

**Table 4.10.5.** Overlap of three methods for cluster 2.

| ORF | Gene Name | Description | GO Molecular Function | GO Biological Process | Cho Annotation | Evidence |
|---|---|---|---|---|---|---|
| YLR286C | CTS1 | endochitinase | chitinase activity | cytokinesis, completion of separation | Late G1 | M (King, et al., Kuranda, et al.) |
| YKL185W | ASH1 | Zinc-finger inhibitor of HO transcription; mRNA is localized and translated in the distal tip of anaphase cells, resulting in accumulation of Ash1p in daughter cell nuclei and inhibition of HO expression; potential Cdc28p substrate | specific transcriptional repressor activity | regulation of transcription, mating-type specific, pseudohyphal growth | Early G1 | M (Long, et al.), M (Takizawa, et al.), Early G1(McBride, et al.) |
| *Other Genes* | | | | | | |
| YDR368W | YPR1 | 2-methylbutyraldehyde reductase, may be involved in isoleucine catabolism | aldehyde reductase activity, aldo-keto reductase activity, oxidoreductase activity | arabinose metabolism, D-xylose metabolism | Early G1 | |
| YHR113W | NA | Hypothetical ORF | aminopeptidase activity | biological_process unknown | Late G1 | |
| YER017C | AFG3 | ATP dependent metalloprotease | ATPase activity, metallopeptidase activity | proteolysis and peptidolysis, protein complex assembly, mitochondrial intermembrane space protein import | S | |
| YJR148W | BAT2 | branched-chain amino acid transaminase | branched-chain-amino-acid transaminase activity | branched chain family amino acid biosynthesis, amino acid catabolism | Late G1 | |
| YCR005C | CIT2 | citrate synthase | citrate (Si)-synthase activity | glutamate biosynthesis, citrate metabolism, glyoxylate cycle | Early G1 | |
| YNR001C | CIT1 | citrate synthase | citrate (Si)-synthase activity | glutamate biosynthesis, citrate metabolism, tricarboxylic acid cycle | Early G1 | |
| YKL150W | MCR1 | NADH-cytochrome b5 reductase | cytochrome-b5 reductase activity | response to oxidative stress, electron transport, ergosterol biosynthesis | Early G1 | |
| YOR065W | CYT1 | cytochrome c1 | electron transporter, transferring electrons within CoQH2-cytochrome c reductase complex activity | mitochondrial electron transport, ubiquinol to cytochrome c, oxidative phosphorylation | Early G1 | |
| YDL181W | INH1 | ATPase inhibitor | enzyme inhibitor activity | ATP synthesis coupled proton transport | Early G1 | |
| YCL040W | GLK1 | glucokinase | glucokinase activity | carbohydrate metabolism | Early G1 | |
| YLR258W | GSY2 | glycogen synthase (UDP-glucose-starch glucosyltransferase) | glycogen (starch) synthase activity | glycogen metabolism | Early G1 | |
| YHR005C | GPA1 | G protein alpha subunit\|coupled to mating factor receptor | GTPase activity | signal transduction during conjugation with cellular fusion | Early G1 | |
| YIL009W | FAA3 | acyl-CoA synthase | long-chain-fatty-acid-CoA ligase activity | lipid metabolism, N-terminal protein myristoylation | Early G1 | |

**Table 4.10.5.** Overlap of three methods for cluster 2.

| ORF | Gene Name | Description | GO Molecular Function | GO Biological Process | Cho Annotation | Evidence |
|---|---|---|---|---|---|---|
| YOR317W | FAA1 | long chain fatty acyl:CoA synthetase | long-chain-fatty-acid-CoA ligase activity | lipid metabolism, N-terminal protein myristoylationtion, lipid transport | Late G1 | |
| YNL073W | MSK1 | lysine-tRNA ligase | lysine-tRNA ligase activity | lysyl-tRNA aminoacylation | S | |
| YBR231C | SWC5 | Protein of unknown function, component of the Swr1p complex that incorporates Htz1p into chromatin | molecular_function unknown | chromatin remodeling | Early G1 | |
| YDR511W | ACN9 | Protein of the mitochondrial intermembrane space, required for acetate utilization and gluconeogenesis; has orthologs in higher eukaryotes | molecular_function unknown | gluconeogenesis, carbon utilization by utilization of organic compounds | Early G1 | |
| YLL040C | VPS13 | homologous to human COH1 | molecular_function unknown | late endosome to vacuole transport, protein-Golgi retention | Early G1 | |
| YHR039C | MSC7 | Protein of unknown function, green fluorescent protein (GFP)-fusion protein localizes to the endoplasmic reticulum; msc7 mutants are defective in directing meiotic recombination events to homologous chromatids | molecular_function unknown | meiotic recombination | Late G1 | |
| YPR002W | PDH1 | prpD homologue; (62% identical to the prpD genes of Escherichia coli and Salmonella typhimurium, which play an unknown but essential role in propionate catabolism) | molecular_function unknown | propionate metabolism | Early G1 | |
| YNL078W | NIS1 | | molecular_function unknown | regulation of mitosis | Early G1 | |
| YNL173C | MDG1 | multicopy suppressor of bem1 mutation, may be involved in G-protein mediated signal transduction; binds cruciform DNA | molecular_function unknown | signal transduction during conjugation with cellular fusion | Late G1 | |
| YKL161C | NA | Mpk1-like protein kinase; associates with Rlm1p | protein kinase activity | biological_process unknown | Late G1 | |
| YLR273C | PIG1 | similar to Gac1p, a putative type 1 protein phosphatase targeting subunit | protein phosphatase regulator activity | regulation of glycogen biosynthesis | Early G1 | |
| YKL116C | PRR1 | protein kinase | receptor signaling protein serine/threonine kinase activity, receptor signaling protein serine/threonine kinase activity | MAPKKK cascade | Early G1 | |
| YDR309C | GIC2 | Gtpase-interacting component 2 | small GTPase regulatory/interacting protein activity | establishment of cell polarity (sensu Saccharomyces), Rho protein signal transduction, axial budding | Late G1 | |

**Table 4.10.5.** Overlap of three methods for cluster 2.

| ORF | Gene Name | Description | GO Molecular Function | GO Biological Process | Cho Annotation | Evidence |
|---|---|---|---|---|---|---|
| YBR083W | TEC1 | TEA/ATTS DNA-binding domain family, regulator of Ty1 expression\|transcription factor | specific RNA polymerase II transcription factor activity | positive regulation of transcription from Pol II promoter, pseudohyphal growth | Early G1 | |
| YGR044C | RME1 | zinc finger protein\|negative regulator of meiosis; directly repressed by a1-alpha 2 regulator | specific transcriptional repressor activity | meiosis, negative regulation of transcription from Pol II promoter | Early G1 | |
| YKL163W | PIR3 | Protein containing tandem internal repeats | structural constituent of cell wall | cell wall organization and biogenesis | Early G1 | |
| YHR038W | RRF1 | mitochondrial ribosome recycling factor | translation termination factor activity | protein biosynthesis | Late G1 | |
| YDL119C | NA | Hypothetical ORF | transporter activity | transport | Late G1 | |
| YGR183C | QCR9 | ubiquinol cytochrome c oxidoreductase complex 7.3 kDa subunit 9 | ubiquinol-cytochrome-c reductase activity | iron-sulfur cluster assembly, mitochondrial electron transport, ubiquinol to cytochrome c, aerobic respiration | Early G1 | |
| YML110C | COQ5 | C-methyltransferase (putative) | ubiquinone biosynthesis methyltransferase activity | ubiquinone metabolism, aerobic respiration | Early G1 | |
| YOR316C | COT1 | Vacuolar transporter that mediates zinc transport into the vacuole; overexpression confers resistance to cobalt and rhodium | zinc ion transporter activity, cobalt ion transporter activity, di-, tri-valent inorganic cation transporter activity | zinc ion transport, zinc ion homeostasis, cobalt ion transport | Late G1 | |

A comparison of the composition of cluster 1 across all three methods can be seen in Table 4.10.6. below. Clearly, all three analogous clusters contain a large number of genes with Late G1 annotations.

| Phase | UPGMACOR | UPGMAEUC | DIANAEUC |
|---|---|---|---|
| Early G1 | 4 | 4 | 4 |
| Late G1 | 113 | 116 | 116 |
| S | 25 | 41 | 44 |
| G2 | 0 | 1 | 1 |
| M | 0 | 0 | 0 |
| Total Genes | 142 | 162 | 165 |

**Table 4.10.6.** Clusters analogous to UPGMACOR cluster 1. Note the large number of genes with Late G1 annotations.

The overlap between these three analogous clusters was calculated. A summary table can be seen in Table 4.10.7. Note that the overlap captures 111 of the 135 genes with Late G1 annotations. That these genes are found grouped together across three clusters suggest that the cluster is quite stable across these three methods. The three methods group 25 genes with S annotations consistently as well.

| Phase | Overlap | # of Members |
|---|---|---|
| Early G1 | 4 | 67 |
| Late G1 | 111 | 135 |
| S | 25 | 75 |
| G2 | 0 | 52 |
| M | 0 | 55 |
| Total Genes | 140 | 384 |

**Table 4.10.7.** Overlap of clusters. Note the high number of genes with late G1 annotation that remain.

A list of genes that occur in the cluster and their GO functional annotations can be seen in table 4.10.8. Genes that had no GO molecular function annotation or the annotation “molecular_function_unknown” were eliminated from the list. There are

three major groupings of genes that immediately stick out as being associated with each other within this overlapping cluster. The first grouping has to do with DNA replication. DNA Replication occurs during the Late G1 phase. The second grouping has to do with cytostructural components, which are needed for replication. A third group, the cyclin-dependent kinases, is well associated with the Late G1 phase. McBride and Wittenberg are both excellent discussions of these kinases. [28, 32] In fact, even though there are a number of GO annotations that simply refer to "protein_kinase_activity," it is tempting to guess that they are cyclin-associated.

However, there are a number of genes on first examination that are somewhat questionable members of this cluster, HCM1, TOS4, SWI4, and STB1, all of which are transcription factors. HCM1 and TOS4, however, are the targets of a cyclin dependent kinase and HCM1, STB1 and SWI4 are also associated with the G1/S transition. [37, 38] BNI4 is a gene associated with cytokinesis, a M phase function.

Another gene that potentially doesn't belong in this cluster is TUB4. TUB4 is considered a housekeeping gene whose transcripts are usually constant unless the cell is dividing into two daughter cells. However, TUB4 also regulates microtubules in budding yeast, which may explain why it is in this cluster. [39]

All in all, this cluster seems to be relatively well grouped by function, and better defined than cluster 2.

**Table 4.10.8**. Composition of overlap for Cluster 1.

| ORF | Gene Name | Description | GO Molecular Function | GO Biological Process | Cho annotation | Evidence |
|---|---|---|---|---|---|---|
| *Structural Components* | | | | | | |
| YNL225C | CNM67 | chaotic nuclear migration; predicted mass is 67kDa | structural constituent of cytoskeleton | microtubule nucleation | Late G1 | Late G1 |
| YKR083C | DAD2 | Duo1 And Dam1 interacting; Helper of AsK1 | structural constituent of cytoskeleton | mitotic spindle assembly (sensu Fungi) | Late G1 | Late G1 |
| YPL124W | SPC29 | spindle pole body component | structural constituent of cytoskeleton | spindle pole body duplication (sensu Saccharomyces), microtubule nucleation | Late G1 | Late G1 |
| YKL042W | SPC42 | spindle pole body component | structural constituent of cytoskeleton | spindle pole body duplication (sensu Saccharomyces), microtubule nucleation | Late G1 | Late G1 |
| *CDKs associated with G1/S transition* | | | | | | |
| YDL127W | PCL2 | G1 cyclin | cyclin-dependent protein kinase regulator activity | cell cycle | Late G1 | Late G1 (McBride, et al. [32]) |
| YNL289W | PCL1 | G1 cyclin\|associates with PHO85 | cyclin-dependent protein kinase regulator activity | cell cycle | Late G1 | Late G1 (McBride, et al. [32]) |
| YPL256C | CLN2 | G1 cyclin | cyclin-dependent protein kinase regulator activity | re-entry into mitotic cell cycle after pheromone arrest, regulation of CDK activity | Late G1 | Late G1 (Wittenberg, et al. [28]) |
| YMR199W | CLN1 | G1 cyclin | cyclin-dependent protein kinase regulator activity | regulation of CDK activity | Late G1 | Late G1 (Wittenberg, et al. [28]) |
| YDR507C | GIN4 | serine/threonine kinase (putative) | protein kinase activity | axial budding, bud growth, septin checkpoint, septin ring assembly, protein amino acid phosphorylation | Late G1 | |
| YCL024W | KCC4 | S. pombe Nim1 homolog\|protein kinase | protein kinase activity | axial budding, septin ring assembly, bud growth, septin checkpoint, protein amino acid phosphorylation | Late G1 | |
| YDL101C | DUN1 | protein kinase | protein kinase activity | cell cycle checkpoint, DNA damage response, signal transduction resulting in cell cycle arrest, protein amino acid phosphorylation | Late G1 | |

**Table 4.10.8**. Composition of overlap for Cluster 1.

| ORF | Gene Name | Description | GO Molecular Function | GO Biological Process | Cho annotation | Evidence |
|---|---|---|---|---|---|---|
| *DNA Repair* | | | | | | |
| YKL113C | RAD27 | 42 kDa 5' to 3' exonuclease required for Okazaki fragment processing | 5' flap endonuclease activity | DNA repair, DNA replication, DNA repair synthesis, replicative cell aging | Late G1 | Late G1/S |
| YBL035C | POL12 | DNA polymerase alpha-primase complex B subunit | alpha DNA polymerase activity | lagging strand elongation, DNA replication initiation, DNA replication, priming | Late G1 | Late G1 |
| YNL102W | POL1 | DNA polymerase I alpha subunit p180 | alpha DNA polymerase activity | DNA repair synthesis, lagging strand elongation, DNA replication initiation, DNA replication, priming | Late G1 | Late G1/S |
| YKL045W | PRI2 | DNA primase p58 polypeptide | alpha DNA polymerase activity | DNA repair synthesis, lagging strand elongation, DNA replication initiation, DNA replication, priming | Late G1 | Late G1/S |
| YLR032W | RAD5 | ATPase (putative)\|DNA helicase (putative) | ATPase activity | DNA repair | Late G1 | Late G1 |
| YOL090W | MSH2 | mutS homolog | ATPase activity, ATP binding, damaged DNA binding | DNA recombination, mitotic recombination, removal of nonhomologous ends, mismatch repair | Late G1 | Late G1 |
| YGR109C | CLB6 | B-type cyclin | cyclin-dependent protein kinase regulator activity | G1/S transition of mitotic cell cycle, premeiotic DNA synthesis, regulation of CDK activity | Late G1 | Late G1/S |
| YPR120C | CLB5 | B-type cyclin | cyclin-dependent protein kinase regulator activity | G2/M transition of mitotic cell cycle, G1/S transition of mitotic cell cycle, premeiotic DNA synthesis, regulation of CDK activity | Late G1 | Late G1/S, G2/M |
| YDR013W | PSF1 | a subunit of the GINS complex required for chromosomal DNA replication | DNA binding | DNA dependent DNA replication | Late G1 | Late G1 |
| YNL312W | RFA2 | 29% identical to the human p34 subunit of RF-A\|replication factor RF-A subunit 2 | DNA binding | DNA recombination, double-strand break repair, postreplication repair, nucleotide-excision repair, DNA strand elongation, DNA replication, priming, DNA unwinding | Late G1 | Late G1 |
| YJL173C | RFA3 | replication factor-A subunit 3 | DNA binding | DNA recombination, double-strand break repair, postreplication repair, nucleotide-excision repair, DNA strand elongation, DNA replication, priming, DNA unwinding | Late G1 | Late G1 |

**Table 4.10.8**. Composition of overlap for Cluster 1.

| ORF | Gene Name | Description | GO Molecular Function | GO Biological Process | Cho annotation | Evidence |
|---|---|---|---|---|---|---|
| YLR103C | CDC45 | chromosomal DNA replication initiation protein | DNA binding | pre-replicative complex formation and maintenance, DNA replication initiation, DNA strand elongation | Late G1 | Late G1 |
| YML061C | PIF1 | 5'-3' DNA helicase | DNA helicase activity | DNA recombination, chromosome organization and biogenesis (sensu Eukarya), telomere maintenance | Late G1 | Late G1 |
| YDL164C | CDC9 | DNA ligase | DNA ligase (ATP) activity | DNA recombination, nucleotide-excision repair, base-excision repair, lagging strand elongation, lagging strand elongation, DNA ligation, DNA ligation | Late G1 | Late G1 |
| YNL262W | POL2 | DNA polymerase II | epsilon DNA polymerase activity | DNA repair synthesis, nucleotide-excision repair, lagging strand elongation, leading strand elongation, mismatch repair, chromatin silencing at telomere | Late G1 | Late G1 |
| YPR175W | DPB2 | DNA polymerase epsilon subunit B | epsilon DNA polymerase activity | nucleotide-excision repair, lagging strand elongation, leading strand elongation, mismatch repair | Late G1 | Late G1 |
| YBR278W | DPB3 | DNA polymerase II C and C' subunits | epsilon DNA polymerase activity | nucleotide-excision repair, lagging strand elongation, leading strand elongation, mismatch repair, chromatin silencing at telomere | Late G1 | Late G1 |
| YJL115W | ASF1 | anti-silencing protein that causes depression of silent loci when overexpressed | histone binding | DNA damage response, signal transduction resulting in induction of apoptosis | S | Late G1 |
| YAL034W-A | MTW1 | Mis TWelve like (a Schizosaccharomyces pombe kinetochore protein) | molecular_function unknown | chromosome segregation | S | Late G1 |
| YNL273W | TOF1 | topoisomerase I interacting factor 1 | molecular_function unknown | DNA topological change, DNA replication checkpoint | Late G1 | Late G1 |
| YMR048W | CSM3 | Protein required for accurate chromosome segregation during meiosis | molecular_function unknown | meiotic chromosome segregation, DNA replication checkpoint | S | Late G1 |
| YLR383W | RHC18 | Protein involved in recombination repair, homologous to S. pombe rad18. Structural maintenance of chromosomes (SMC) protein. | molecular_function unknown | cell proliferation, DNA repair | Late G1 | Late G1 |
| YML102W | CAC2 | chromatin assembly factor-I (CAF-I) p60 subunit | molecular_function unknown | chromatin silencing, DNA repair, nucleosome assembly | Late G1 | Late G1 |

**Table 4.10.8**. Composition of overlap for Cluster 1.

| ORF | Gene Name | Description | GO Molecular Function | GO Biological Process | Cho annotation | Evidence |
|---|---|---|---|---|---|---|
| YLR381W | CTF3 | Outer kinetochore protein that forms a complex with Mcm16p and Mcm22p; may bind the kinetochore to spindle microtubules | protein binding | chromosome segregation | Late G1 | Late G1 |
| YPL209C | IPL1 | protein kinase | protein kinase activity | chromosome segregation | Late G1 | Late G1 |
| YAR007C | RFA1 | heterotrimeric RPA (RF-A) single-stranded DNA binding protein 69 kDa subunit\|RF-A | single-stranded DNA binding, damaged DNA binding | DNA recombination, double-strand break repair, postreplication repair, nucleotide-excision repair, DNA strand elongation, DNA replication, priming, DNA unwinding | Late G1 | Late G1 |
| YML021C | UNG1 | uracil DNA glycosylase | uracil DNA N-glycosylase activity | DNA repair | Late G1 | Late G1 |
| *Various Genes Expressed during G1* | | | | | | |
| YLR183C | TOS4 | Transcription factor that binds to a number of promoter regions, particularly promoters of some genes involved in pheromone response and cell cycle; potential Cdc28p substrate; expression is induced in G1 by bound SBF | transcription factor activity | biological_process unknown | Late G1 | Late G1 (Horak, et al. [37] , Ubersax, et al. [38]) |
| YER111C | SWI4 | transcription factor | transcription factor activity | transcription, cell cycle, G1/S transition of mitotic cell cycle | Early G1 | Late G1 (Horak, et al. [37] , Ubersax, et al. [38]) |
| YNL309W | STB1 | Protein with a role in regulation of MBF-specific transcription at Start, phosphorylated by Cln-Cdc28p kinases in vitro; unphosphorylated form binds Swi6p and binding is required for Stb1p function; expression is cell-cycle regulated | transcriptional activator activity | G1/S transition of mitotic cell cycle | Late G1 | Late G1 (Horak, et al. [37] , Ubersax, et al. [38]) |
| YCR065W | HCM1 | forkhead protein | specific RNA polymerase II transcription factor activity | transcription initiation from Pol II promoter, spindle assembly | S | Late G1 (Horak, et al. [37] , Ubersax, et al. [38]) |
| YLR212C | TUB4 | gamma tubulin | structural constituent of cytoskeleton | mitotic spindle assembly (sensu Fungi), microtubule nucleation | Late G1 | G1 (Vogel, et al. [39]) |
| YDL227C | HO | homothallic switching endonuclease | endonuclease activity | mating-type switching/recombination, gene conversion at MAT locus | Late G1 | Late G1 (Dohrman, et al. ) |

**Table 4.10.8**. Composition of overlap for Cluster 1.

| ORF | Gene Name | Description | GO Molecular Function | GO Biological Process | Cho annotation | Evidence |
|---|---|---|---|---|---|---|
| YML027W | YOX1 | homeobox-domain containing protein | DNA binding, specific transcriptional repressor activity | regulation of mitotic cell cycle, negative regulation of transcription from Pol II promoter, mitotic | Late G1 | M/G1 phase (McBride, et al. [32]) |
| *Genes possibly misplaced* | | | | | | |
| YNL233W | BNI4 | required to link Chs3p and Chs4p to the septins | protein binding | cytokinesis | Late G1 | M |
| YKL101W | HSL1 | serine-threonine kinase | protein kinase activity | G2/M transition of mitotic cell cycle, cell morphogenesis checkpoint, protein amino acid phosphorylation, septin checkpoint, septin checkpoint, regulation of cell cycle | Late G1 | M |
| YJL187C | SWE1 | tyrosine kinase | protein kinase activity | regulation of CDK activity, G2/M transition of mitotic cell cycle, regulation of meiosis, cell morphogenesis checkpoint | Late G1 | M |
| YNL072W | RNH201 | Ribonuclease H2 catalytic subunit, removes RNA primers during Okazaki fragment synthesis; cooperates with Rad27p nuclease | ribonuclease H activity | DNA replication | S | S |
| YER070W | RNR1 | ribonucleotide reductase, large (R1) subunit | ribonucleoside-diphosphate reductase activity | DNA replication | Late G1 | S |
| YOR074C | CDC21 | thymidylate synthase | thymidylate synthase activity | DNA dependent DNA replication, dTMP biosynthesis | Late G1 | S |
| *Other Genes* | | | | | | |
| YNR016C | ACC1 | acetyl CoA carboxylase | acetyl-CoA carboxylase activity, biotin carboxylase activity, | nuclear membrane organization and biogenesis, fatty acid biosynthesis, protein-nucleus import | Early G1 | |
| YJR155W | AAD10 | aryl-alcohol dehydrogenase (putative) | aryl-alcohol dehydrogenase activity | aldehyde metabolism | S | |
| YLR121C | YPS3 | GPI-anchored aspartic protease | aspartic-type endopeptidase activity, aspartic-type endopeptidase activity | protein metabolism, protein metabolism | Late G1 | |
| YNL082W | PMS1 | mutL homolog\|similar to Mlh1p, associates with Mlh1p, possibly forming a heterodimer, Pms1p and Msh1p act in concert to bind to a Msh2p-heteroduplex complex containing a G-T mismatch | ATP binding, DNA binding, ATPase activity | mismatch repair, meiosis | Late G1 | |

**Table 4.10.8**. Composition of overlap for Cluster 1.

| ORF | Gene Name | Description | GO Molecular Function | GO Biological Process | Cho annotation | Evidence |
|---|---|---|---|---|---|---|
| YJL074C | SMC3 | SMC chromosomal ATPase family member | ATPase activity | sporulation (sensu Saccharomyces), synaptonemal complex formation, mitotic sister chromatid cohesion | Late G1 | |
| YDR097C | MSH6 | human GTBP protein homolog | ATPase activity, ATP binding, DNA binding | mismatch repair | Late G1 | |
| YIL159W | BNR1 | Formin, nucleates the formation of linear actin filaments, involved in cell processes such as budding and mitotic spindle orientation which require the formation of polarized actin cables, functionally redundant with BNI1 | cytoskeletal protein binding | axial budding, response to osmotic stress, actin filament organization | Late G1 | |
| YLR313C | SPH1 | Spa2p homolog | cytoskeletal regulatory protein binding | polar budding, pseudohyphal growth, establishment of cell polarity (sensu Saccharomyces), establishment of cell polarity (sensu Saccharomyces), Rho protein signal transduction, actin filament organization | Late G1 | |
| YJR043C | POL32 | 55 kDa\|DNA polymerase delta subunit | delta DNA polymerase activity | postreplication repair, mismatch repair, nucleotide-excision repair, base-excision repair, mutagenesis, lagging strand elongation, leading strand elongation | Late G1 | |
| YBR073W | RDH54 | helicase (putative)\|similar to RAD54 | DNA dependent ATPase activity, DNA supercoiling activity | meiotic recombination, meiotic recombination, heteroduplex formation, double-strand break repair via break-induced replication | Late G1 | |
| YBR088C | POL30 | Proliferating Cell Nuclear Antigen (PCNA) | DNA polymerase processivity factor activity | postreplication repair, mismatch repair, nucleotide-excision repair, base-excision repair, mutagenesis, lagging strand elongation, leading strand elongation | Late G1 | |
| YLR234W | TOP3 | DNA topoisomerase III | DNA topoisomerase type I activity | telomerase-dependent telomere maintenance, meiotic recombination, regulation of DNA recombination | Late G1 | |

**Table 4.10.8**. Composition of overlap for Cluster 1.

| ORF | Gene Name | Description | GO Molecular Function | GO Biological Process | Cho annotation | Evidence |
|---|---|---|---|---|---|---|
| YDL095W | PMT1 | dolichyl phosphate-D-mannose:protein O-D-mannosyltransferase | dolichyl-phosphate-mannose-protein mannosyltransferase activity, dolichyl-phosphate-mannose-protein mannosyltransferase activity | O-linked glycosylation, O-linked glycosylation | S | |
| YFL008W | SMC1 | SMC chromosomal ATPase family member | double-stranded DNA binding, AT DNA binding, DNA secondary structure binding, ATPase activity | mitotic chromosome segregation | Late G1 | |
| YJL196C | ELO1 | elongase | fatty acid elongase activity | fatty acid elongation, unsaturated fatty acid | Late G1 | |
| YNL304W | YPT11 | acts positively on mitochondrial distribution toward the bud. | GTPase activity | mitochondrion inheritance | S | |
| YGR152C | RSR1 | Gtp-binding protein of the ras superfamily involved in bud site selection | GTPase activity, GTPase activity, signal transducer activity | polar budding, axial budding, bud site selection, small GTPase mediated signal transduction | Late G1 | |
| YNL272C | SEC2 | GDP/GTP exchange factor | guanyl-nucleotide exchange factor activity | exocytosis | Late G1 | |
| YFR038W | NA | Hypothetical ORF | helicase activity | biological_process unknown | S | |
| YLR382C | NAM2 | leucine-tRNA ligase | leucine-tRNA ligase activity, mRNA binding | Group I intron splicing, leucyl-tRNA aminoacylation | Late G1 | |
| YGL027C | CWH41 | glucosidase I | mannosyl-oligosaccharide glucosidase activity | cell wall organization and biogenesis | Late G1 | |
| YDR488C | PAC11 | Protein required in the absence of Cin8p | microtubule motor activity | microtubule-based process | S | |
| YOR284W | HUA2 | Cytoplasmic protein of unknown function; computational analysis of large-scale protein-protein interaction data suggests a possible role in actin patch assembly | molecular_function unknown | actin cortical patch assembly | S | |
| YGR041W | BUD9 | Protein involved in bud-site selection; diploid mutants display a unipolar budding pattern instead of the wild-type bipolar pattern, and bud at the distal pole | molecular_function unknown | bud site selection | Late G1 | |
| YIL140W | AXL2 | Integral plasma membrane protein required for axial budding in haploid cells, localizes to the incipient bud site and bud neck; glycosylated by Pmt4p; potential Cdc28p substrate | molecular_function unknown | bud site selection, axial budding | S | |

**Table 4.10.8**. Composition of overlap for Cluster 1.

| ORF | Gene Name | Description | GO Molecular Function | GO Biological Process | Cho annotation | Evidence |
|---|---|---|---|---|---|---|
| YML109W | ZDS2 | Protein that interacts with silencing proteins at the telomere, involved in transcriptional silencing; paralog of Zds1p | molecular_function unknown | cell aging, chromatin silencing at ribosomal DNA, establishment of cell polarity (sensu Saccharomyces) | Early G1 | |
| YOL017W | ESC8 | Protein involved in telomeric and mating-type locus silencing, interacts with Sir2p and also interacts with the Gal11p, which is a component of the RNA pol II mediator complex | molecular_function unknown | chromatin silencing | Late G1 | |
| YDL197C | ASF2 | anti-silencing protein that causes depression of silent loci when overexpressed | molecular_function unknown | chromatin silencing at silent mating-type cassette (sensu Fungi) | S | |
| YCL061C | MRC1 | Mediator of the Replication Checkpoint; required for full activation of Rad53p in response to replication stress. | molecular_function unknown | chromatin silencing at silent mating-type cassette (sensu Fungi), DNA replication checkpoint, chromatin silencing at telomere | Late G1 | |
| YLL022C | HIF1 | Non-essential component of the HAT-B histone acetyltransferase complex (Hat1p-Hat2p-Hif1p), localized to the nucleus; has a role in telomeric silencing | molecular_function unknown | chromatin silencing at telomere | Late G1 | |
| YGR238C | KEL2 | Protein that functions in a complex with Kel1p to negatively regulate mitotic exit, interacts with Tem1p and Lte1p; localizes to regions of polarized growth; potential Cdc28p substrate | molecular_function unknown | conjugation with cellular fusion | Late G1 | |
| YKL108W | SLD2 | Protein required for DNA replication, phosphorylated in S phase by S-phase cyclin-dependent kinases (Cdks), phosphorylation is essential for DNA replication and for complex formation with Dpb11p; potential Cdc28p substrate | molecular_function unknown | DNA strand elongation | Late G1 | |
| YKL165C | MCD4 | Required for GPI anchor synthesis | molecular_function unknown | GPI anchor biosynthesis, ATP transport | Late G1 | |
| YPL241C | CIN2 | tubulin folding cofactor C | molecular_function unknown | microtubule-based process | Late G1 | |
| YDL003W | MCD1 | Mitotic Chromosome Determinant; similar to S. pombe RAD21; may function in chromosome morphogenesis from S phase through mitosis | molecular_function unknown | mitotic chromosome condensation, mitotic sister chromatid cohesion, mitotic sister chromatid cohesion | Late G1 | |

**Table 4.10.8**. Composition of overlap for Cluster 1.

| ORF | Gene Name | Description | GO Molecular Function | GO Biological Process | Cho annotation | Evidence |
|---|---|---|---|---|---|---|
| YOR026W | BUB3 | Protein required for cell cycle arrest in response to loss of microtubule function | molecular_function unknown | mitotic spindle checkpoint | S | |
| YOR144C | ELG1 | Protein required for S phase progression and telomere homeostasis, forms an alternative replication factor C complex important for DNA replication and genome integrity; mutants are sensitive to DNA damage | molecular_function unknown | negative regulation of DNA recombination, negative regulation of DNA transposition, DNA replication, telomere maintenance | Late G1 | |
| YHR154W | RTT107 | Regulator of Ty1 Transposition; Establishes Silent Chromatin | molecular_function unknown | negative regulation of DNA transposition | Late G1 | |
| YLL002W | RTT109 | Regulator of Ty1 Transposition; Regulation of mitochondrial network; Killed in Mutagen, sensitive to diepoxybutane and/or mitomycin C | molecular_function unknown | negative regulation of DNA transposition | Late G1 | |
| YPR018W | RLF2 | chromatin assembly factor-I (CAF-I) p90 subunit | molecular_function unknown | nucleosome assembly | Late G1 | |
| YMR179W | SPT21 | non-specific DNA binding protein | molecular_function unknown | regulation of transcription from Pol II promoter | Late G1 | |
| YDL018C | ERP3 | p24 protein involved in membrane trafficking | molecular_function unknown | secretory pathway | Late G1 | |
| YHR110W | ERP5 | p24 protein involved in membrane trafficking | molecular_function unknown | secretory pathway | Late G1 | |
| YMR078C | CTF18 | Subunit of a complex with Ctf8p that shares some subunits with Replication Factor C and is required for sister chromatid cohesion; may have overlapping functions with Rad24p in the DNA damage replication checkpoint | molecular_function unknown | sister chromatid cohesion | Late G1 | |
| YHR153C | SPO16 | Protein of unknown function, required for spore formation | molecular_function unknown | sporulation (sensu Saccharomyces) | Late G1 | |
| YBR275C | RIF1 | RAP1-interacting factor | molecular_function unknown | telomerase-dependent telomere maintenance, chromatin silencing at telomere | S | |
| YGL200C | EMP24 | type I transmembrane protein | molecular_function unknown | vesicle organization and biogenesis, ER to Golgi transport | Late G1 | |
| YER118C | SHO1 | transmembrane osmosensor | osmosensor activity | pseudohyphal growth, osmosensory signaling pathway via Sho1 osmosensor | S | |

**Table 4.10.8**. Composition of overlap for Cluster 1.

| ORF | Gene Name | Description | GO Molecular Function | GO Biological Process | Cho annotation | Evidence |
|---|---|---|---|---|---|---|
| YIL026C | IRR1 | cohesin complex subunit | protein binding | cytogamy, germination (sensu Saccharomyces), mitotic sister chromatid cohesion | Late G1 | |
| YPL153C | RAD53 | protein kinase | protein threonine/tyrosine kinase activity | nucleobase, nucleoside, nucleotide and nucleic acid metabolism, DNA repair | Late G1 | |
| YBR276C | PPS1 | dual specificity protein phosphatase | protein tyrosine/threonine phosphatase activity | vitamin B12 reduction, regulation of S phase of mitotic cell cycle | S | |
| YML060W | OGG1 | 43 kDa 8-oxo-guanine DNA glycosylase | purine-specific oxidized base lesion DNA N-glycosylase activity | base-excision repair, AP site formation, DNA repair | Late G1 | |
| YLR151C | PCD1 | coenzyme A diphosphatase | pyrophosphatase activity | biological_process unknown | S | |
| YGL055W | OLE1 | delta-9-fatty acid desaturase | stearoyl-CoA 9-desaturase activity | fatty acid desaturation, mitochondrion inheritance | Early G1 | |
| YMR076C | PDS5 | Precocious Dissociation of Sister chromatids | structural molecule activity | mitotic chromosome condensation, mitotic sister chromatid cohesion | Late G1 | |
| YLR233C | EST1 | Telomere elongation protein | telomerase activity, telomerase activity, single-stranded DNA binding, RNA binding | telomerase-dependent telomere maintenance | Late G1 | |
| YAR003W | SWD1 | compass (complex proteins associated with Set1p) component | transcriptional activator activity, chromatin binding, histone lysine N-methyltransferase activity (H3-K4 specific) | chromatin silencing at telomere, histone methylation | Late G1 | |
| YAR008W | SEN34 | tetrameric tRNA splicing endonuclease 34 kDa subunit | tRNA-intron endonuclease activity | tRNA splicing | S | |
| YDL103C | QRI1 | UDP-N-acetylglucosamine pyrophosphorylase | UDP-N-acetylglucosamine diphosphorylase activity | UDP-N-acetylglucosamine biosynthesis | Late G1 | |

The membership of the above three groups can be visualized in figure 4.10.5 for UPMGMACOR. As can be seen from the dendrogram, those genes responsible for DNA replication are mostly grouped together, though there are a few others of these genes scattered throughout the cluster.

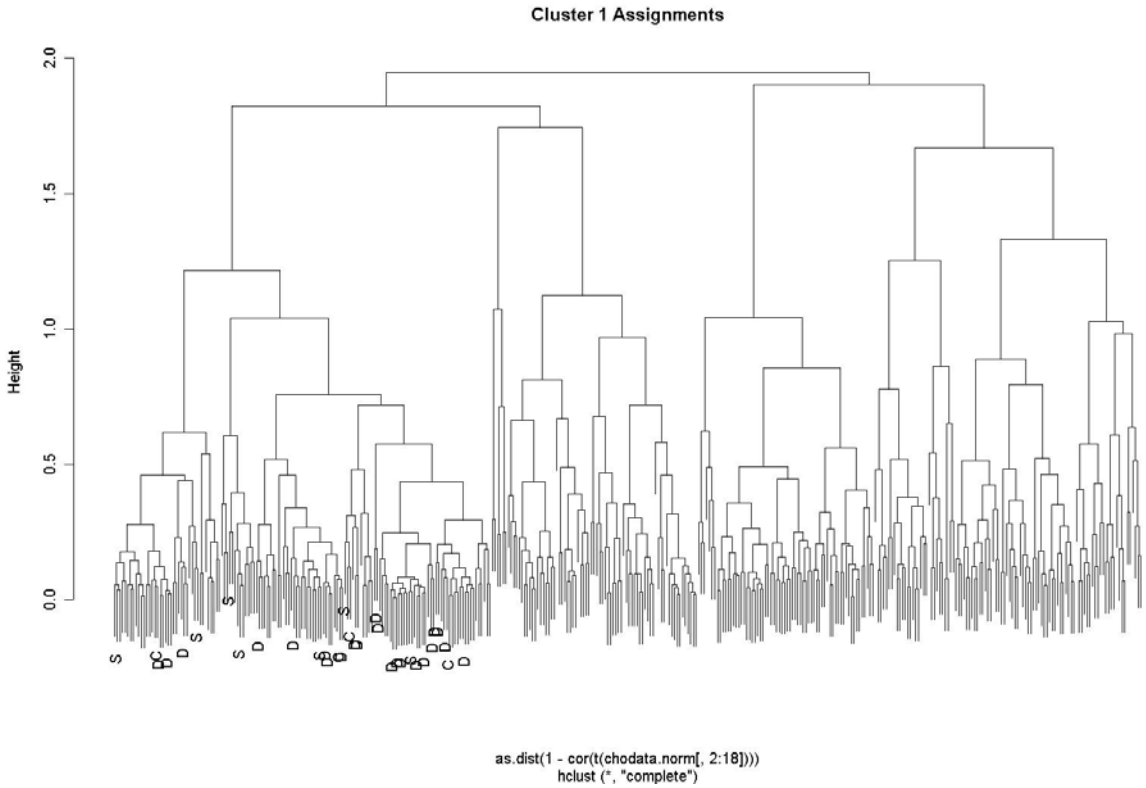



**Figure 4.10.5** Cluster 1 with functional annotations shown for UPGMACOR. D indicates genes used in gene replication, S indicates cytostructural genes, and C are Cyclin-dependent kinases.

The results of cluster 4 are interesting. As before, the composition of the analogous clusters can be seen in table 4.10.9. Note that each cluster contains at least 51 genes with M annotation. In addition, note that each cluster contains a number of Early G1 and G2 annotiations.

| Phase | UPGMACOR | UPGMAEUC | DIANAEUC |
|---|---|---|---|
| Early G1 | 12 | 12 | 16 |
| Late G1 | 0 | 0 | 2 |
| S | 6 | 1 | 0 |
| G2 | 24 | 13 | 12 |
| M | 52 | 51 | 52 |
| Total # | 94 | 77 | 82 |

**Table 4.10.9.** Composition of clusters analogous to UPGMACOR cluster 4.

The overlap can be seen in summarized form in the table below. 51 out of the 55 total genes with M annotation cluster together over these three methods.

| Phase | Overlap | # of Members |
|---|---|---|
| Early G1 | 12 | 67 |
| Late G1 | 0 | 135 |
| S | 1 | 75 |
| G2 | 10 | 52 |
| M | 51 | 55 |
| Total Genes | 74 | 384 |

**Table 4.10.10.** Overlap of clusters from the above three methods.

Table 4.10.11 shows the genes that remain in this cluster after the overlap has been calculated. Of these genes, several have related functions to cytokinesis, or cell division, which takes place in M phase. CLB1 and CLB2 are B-cyclins, which are involved with the G2/M transition. [40, 41] MYO1, MYO3, HOF, and IQG1 are necessary genes for the manufacture of the actomyosin ring, a complex necessary for cell division. [35] Morgan has discussed the role of CDC20 in breaking down CLB1 and CLB2 in order to allow for mitotic exit. [42] SRC1 and ASE1 are involved in the construction of the mitotic spindle. [38]

There are also a number of genes whose functions are not associated with M phase genes. McBride has noted that ACE2 and SWI5 are transcription factors that

regulate G1-specific genes. [32] CDC54 is necessary to start replication and synthesis of DNA, which is a function associated with the S phase. [43]

**Table 4.10.11.** Overlap of three methods for cluster 4.

| ORF | Symbol | Description | GO Molecular Function | GO Biological Process | Cho Annotation | Evidence |
|---|---|---|---|---|---|---|
| *Genes associated with cytokinesis* | | | | | | |
| YCL038C | MYO1 | class II myosin | microfilament motor activity | axial budding, response to osmotic stress, cytokinesis | M | M (Guertin et al. [35]) |
| YGR279C | MYO3 | myosin I | microfilament motor activity | polar budding, cell wall organization and biogenesis, response to osmotic stress, endocytosis, exocytosis | M | M (Guertin et al. [35]) |
| YNL053W | ALK1 | haspin | protein serine/threonine kinase activity | mitosis | M | M (Higgins et al. [44]) |
| YOR025W | CLB2 | B-type cyclin | cyclin-dependent protein kinase regulator activity | G2/M transition of mitotic cell cycle, G2/M transition of mitotic cell cycle, regulation of CDK activity | M | M (Ito, et al. [41], Richardson [40]) |
| YEL032W | CLB1 | B-type cyclin | cyclin-dependent protein kinase regulator activity | meiotic G2/MI transition, mitotic spindle assembly (sensu Fungi), G2/M transition of mitotic cell cycle, regulation of CDK activity | M | M (Ito, et al. [41], Richardson [40]) |
| YPR119W | CDC5 | protein kinase | protein serine/threonine kinase activity | DNA dependent DNA replication, DNA dependent DNA replication, protein amino acid phosphorylation, DNA dependent DNA replication | M | M (Ito, et al. [41], Song, et al. [45]) |
| YPR019W | CDC20 | anaphase promoting complex (APC) subunit | enzyme activator activity | cyclin catabolism, mitotic chromosome segregation, mitotic metaphase/anaphase transition, ubiquitin-dependent protein catabolism, mitotic spindle elongation | M | M (Morgan, et al. [42]) |
| YBR038W | HOF1 | Bud neck-localized, SH3 domain-containing protein required for cytokinesis; regulates actomyosin ring dynamics and septin localization; interacts with the formins, Bni1p and Bnr1p, and with Cyk3p, Vrp1p, and Bni5p | cytoskeletal protein binding | cytokinesis | M | M (Song, et al. [45]) |

**Table 4.10.11.** Overlap of three methods for cluster 4.

| ORF | Symbol | Description | GO Molecular Function | GO Biological Process | Cho Annotation | Evidence |
|---|---|---|---|---|---|---|
| YGR108W | IQG1 | Essential protein required for determination of budding pattern, promotes localization of axial markers Bud4p and Cdc12p and functionally interacts with Sec3p, localizes to the contractile ring during anaphase, member of the IQGAP family | cytoskeletal protein binding | cytokinesis, response to osmotic stress, actin filament organization | M | M (Song, et al. [45]) |
| YHR023W | SRC1 | Protein with a putative role in sister chromatid segregation, potentially phosphorylated by Cdc28p; green fluorescent protein (GFP)-fusion protein localizes to the nuclear periphery | molecular_function unknown | mitotic sister chromatid separation | M | M (Ubersax, et al. [38]) |
| YKL129C | ASE1 | spindle midzone component | microtubule binding | mitotic spindle assembly (sensu Fungi), mitotic anaphase B | M | M (Ubersax, et al. [38]) |
| YOR273C | MOB1 | Mps One Binder | kinase regulator activity | regulation of exit from mitosis, protein amino acid phosphorylation | G2 | M (Ubersax, et al. [38]) |
| YHL028W | CHS2 | chitin synthase 2 | chitin synthase activity | response to osmotic stress, cytokinesis | G2 | M (Ubersax, et al. [38]) |
| YBL023C | SPO12 | 20 kDa protein with negatively charged C-terminus required for function\|positive regulator of exit from M-phase in mitosis and meiosis (putative) | molecular_function unknown | meiosis I, mitotic cell cycle, regulation of exit from mitosis | M | M (Jensen, et al. [46]) |
| *Genes possibly misplaced* | | | | | | |
| YMR032W | TAF2 | TATA binding protein-associated factor | general RNA polymerase II transcription factor activity | G1-specific transcription in mitotic cell cycle, transcription initiation from Pol II promoter | M | G1 |
| YAL022C | ACE2 | zinc finger transcription factor | transcriptional activator activity | G1-specific transcription in mitotic cell cycle | M | Late G1 (McBride, et al. [32]) |
| YDL198C | SWI5 | transcriptional activator | transcriptional activator activity | G1-specific transcription in mitotic cell cycle | M | Late G1 (McBride, et al. [32]) |
| YHR152W | CDC54 | essential for initiation of DNA replication; homolog of S. pombe CDC21 | chromatin binding, ATP dependent DNA helicase activity | pre-replicative complex formation and maintenance, DNA replication initiation, DNA unwinding | Early G1 | S |

**Table 4.10.11.** Overlap of three methods for cluster 4.

| ORF | Symbol | Description | GO Molecular Function | GO Biological Process | Cho Annotation | Evidence |
|---|---|---|---|---|---|---|
| YML034W | MCM2 | Member of complex that acts at ARS's to initiate replication | chromatin binding, ATP dependent DNA helicase activity | pre-replicative complex formation and maintenance, DNA replication initiation, DNA unwinding | Early G1 | S |
| YIL162W | MCM3 | Member of complex that acts at ARS's to initiate replication | chromatin binding, ATP dependent DNA helicase activity | pre-replicative complex formation and maintenance, DNA replication initiation, DNA unwinding | Early G1 | S |
| *Other Genes* | | | | | | |
| YLR131C | WSC4 | contains novel cysteine motif\|integral membrane protein (putative)\|similar to SLG1 (WSC1), WSC2 and WSC3 | transmembrane receptor activity | actin cytoskeleton organization and biogenesis, response to heat, Rho protein signal transduction, NA, cell wall organization and biogenesis | M | |
| YGL021W | MSG5 | protein tyrosine phosphatase | prenylated protein tyrosine phosphatase activity | adaptation to pheromone during conjugation with cellular fusion | M | |
| YOR058C | ATG22 | Autophagy gene essential for breakdown of autophagic vesicles in the vacuole | molecular_function unknown | autophagy, protein-vacuolar targeting, protein-vacuolar targeting | M | |
| YBR200W | SKN1 | highly homologous to Kre6p\|type II membrane protein (putative) | glucosidase activity, glucosidase activity | beta-1,6 glucan biosynthesis, beta-1,6 glucan biosynthesis, cell wall organization and biogenesis | M | |
| YGR230W | BUD4 | Protein involved in bud-site selection and required for axial budding pattern; localizes with septins to bud neck in mitosis and may constitute "axial landmark" for next round of budding; potential Cdc28p substrate | GTP binding | bud site selection, axial budding | M | |
| YJR092W | SAC7 | GTPase activating protein (GAP) for RHO1 | Rho GTPase activator activity, signal transducer activity | cell cycle dependent actin filament reorganization, small GTPase mediated signal transduction | G2 | |
| YLR353W | BEM1 | SH3-domain protein that binds Cdc24p, Ste5p and Ste20p, and the Rsr1p/Bud2p/Bup5p GTPase | protein binding | cellular morphogenesis during conjugation with cellular fusion, establishment of cell polarity (sensu Saccharomyces) | Early G1 | |
| YGL116W | HST3 | Homolog of SIR2 | DNA binding | chromatin silencing at telomere, short-chain fatty acid metabolism | M | |
| YBR202W | KIN3 | protein kinase | protein kinase activity | chromosome segregation | M | |
| YMR001C | SCW4 | soluble cell wall protein | glucosidase activity | conjugation with cellular fusion | M | |

**Table 4.10.11.** Overlap of three methods for cluster 4.

| ORF | Symbol | Description | GO Molecular Function | GO Biological Process | Cho Annotation | Evidence |
|---|---|---|---|---|---|---|
| YGR092W | MCM6 | Member of the MCM/P1 family of proteins involved in DNA replication | chromatin binding, ATP dependent DNA helicase activity | DNA replication initiation, DNA replication initiation, DNA unwinding | M | |
| YPL242C | ZRT1 | High-affinity zinc transporter of the plasma membrane, responsible for the majority of zinc uptake; transcription is induced under low-zinc conditions by the Zap1p transcription factor | high affinity zinc uptake transporter activity | high-affinity zinc ion transport | G2 | |
| YAR018C | BNS1 | Protein with some similarity to Spo12p; overexpression bypasses need for Spo12p, but not required for meiosis | molecular_function unknown | meiosis | M | |
| YGL201C | NUF2 | Spindle pole body protein, required for chromosome segregation during mitosis; part of a centromeric complex along with Tid3p, Spc25p, and Spc24p | structural constituent of cytoskeleton | microtubule nucleation, chromosome segregation | M | |
| YIL106W | GGC1 | Member of the mitochondrial carrier family, displays GTP/GDP transport activity in vitro; mutant phenotypes suggest a role in mitochondrial iron transport and/or GTP and GDP transport; essential for mitochondrial genome maintenance | transporter activity | mitochondrial genome maintenance, transport, iron ion homeostasis | G2 | |
| YOL069W | SHE2 | RNA-binding protein that binds specific mRNAs and interacts with She3p; part of the mRNA localization machinery that restricts accumulation of certain proteins to the bud | mRNA binding | mRNA localization, intracellular | M | |
| YPL058C | DBF2 | Kinase required for late nuclear division. Cdc15 promotes the exit from mitosis by directly switching on the kinase activity of Dbf2. | protein kinase activity | nuclear division, protein amino acid phosphorylation | M | |
| YDL138W | FUN26 | Nucleoside transporter with broad nucleoside selectivity; localized to intracellular membranes | nucleoside transporter activity | nucleoside transport | M | |

**Table 4.10.11.** Overlap of three methods for cluster 4.

| ORF | Symbol | Description | GO Molecular Function | GO Biological Process | Cho Annotation | Evidence |
|---|---|---|---|---|---|---|
| YDR389W | PDR12 | multidrug resistance transporter | organic acid transporter activity, xenobiotic-transporting ATPase activity | organic acid transport, propionate metabolism, transport | Early G1 | |
| YKL130C | TPO4 | Polyamine transport protein | spermidine transporter activity, spermine transporter activity | polyamine transport | Early G1 | |
| YGR143W | CDC47 | Component of the hexameric MCM complex, which is important for priming origins of DNA replication in G1 and becomes an active ATP-dependent helicase that promotes DNA melting and elongation when activated by Cdc7p-Dbf4p in S-phase | chromatin binding, ATP dependent DNA helicase activity, ATP binding | pre-replicative complex formation and maintenance, DNA replication initiation, DNA unwinding | Early G1 | |
| YDR146C | UTP4 | U3 snoRNP protein | snoRNA binding | processing of 20S pre-rRNA, processing of 20S pre-rRNA | G2 | |
| YCR042C | BUD8 | Protein involved in bud-site selection; diploid mutants display a unipolar budding pattern instead of the wild-type bipolar pattern, and bud at the proximal pole | molecular_function unknown | pseudohyphal growth, bud site selection | M | |
| YDR324C | WTM2 | transcriptional modulator | transcription corepressor activity | regulation of meiosis | M | |
| YOR229W | RGT2 | glucose receptor | glucose transporter activity, glucose binding, receptor activity | signal transduction, response to glucose stimulus | M | |
| YBR104W | SUC2 | invertase (sucrose hydrolyzing enzyme) | beta-fructofuranosidase activity | sucrose catabolism | M | |
| YGL255W | YMC2 | Putative mitochondrial inner membrane transporter, member of the mitochondrial carrier (MCF) family | transporter activity | transport | G2 | |

## 4.11 Conclusions and Lessons Learned

There are a number of lessons to be learned from running the tool on both the simulated and Cho datasets.

First of all, the tool is useful at finding consensus between clustering methods. This was seen for both within-method and between-method metrics for the simulated datasets, and for the within-method metrics on the Cho-dataset. The between-method metrics agree with each other, but show different similarities than the within-method metrics.

Secondly, as shown with the Cho-dataset, the tool is useful in finding stable clusterings across methods. This is especially apparent when the methods examined are compared by overlapping analogous clusters. Currently, the overlaps were calculated by hand inspection of the clusterings. Future functionality of the framework should be able to automate this overlap by calculating the best match of a cluster across methods.

Lastly, the tool seems to be useful on real microarray data. It remains to be seen, however, how useful the tool is in finding consensus on microarray data with less obvious geometry. However, it should be noted that in this study we have robbed hierarchical methods of one of their potential strengths by specifying the number of clusters, namely that they are a good exploration tool for when the geometry is not specified. A further, more realistic study would be to specify the geometry to only k-means and SOM, and let the hierarchical methods run without specification. R has facilities for visually choosing clusters from hierarchical methods, and these could be easily implemented in a later version of the framework.

Also, there are a number of difficulties with working with this dataset that should be noted. The genes were chosen by visual inspection from a larger subset of genes. This dataset is similar to datasets that have been filtered by other means, and so results on this dataset should apply to other filtered datasets. Also, there appears to be only 3 actual clusters in the data. The functional annotations do point at least to one strong clustering, cluster 1, with genes involved in DNA replication. Cluster 4 does appear to have genes associated with mitotic division.

# *Chapter 5: Future Directions*

## 5.1 Visualization component

In addition to the numerical portion of the tool discussed above, a further extension that will be especially useful for researchers is a component that will allow them to visually compare clusterings. For example, given a set of different dendrograms, it will be useful to be able to track if gene X stays with gene Y across the clusterings (see figure below).

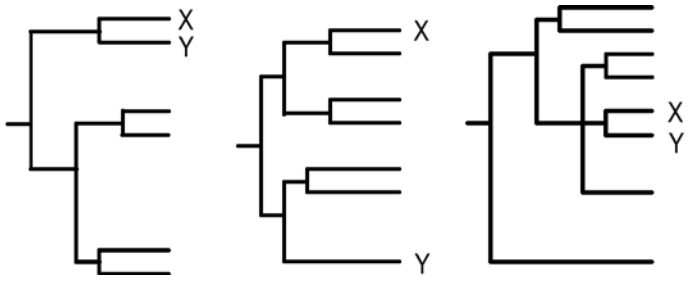



**Figure 5.1**. Three dendrograms. In the first, gene X clusters with gene Y. In the second, they are apart. In the third, the two genes cluster together again.

A further extension will allow researchers to visualize exactly where the clusterings overlap, much as shown in section 4.10. Of course, tools to find analogous clusters need to be developed, as they may not be as apparent as they were with the Cho dataset.

## 5.2 Research Using the Evaluation Framework

Additionally, we would like to utilize this tool to address key questions in bioinformatics research with respect to the impact of pre-processing on clustering results. Briefly, preprocessing of microarray data is done in order to reduce the effects of noise on further analysis. The impact that types of normalization (statistical protocols to remove systematic variation), background subtraction, and filtering have on clustering has not been addressed. Since clustering methods are sensitive to noise, we speculate that the effect of such preprocessing methods may have a greater impact on the final clusterings than the clustering methods themselves. We suspect that this is the case because a rigorous preprocessing protocol can reduce the number of noisy genes to such an extent that most methods would perform equally well.

## 5.3 Further extensions of this tool

Further extensions to this tool include implementing metrics to assess the reliability of clustering methods. Many of these methods rely on perturbing the original dataset with noise and comparing the clusterings of the noisy dataset with that of the original dataset. [47] Another tactic is to use the "leave one out" approach of leaving out a technical replicate and compare the clustering with the original dataset's clustering. [22, 48]

A final extension to the tool would be to implement a metric that assesses the biological, or functional, significance of a clustering. Jakt has suggested a method for measuring the functional significance by measuring the probability that elements of a cluster

match a specific functional motif. [49] This method is possible because genes that are similarly expressed tend to be regulated by the same transcription factors. Such a metric would increase the utility of this tool to biologists.

## 5.4 Deployment/Beta Testing

Another step in the development of this tool would be to deploy it to researchers for beta testing and further feedback. Such a step provides valuable input for the improvement of the interface. This can be done at OHSU in conjunction with the OHSU Gene Microarray Shared Resource, which has a large base of researchers utilizing both spotted and Affymetrix platforms.

# *Appendix A: Code*

## Main Body of Code

```
##testMain produces an object called report
##report is available as results$report
##clusterings are available as results$clusters
##params are available as results$params
testMain <- function(data) {

        library(cluster)
        ##grab list of methods and associated parameters
        methodlist <- getClusterInfo()
        ##produce clusterings as clusterlist
        clusterlist <- clusterList(data, methodlist)
        ##run within-method metrics
        within <- withinmetrics(clusterlist)
        ##run between-method metrics
        between <- betweenmetrics(clusterlist)
        ##initialize objects
        clusters <- list()
        params <- list()
        for(i in 1:length(clusterlist)){
                clusters[[i]] <- as.matrix(as.factor(clusterlist[[i]]$clustering[,1]))
                row.names(clusters[[i]]) <- row.names(clusterlist[[i]]$clustering)
                params[[i]] <- clusterlist[[i]]$params
                }
        results <- list(report=list(within = within, between = between),
                clusters=clusters, params=params)
        results

}


clusterList <- function(data, methodlist) {
        ##clusterList takes data and a methodlist and
        ##outputs a series of clusterings

   n <- length(methodlist)
   if(n > 0 && is.list(methodlist)){

        clustlist <- list()
        for(i in 1:n)
                {
                  params <- as.list(methodlist[[i]])
                  ##initialize output function
                  func <- function(data, params){}
                  ##grab appropriate output function from list
                  func <- methodlist[[i]]$func

                  ##do the clustering with proper output
                  clustering <- func(data, methodlist[[i]])

                  ## put output in clustering list
                  clustlist[[i]] <- list(clustering = clustering, params = params)

                }

        clustlist
              }

   else{stop("improper input to clusterlist")}
```

```
}


##withinmetrics calculates the metrics for each cluster
##produces a table with all metrics in a report
withinmetrics <- function(clusterlist) {

        #list of metrics here
        withinnames <- c("n-clusters", "homogeneity", "separation", "silhouette")
        row <- length(clusterlist)
        col <- length(withinnames)

        withinresult <- matrix(ncol = col, nrow = row)
        colnames(withinresult) <- withinnames
        methodnames <- vector()

        #calculate correlation distance matrix
        #needed for silhouette index calculation
        data <- clusterlist[[1]]$clustering
        data <- data[,2:ncol(data)]
        cordist <- as.dist(1 - cor(t(data)))

        for(i in 1:length(clusterlist)){
                clustering <- clusterlist[[i]]$clustering

                #grab clustering id
                id <- clusterlist[[i]]$params$id
                n <- max(clustering[,1])
                #calcuate metrics
                #calculate silhouette value
                sil <- silhouette(clustering[,1], cordist)
                #return average silhouette value
                summ <- summary(sil)

                avgsil <- as.numeric(summ$avg.width)
                if(length(avgsil)>1)
                        avgsil <- avgsil[1]
                if(length(avgsil)==0)
                        avgsil <- 0

                #sort clustering by assignment
                clustering <- clustering[order(clustering[,1]),]
                #calculate cluster centers
                centers <- clusterCenters(clustering)

                #grab number of clusters
                #calculate separation
                separation <- separation(clustering, centers)
                homogeneity <- homogeneity(clustering, centers)

                #print(c(n, separation, homogeneity, avgsil))
                withinresult[i,] <- c(n, homogeneity, separation, avgsil)
                methodnames[i] <- id
        }

        rownames(withinresult) <- methodnames
        withinresult
}

betweenmetrics <- function(clusterlist) {

        #list of between-methods
        between <- c("jaccard", "variationinformation")
        n <- length(clusterlist)

        ##initialize results matrix
        jaccardresult <- matrix(ncol = n, nrow = n)
        variationresult <- matrix(ncol = n, nrow = n)

        methodnames1 <- vector()
```

```
        methodnames2 <- vector()

        for(i in 1:n){
                clustering1 <- clusterlist[[i]]$clustering
                id1 <- clusterlist[[i]]$params$id
                methodnames1[i] <- id1
                for(j in 1:n){

                        id2 <- clusterlist[[j]]$params$id
                        methodnames2[j] <- id2
                        if(i>j){
                                clustering2 <- clusterlist[[j]]$clustering


                                jacc <- jaccard(clustering1, clustering2)
                                jaccardresult[i,j] <- jacc


                                variationinformation <-
                                        variationInformation(clustering1, clustering2)
                                variationresult[i,j] <- variationinformation
                         }
                }
        }

        colnames(jaccardresult) <- methodnames1
        rownames(jaccardresult) <- methodnames2

        colnames(variationresult) <- methodnames1
        rownames(variationresult) <- methodnames2

        clusters <- list()


        betweenresult <- list(jaccardresult=as.dist(jaccardresult),
                variationresult=as.dist(variationresult))

        betweenresult
}
```

## Parameter/List of Methods Input

```
##getClusterInfo utilizes a simple GUI
##to grab a list of Methods
##and appropriate parameters for each parameter
##notes: widgetTools is really buggy
##labels don't show for each input box
##This isn't very user-friendly as is.

getClusterInfo <- function() {

        require(widgetTools)

        pwidth <- 3
        PWEnv <- new.env(hash = TRUE, parent = parent.frame(1))

        upgma <- checkButton(wName = "upgma", wValue = c(upgma = TRUE),
                wEnv = PWEnv, wWidth = pwidth)
        upgma.distfunc <- radioButton(wName = "upgma.dist",
                wValue = c(euclidean = TRUE, correlation = FALSE),
                wEnv = PWEnv, wWidth = pwidth)
        upgma.height <- entryBox(wName = "upgma.height",
                wValue = c(Height = 2), wEnv  = PWEnv, wWidth = pwidth)

        diana <- checkButton(wName = "diana", wValue = c(diana = TRUE),
                wEnv = PWEnv)
        diana.distfunc <- radioButton(wName = "diana.dist",
                wValue = c(euclidean = TRUE, correlation = FALSE),
```

```
        wEnv = PWEnv, wWidth = pwidth)
   diana.height <- entryBox(wName = "diana.height",
        wValue = c(Height = 2), wEnv  = PWEnv, wWidth = pwidth)

   som <- checkButton(wName = "som", wValue = c(som = TRUE),
        wEnv = PWEnv)
   som.xdim <- entryBox(wName = "som.xdim", wValue = c(xdim = 0),
               wEnv  = PWEnv, wWidth = pwidth)
   som.ydim <- entryBox(wName = "som.ydim", wValue = c(ydim = 0),
               wEnv   = PWEnv, wWidth = pwidth)

   kmeans <- checkButton(wName = "kmeans", wValue = c(kmeans = TRUE),
        wEnv = PWEnv)
   kmeans.k <- entryBox(wName = "k", wValue = c(k = 0), wEnv  = PWEnv,
        wWidth = pwidth)
   kmeans.iterations <- entryBox(wName = "iterations", wValue = c(num=10000),
        wEnv = PWEnv, wWidth = pwidth)


   pWidgets <- list(upgma = list(upgma = upgma, height = upgma.height,
                              distfunc = upgma.distfunc),
                    diana = list(diana = diana, height = diana.height,
                              distfunc = diana.distfunc),
                    som = list(som = som, xdim = som.xdim, ydim = som.ydim),
                    kmeans = list(kmeans = kmeans, k = kmeans.k,
                              iterations = kmeans.iterations)
                       )

   if(interactive()) {
   methodWidget <- widget(wTitle = "Select Your Methods",
        pWidgets, funs = list(), env = PWEnv)

   i <- 1
   methodlist <- list()

        }

if(wValue(pWidgets(methodWidget)[["upgma"]][["upgma"]]) == TRUE) {
         id <- "UPGMAEUC"
         func <- agglomOutput
         method <- "upgma"
         clustMethod <- "average"
         distfunc <- "euclidean"
                     height <-
         as.numeric(wValue(pWidgets(methodWidget)[["upgma"]][["height"]]))
         params <- list(id = id, method=method, func=func,
                distfunc = distfunc, height = height,
                clustMethod = clustMethod)
         methodlist[[i]] <- params
         i <- i + 1
         }

   if(wValue(pWidgets(methodWidget)[["upgma"]][["upgma"]]) == TRUE) {
         id <- "UPGMACOR"
         func <- agglomOutput
         method <- "upgma"
         clustMethod <- "average"

         #return proper distance
         distfunc <- "correlation"

         height <-
                as.numeric(wValue(pWidgets(methodWidget)[["upgma"]][["height"]]))
         params <- list(id = id, method=method, func=func,
                distfunc = distfunc, height = height,
                clustMethod = clustMethod)
         methodlist[[i]] <- params
         i <- i + 1
         }

   if(wValue(pWidgets(methodWidget)[["diana"]][["diana"]]) == TRUE) {
```

```
                id <- "DIANAEUC"
                func <- dianaOutput
                method <- "diana"
                distfunc <- "euclidean"
                height <-
                        as.numeric(wValue(pWidgets(methodWidget)[["diana"]][["height"]]))
                params <- list(id = id, method=method, func=func, distfunc = distfunc,
                        height = height)
                methodlist[[i]] <- params
                i <- i + 1
                }


        if(wValue(pWidgets(methodWidget)[["diana"]][["diana"]]) == TRUE) {
                id <- "DIANACOR"
                func <- dianaOutput
                method <- "diana"
                distfunc <- "correlation"
                height <-
                        as.numeric(wValue(pWidgets(methodWidget)[["diana"]][["height"]]))
                params <- list(id = id, method=method, func=func, distfunc = distfunc,
                        height = height)
                methodlist[[i]] <- params
                i <- i + 1
                }


        if(wValue(pWidgets(methodWidget)[["som"]][["som"]]) == TRUE) {
                id <- "SOM1"
                func <- somOutput
                method <- "som"
                distfunc <- "euclidean"
                xdim <- as.numeric(wValue(pWidgets(methodWidget)[["som"]][["xdim"]]))
                ydim <- as.numeric(wValue(pWidgets(methodWidget)[["som"]][["ydim"]]))
                params <- list(id=id, method=method, func=func, xdim=xdim, ydim=ydim,
                         distfunc=distfunc)
                methodlist[[i]] <- params
                i <- i + 1
                }

        if(wValue(pWidgets(methodWidget)[["kmeans"]][["kmeans"]]) == TRUE) {
                id <- "KMEANS1"
                func <- kmeansOutput
                method <- "kmeans"
                distfunc <- "euclidean"
                k <- as.numeric(wValue(pWidgets(methodWidget)[["kmeans"]][["k"]]))
                iterations <- as.numeric(wValue(pWidgets(methodWidget)[["kmeans"]]
                        [["iterations"]]))
                params <- list(id = id, method=method, func=func, k=k,
                        iterations = iterations, distfunc=distfunc)
                methodlist[[i]] <- params
                i <- i + 1
                }

methodlist
}
```

## Output Functions

```
somOutput <- function(matrix, params) {

        require(som)
        if(params$method == "som"){
        xdim <- params$xdim
        ydim <- params$ydim}

        else {stop("improper input to method")}
```

```
        ##somvector takes input from som$visual
        ##(x,y coords of cluster) and converts
        ##that to a cluster number
        somvector <- function(somnum, xdim, ydim) {
                n <- nrow(somnum)
                ##validate input
                ##if not validated, stop
                if(ncol(somnum) == 2)
                {
                        ##initialize cluster number vector
                        somvector <- vector(length=(1:n))

                        for(i in 1:nrow(somnum))
                                { somvector[i] <-      somnum[i,1] + somnum[i,2] +
                                        (somnum[i,1] * (ydim-1)) + 1
                                }

                        ##output vector
                        somvector
                }

                else { stop("Improper Input to somInput.")}

                }

#main body of function
if(is.matrix(matrix))
        {
        ##perform som
        som <- som(matrix, xdim, ydim)
        ##convert coords to cluster numbers
        somvec <- somvector(som$visual[,1:2], xdim, ydim)
        ##produce output file

        somvec <- as.matrix(somvec)
        output <- as.data.frame(cbind(somvec, matrix))

        output
        }

else {
        stop("Input is not a matrix")}
}


agglomOutput <- function(matrix, params) {

                clustMethod <- params$clustMethod
                distfunc <- params$distfunc
                k <- params$height

        ##check to see if input is a matrix
        ##otherwise, stop
        if(is.matrix(matrix)) {

                ##calculate appropriate distance matrix
                distmatrix <- switch(distfunc, euclidean = dist(matrix),
                        correlation = as.dist(1 - cor(t(matrix))))

                ##do clustering
                tree <- hclust(distmatrix, method = clustMethod)

                ##cut tree at selected height
                tmp <- cutree(tree, k=k)

                ##output data file with cluster info
                output <- as.data.frame(cbind(tmp, matrix))

                ##output file
                output
```

```
                }

        else{
        stop("Input is not a matrix.")
        }
}

dianaOutput <- function(matrix, params) {

        require(cluster)

                clustMethod <- params$clustMethod
                distfunc <- params$distfunc
                k <- params$height

        ##check to see if input is a matrix
        ##otherwise, stop
        if(is.matrix(matrix)) {

                ##calculate appropriate distance matrix
                distmatrix <- switch(distfunc, euclidean = dist(matrix),
                        correlation = as.dist(1 - cor(t(matrix))))

                ##do clustering
                tree <- diana(x = distmatrix, diss = TRUE)

                ##cut tree at selected height
                tmp <- cutree(as.hclust(tree), k=k)

                ##output data file with cluster info
                output <- as.data.frame(cbind(tmp, matrix))

                ##output file
                output

                }

        else{
        stop("Input is not a matrix.")
        }
}


kmeansOutput <- function(matrix, params) {

        if(params$method == "kmeans") {
                        k <- params$k
                        iterations <- params$iterations
        }

        else {stop("improper params")}

        ## checks input to see if it's a matrix
        ## otherwise, stop
        if(is.matrix(matrix)) {

                ##do k-means clustering
                tmp <- kmeans(matrix, k, iterations)

                ##produce output file
                output <- as.data.frame(cbind(tmp$cluster, matrix))

                ##output file
                output
        }

        else {
        stop("Input is not a matrix.")}

}
```

## Metric Code

```
##Within method metrics here

##clusterCenters calculates centers of clusters
##given standard output format of clusters
##grab row length of matrix

clusterCenters <- function(x) {

n <- ncol(x)
max <- max(x[1])

if(is.data.frame(x)) {

        ##calculate vector centers of each cluster
        agg <- aggregate(x[,2:n], list(cluster=x[,1]), mean)

        ##initialize matrix for centers
        centers <- matrix(0, max, n-1)

        p <- ncol(agg)

        ##transfer cluster info to matrix
        for (i in 1:nrow(agg))
           {centers[as.numeric(levels(agg$cluster))[i],] <-
                       as.matrix(agg[i,2:ncol(agg)])
           }
                ##grabs vector info out of list

        ##return matrix of cluster centers
        centers
        }

else { stop("input is not a data frame")}

}


##homogeneity takes both standard clustering output
##and cluster centers as input
homogeneity <- function(clustering, centers, distfunc="euclidean") {

        ##initialize count
        tally <- c(0)

        euclidean <- function(clustering, centers) {

                for(i in 1:nrow(clustering)) {
                tally <- tally + sqrt(sum((clustering[i, c(2:ncol(clustering))]
                               - centers[clustering[i,1],])^2)) }
                }

        correlation <- function (clustering, centers) {

                for(i in 1:nrow(clustering)) {
                tally <- tally + (1-cor(clustering[i, c(2:ncol(clustering))],
                               centers[clustering[i,1],]))
                               }
                       }

        tally <- switch(distfunc, euclidean=euclidean(clustering, centers),
                       correlation = correlation(clustering, centers))

        average <- tally / nrow(data)

        average

}
```

```
#separation takes both clustering output
##and cluster centers as input
separation <- function(clustering, centers) {
if(is.matrix(centers))
        {

        countcluster <- function(output) {
                countclust <- as.vector(table(output[,1]))
                countclust}

        sepmatrix <- dist(centers)

        # L is number of clusters
        L <- nrow(centers)

        ##if number of clusters is >= 2
        ##calculate separation
        if(L >= 2) {

                countclust <- countcluster(clustering)
                colclust <-    as.matrix(countclust[2:L])
                rowclust <- t(countclust[1:(L-1)])
                prod <- colclust %*% rowclust

                sepmatrix <- sepmatrix * prod

                separation <- sum(sepmatrix)/sum(prod)
        }

        ##if number of clusters is 1, return 0
        else {separation <- 0}

        separation
        }
else{
stop("input is not a matrix")}
}


##Between-method Metrics Here

##contingencytable returns a contingency table
##given two clusters grouped by cluster assignment
contingencytable <- function(by1, by2)
{
        ##function intersect returns number in
        ##intersection between sets x and  y
        intersect <- function(x, y) length(y[match(x, y, nomatch = 0)])

        contingency <- matrix(nrow=length(by1), ncol=length(by2))

        for(i in 1:length(by1))
                {for(j in 1:length(by2))
                        {contingency[i,j] <- intersect(by1[[i]], by2[[j]])}
                }
        contingency
}

jaccard <- function(clustering1, clustering2)
{
if(is.data.frame(clustering1) && is.data.frame(clustering2)
                && (nrow(clustering1) == nrow(clustering2)))
        {

        ##separate the data frame names by their individual clusters
        by1 <- by(clustering1, clustering1[1],
                        function(x) {row.names(x)})

        by2 <- by(clustering2, clustering2[1],
                        function(x) {row.names(x)})
```

```
        contingency <- contingencytable(by1, by2)

        Z <- sum(contingency^2)
        n <- nrow(clustering1)

        sumsquarerow <- sum(colSums(contingency)^2)
        sumsquarecol <- sum(rowSums(contingency)^2)

        jaccard <- (Z - n) / ( sumsquarerow + sumsquarecol - Z - n)

        jaccard

        }

else  {stop("Improper input to function")}

}



variationInformation <- function(clustering1, clustering2) {
        ##function intersect returns number in
        ##intersection between sets x and  y
        intersect <- function(x, y) length(y[match(x, y, nomatch = 0)])

n <- nrow(clustering1)

if(is.data.frame(clustering1) && is.data.frame(clustering2))
        {

        ##separate the data frame names by their individual clusters
        by1 <- by(clustering1, clustering1[1],
                        function(x) {row.names(x)})

        by2 <- by(clustering2, clustering2[1],
                        function(x) {row.names(x)})

        prob1 <- countcluster(clustering1)/n
        prob2 <- countcluster(clustering2)/n

        contingency <- contingencytable(by1, by2)

        jointprob <- contingency/n

        probproduct <- as.matrix(prob1) %*% t(prob2)

        logprob <- ifelse(contingency==0,0,log10(jointprob/probproduct))

        mutual <- sum(jointprob * logprob)

        entropy1 <- -sum(prob1 * log10(prob1))
        entropy2 <- -sum(prob2 * log10(prob2))

        variation <- entropy1 + entropy2 - (2*mutual)

        variation

        }

else  {stop("improper input to function")}
}
```

## Data simulation code:

The following script is an example of the code used to generate and test the simulated datasets.

```
##script to generate simulated dataset
##Assume S/N ratio of around 3
##Parameters
datamean <- 3.0
datadev <- 1.0
##noise mean is centered around 0
noisedev <- 0.05

##generate cluster1 - descending
clusterseeds1 <- rnorm(250, mean=datamean, sd=datadev)
##pattern is outer-multiplied by clusterseeds1
##to form correlated data
pattern1 <- c(4, 3, 2, 1)
cluster1 <- as.matrix(clusterseeds1) %*% t(pattern1)
##generate noise
noisematrix <- matrix(ncol=4, nrow=250)
for(i in 1:4)
       {
       noise <- rnorm(250, mean=0, sd=noisedev)
       noisematrix[,i] <- noise
       }
##generate assignment vector
assign1 <- vector(length=250) + 1
##generate cluster by adding cluster1 to noisematrix
cluster1 <- cbind(as.matrix(assign1),(cluster1 + noisematrix))


##generate cluster2 - ascending
clusterseeds2 <- rnorm(250, mean=datamean, sd=datadev)
pattern2 <- c(1, 2, 3, 4)
cluster2 <- as.matrix(clusterseeds2) %*% t(pattern2)
noisematrix <- matrix(ncol=4, nrow=250)
for(i in 1:4)
       {
       noise <- rnorm(250, mean=0, sd=noisedev)
       noisematrix[,i] <- noise
       }

assign2 <- vector(length=250) + 2
cluster2 <- cbind(as.matrix(assign2),(cluster2 + noisematrix))


##generate cluster3 - up then down
clusterseeds3 <- rnorm(250, mean=datamean, sd=datadev)
pattern3 <- c(2, 4, 2, 1)
cluster3 <- as.matrix(clusterseeds3) %*% t(pattern3)
noisematrix <- matrix(ncol=4, nrow=250)
for(i in 1:4)
       {
       noise <- rnorm(250, mean=0, sd=noisedev)
       noisematrix[,i] <- noise
       }
assign3 <- vector(length=250) + 3
cluster3 <- cbind(as.matrix(assign3),(cluster3 + noisematrix))


##generate cluster4 - down then up
clusterseeds4 <- rnorm(250, mean=datamean, sd=datadev)
pattern4 <- c(2, 1, 2, 4)
cluster4 <- as.matrix(clusterseeds4) %*% t(pattern4)
noisematrix <- matrix(ncol=4, nrow=250)
for(i in 1:4)
       {
       noise <- rnorm(250, mean=0, sd=noisedev)
       noisematrix[,i] <- noise
```

```
        }
assign4 <- vector(length=250) + 4
cluster4 <- cbind(as.matrix(assign4),(cluster4 + noisematrix))


##bind clusters together
simdata2 <- rbind(cluster1, cluster2, cluster3, cluster4)
##shuffle the data
simdata2 <- simdata2[sample(1000),]
##assign rownames to matrix
row.names(simdata2) <- as.character(c(1:1000))
simdata2.true <- as.factor(simdata2[,1])

##run clusterings and metrics
simdata2.clusters <- testMain(simdata2[,2:5])
##generate report
simdata2.clusters$report

##generate jaccard index tree
simdata2.jaccard <- simdata2.clusters$report$between$jaccardresult
plot(hclust(1-simdata.jaccard), main="Jaccard Index")

##generate variation index tree
simdata2.variation <- simdata2.clusters$report$between$variationresult
plot(hclust(simdata2.variation), main="Variation of Information")

##generate confusion matrices
simtable2 <- list()
library(e1071)
for(i in 1:length(simdata2.clusters$clusters)){
        print(simdata2.clusters$params[[i]]$id)
        print("True Clustering")
        print(table(simdata2.true))
        print("Method Clustering")
        print(table(simdata2.clusters$clusters[[i]]))
        simdata2.order <-
                simdata2.clusters$clusters[[i]]
                        [order(as.numeric(row.names(simdata2.clusters$clusters[[i]])))]
        simtable2[[i]] <- table(simdata2.true, simdata2.order)
        class.match <- matchClasses(as.matrix(simtable2[[i]]),method="exact")
        print(simtable2[[i]][,class.match])

}
```

## Overlap of Clusterings

The following is a script that was written to calculate the overlaps for three sets of clusters, belonging to UPGMACOR, UPGMAEUC, and DIANAEUC.

```
##grab clusterings and associate them with functional annotation
chonorm.upgmaeuc <- cbind(as.matrix(chonorm.clusters$clusters[[1]]), as.matrix(chonorm.true))
chonorm.upgmacor <- cbind(as.matrix(chonorm.clusters$clusters[[2]]), as.matrix(chonorm.true))
chonorm.dianaeuc <- cbind(as.matrix(chonorm.clusters$clusters[[3]]), as.matrix(chonorm.true))

##group clusterings by method assignment
chonorm.upgmaeuc.by <- by(chonorm.upgmaeuc, chonorm.upgmaeuc[,1], function(x){x})
chonorm.upgmacor.by <- by(chonorm.upgmacor, chonorm.upgmacor[,1], function(x){x})
chonorm.dianaeuc.by <- by(chonorm.dianaeuc, chonorm.dianaeuc[,1], function(x){x})

##intersect returns a vector of length x of matches of x in y
intersect <- function(x, y) {match(x, y, nomatch = 0)}

##for cluster with mostly annotation LateG1
##calculate intersection between UPGMAEUC and UPGMACOR
inter <- intersect(row.names(chonorm.upgmaeuc.by[[1]]), row.names(chonorm.upgmacor.by[[1]]))

##show intersection of UPGMAEUC and UPGMACOR
chonorm.inter <- chonorm.upgmacor.by[[1]][inter,]

##calculate intersection between above intersection and DIANACOR
inter <- intersect(row.names(chonorm.inter), row.names(chonorm.dianaeuc.by[[1]]))

##show intersection of UPGMAEUC, UPGMACOR, and DIANACOR
chonorm.inter <- chonorm.dianaeuc.by[[1]][inter,]

##write results to a table
write.table(chonorm.inter, "chonorm-matcheslateg1.txt")


##for cluster with mostly annotation EarlyG1
##calculate intersection between UPGMAEUC and UPGMACOR
inter <- intersect(row.names(chonorm.upgmaeuc.by[[2]]), row.names(chonorm.upgmacor.by[[2]]))

##show intersection of UPGMAEUC and UPGMACOR
chonorm.inter <- chonorm.upgmacor.by[[2]][inter,]
chonorm.inter

##calculate intersection between above intersection and DIANACOR
inter <- intersect(row.names(chonorm.inter), row.names(chonorm.dianaeuc.by[[2]]))

##show intersection of UPGMAEUC, UPGMACOR, and DIANACOR
chonorm.inter <- chonorm.dianaeuc.by[[2]][inter,]
chonorm.inter

##write table
write.table(chonorm.inter, "chonorm-matchesearlyg1.txt")


##for cluster with mostly annotation M
##calculate intersection between UPGMAEUC and UPGMACOR
inter <- intersect(row.names(chonorm.upgmaeuc.by[[4]]), row.names(chonorm.upgmacor.by[[4]]))

##show intersection of UPGMAEUC and UPGMACOR
chonorm.inter <- chonorm.upgmacor.by[[4]][inter,]
chonorm.inter

##calculate intersection between above intersection and DIANACOR
inter <- intersect(row.names(chonorm.inter), row.names(chonorm.dianaeuc.by[[4]]))

##show intersection of UPGMAEUC, UPGMACOR, and DIANACOR
chonorm.inter <- chonorm.dianaeuc.by[[4]][inter,]
chonorm.inter
```

```
##write table
write.table(chonorm.inter, "chonorm-matchesM.txt")
```

## Appendix B: Confusion Matrices for Simulated Datasets

Confusion matrices are described in section 4.5. The true clustering (arranged in rows) is aligned with the method clustering (arranged in columns).

### Low-Variation, Low-noise dataset

```
[1] "UPGMAEUC"
Direct agreement: 4 of 4 pairs
Cases in matched pairs: 100 %
             simdata1.order
simdata1.true   1   2   3   4
            1 250   0   0   0
            2   0 250   0   0
            3   0   0 250   0
            4   0   0   0 250

[1] "UPGMACOR"
Direct agreement: 4 of 4 pairs
Cases in matched pairs: 100 %

             simdata1.order
simdata1.true   1   2   3   4
            1 250   0   0   0
            2   0 250   0   0
            3   0   0 250   0
            4   0   0   0 250


[1] "DIANAEUC"
Direct agreement: 4 of 4 pairs
Cases in matched pairs: 100 %
             simdata1.order
simdata1.true   1   2   3   4
            1 250   0   0   0
            2   0 250   0   0
            3   0   0 250   0
            4   0   0   0 250


[1] "DIANACOR"

             simdata1.order
simdata1.true   1   2   3   4
            1 250   0   0   0
            2   0 250   0   0
            3   0   0 250   0
            4   0   0   0 250

[1] "SOM1"
Error in matchClasses(as.matrix(simtable[[i]]), method = "exact") :
        Unique matching only for square tables.


[1] "KMEANS1"

Direct agreement: 2 of 4 pairs
Cases in matched pairs: 62.9 %

             simdata1.order
simdata1.true   2   3   1   4
            1 250   0   0   0
            2   0   0   0 250
            3   0 121 129   0
            4   0   0   0 250
```

## High-Variation, Low-Noise Dataset

```
[1] "UPGMAEUC"
Direct agreement: 1 of 4 pairs
Iterations for permutation matching: 6
Cases in matched pairs: 46 %
             simdata2.order
simdata2.true 4 3   1   2
            1 2 0 170  78
            2 0 5   0 245
            3 0 0 204  46
            4 0 1   0 249


[1] "UPGMACOR"
Direct agreement: 3 of 4 pairs
Iterations for permutation matching: 1
Cases in matched pairs: 75.1 %
             simdata2.order
simdata2.true   4   2   1 3
            1 250   0   0 0
            2   0 250   0 0
            3   0   0 250 0
            4   0 249   0 1


[1] "DIANAEUC"
Direct agreement: 3 of 4 pairs
Iterations for permutation matching: 1
Cases in matched pairs: 61.5 %
             simdata2.order
simdata2.true   4   3   2  1
            1 201   0   0 49
            2   0 250   0  0
            3   0   0 163 87
            4   0 249   0  1


[1] "DIANACOR"
Direct agreement: 3 of 4 pairs
Iterations for permutation matching: 1
Cases in matched pairs: 75.1 %
             simdata2.order
simdata2.true   4   2   1 3
            1 250   0   0 0
            2   0 250   0 0
            3   0   0 250 0
            4   0 249   0 1

[1] "SOM1"
Direct agreement: 0 of 4 pairs
Iterations for permutation matching: 24
Cases in matched pairs: 53.4 %
             simdata2.order
simdata2.true   4  2   3   1
            1 178  0  72   0
            2   0 82   0 168
            3 150  0 100   0
            4   0 75   1 174

[1] "KMEANS1"
Direct agreement: 0 of 4 pairs
Iterations for permutation matching: 24
Cases in matched pairs: 57.6 %
             simdata2.order
simdata2.true   4   2   3   1
            1 160   0  90   0
            2   0 136   1 113
            3  95   0 155   0
            4   0 124   1 125
```

## Low-Variation, High-Noise Dataset

```
[1] "UPGMAEUC"
Direct agreement: 3 of 4 pairs
Iterations for permutation matching: 1
Cases in matched pairs: 74.9 %
             simdata3.order
simdata3.true   2   1   3 4
            1 248   0   2 0
            2   0 250   0 0
            3   0   0 250 0
            4   0 249   0 1

[1] "UPGMACOR"
Direct agreement: 4 of 4 pairs
Cases in matched pairs: 99.6 %
             simdata3.order
simdata3.true   2   3   4   1
            1 249   0   1   0
            2   0 247   0   3
            3   0   0 250   0
            4   0   0   0 250

[1] "DIANAEUC"
Direct agreement: 4 of 4 pairs
Cases in matched pairs: 99.6 %
             simdata3.order
simdata3.true   2   3   4   1
            1 249   0   1   0
            2   0 248   0   2
            3   0   0 250   0
            4   0   1   0 249

[1] "DIANACOR"
Direct agreement: 4 of 4 pairs
Cases in matched pairs: 99.8 %
             simdata3.order
simdata3.true   2   3   4   1
            1 249   0   1   0
            2   0 249   0   1
            3   0   0 250   0
            4   0   0   0 250

[1] "SOM1"
Direct agreement: 3 of 4 pairs
Iterations for permutation matching: 1
Cases in matched pairs: 66.1 %
             simdata3.order
simdata3.true   4  2   3   1
            1 248  0   2   0
            2   0 21   0 229
            3  98  0 152   0
            4   0 10   0 240

[1] "KMEANS1"
Direct agreement: 4 of 4 pairs
Cases in matched pairs: 99.6 %
             simdata3.order
simdata3.true   4   2   3   1
            1 249   0   1   0
            2   0 248   0   2
            3   0   0 250   0
            4   0   1   0 249
```

References


1. Speed, T.P., *Statistical analysis of gene expression microarray data*. Interdisciplinary statistics. 2003, Boca Raton, FL: Chapman & Hall/CRC. xiii, 222.
2. Parmigiani, G., *The analysis of gene expression data : methods and software*. Statistics for biology and health. 2003, New York: Springer-Verlag. xix, 455 , 38 of plates.
3. Irizarry, R.A., et al., *Exploration, normalization, and summaries of high density oligonucleotide array probe level data.* Biostatistics, 2003. **4**(2): p. 249-64.
4. Hubbell, E., W.M. Liu, and R. Mei, *Robust estimators for expression analysis.* Bioinformatics, 2002. **18**(12): p. 1585-92.
5. Li, C. and W. Hung Wong, *Model-based analysis of oligonucleotide arrays: model validation, design issues and standard error application.* Genome Biol, 2001. **2**(8): p. RESEARCH0032.
6. Li, C. and W.H. Wong, *Model-based analysis of oligonucleotide arrays: expression index computation and outlier detection.* Proc Natl Acad Sci U S A, 2001. **98**(1): p. 31-6.
7. Yang, Y.H., et al. *Normalization for cDNA Microarray Data*. in *SPIE BiOS*. 2001. San Jose, California.
8. Cleveland, W.S. and S.J. Devlin, *Locally Weighted Regression: An Approach to Regression Analysis by Local Fitting.* Journal of the American Statistical Association., 1988. **83**(403): p. 596-610.
9. Brazma, A. and J. Vilo, *Minireview: Gene expression data analysis.* FEBS Letters, 2000. **480**: p. 17-24.
10. Eisen, M.B., et al., *Cluster analysis and display of genome-wide expression patterns.* Proc Natl Acad Sci U S A, 1998. **95**(25): p. 14863-8.
11. McWeeney, S., *Cluster "Analysis" and Visualization*, in *Statistical Analysis of Microarrays*. 2003: Portland.
12. MacNaughton-Smith, P., et al., *Dissimilarity analysis: a new technic of hierarchical subdivision.* Nature, 1964. **202**: p. 1034-5.
13. Ben-Hur, A., A. Elisseeff, and I. Guyon. *A stability based method for discovering structure in clustered data.* in *Pacific Symposium on Biocomputing*. 2002. Hawaii.
14. Davidson, I., *Understanding K-means Non-hierarchical Clustering*. 2002, SUNY.
15. Tamayo, P., et al., *Interpreting patterns of gene expression with self-organizing maps: methods and application to hematopoietic differentiation.* Proc Natl Acad Sci U S A, 1999. **96**(6): p. 2907-12.
16. Team, R.D.C., *R: A language and environment for statistical computing*. 2004, R Foundation for Statistical Coputing: Vienna, Austria.
17. Shamir, R. and S. Sharan, *Algorithmic Approaches to Clustering Gene Expression Data.*, in *Current Topics in Computational Molecular Biology*, MIT Press: Boston.
18. Dudoit, S. and J. Fridlyand, *A prediction-based resampling method for estimating the number of clusters in a dataset.* Genome Biol, 2002. **3**(7): p. RESEARCH0036.
19. Kaufmann, L. and P.J. Rousseeuw, *Finding Groups in Data: An Introduction to Cluster Analysis*. 1990, New York: Wiley.
20. Melia, M., *Comparing Clusterings. Technical Report 418*. 2002, UW Statistics Department: Seattle.

21. Cho, R.J., et al., *A genome-wide transcriptional analysis of the mitotic cell cycle.* Mol Cell, 1998. **2**(1): p. 65-73.
22. Yeung, K.Y., D.R. Haynor, and W.L. Ruzzo, *Validating clustering for gene expression data.* Bioinformatics, 2001. **17**(4): p. 309-18.
23. Yeung, K.Y., et al., *Model-based clustering and data transformations for gene expression data.* Bioinformatics, 2001. **17**(10): p. 977-87.
24. Sharan, R., R. Elkon, and R. Shamir, *Cluster analysis and its applications to gene expression data.* Ernst Schering Res Found Workshop, 2002(38): p. 83-108.
25. Lee, J., et al., *Interaction of yeast Rvs167 and Pho85 cyclin-dependent kinase complexes may link the cell cycle to the actin cytoskeleton.* Curr Biol, 1998. **8**(24): p. 1310-21.
26. Han, B.K., R. Aramayo, and M. Polymenis, *The G1 cyclin Cln3p controls vacuolar biogenesis in Saccharomyces cerevisiae.* Genetics, 2003. **165**(2): p. 467-76.
27. McKinney, J.D. and F.R. Cross, *FAR1 and the G1 phase specificity of cell cycle arrest by mating factor in Saccharomyces cerevisiae.* Mol Cell Biol, 1995. **15**(5): p. 2509-16.
28. Wittenberg, C. and S.I. Reed, *Plugging it in: signaling circuits and the yeast cell cycle.* Curr Opin Cell Biol, 1996. **8**(2): p. 223-30.
29. Piatti, S., C. Lengauer, and K. Nasmyth, *Cdc6 is an unstable protein whose de novo synthesis in G1 is important for the onset of S phase and for preventing a 'reductional' anaphase in the budding yeast Saccharomyces cerevisiae.* Embo J, 1995. **14**(15): p. 3788-99.
30. Takizawa, P.A., et al., *Actin-dependent localization of an RNA encoding a cell-fate determinant in yeast.* Nature, 1997. **389**(6646): p. 90-3.
31. Long, R.M., et al., *Mating type switching in yeast controlled by asymmetric localization of ASH1 mRNA.* Science, 1997. **277**(5324): p. 383-7.
32. McBride, H.J., Y. Yu, and D.J. Stillman, *Distinct regions of the Swi5 and Ace2 transcription factors are required for specific gene activation.* J Biol Chem, 1999. **274**(30): p. 21029-36.
33. Kuranda, M.J. and P.W. Robbins, *Chitinase is required for cell separation during growth of Saccharomyces cerevisiae.* J Biol Chem, 1991. **266**(29): p. 19758-67.
34. King, L. and G. Butler, *Regulation of expression of the chitinase gene CTS1 in Saccharomyces cerevisiae.* Biochem Soc Trans, 1997. **25**(3): p. 555S.
35. Guertin, D.A., S. Trautmann, and D. McCollum, *Cytokinesis in eukaryotes.* Microbiol Mol Biol Rev, 2002. **66**(2): p. 155-78.
36. Wang, Y., et al., *Exit from exit: resetting the cell cycle through Amn1 inhibition of G protein signaling.* Cell, 2003. **112**(5): p. 697-709.
37. Horak, C.E., et al., *Complex transcriptional circuitry at the G1/S transition in Saccharomyces cerevisiae.* Genes Dev, 2002. **16**(23): p. 3017-33.
38. Ubersax, J.A., et al., *Targets of the cyclin-dependent kinase Cdk1.* Nature, 2003. **425**(6960): p. 859-64.
39. Vogel, J., et al., *Phosphorylation of gamma-tubulin regulates microtubule organization in budding yeast.* Dev Cell, 2001. **1**(5): p. 621-31.
40. Richardson, H., et al., *Cyclin-B homologs in Saccharomyces cerevisiae function in S phase and in G2.* Genes Dev, 1992. **6**(11): p. 2021-34.
41. Ito, M., et al., *G2/M-phase-specific transcription during the plant cell cycle is mediated by c-Myb-like transcription factors.* Plant Cell, 2001. **13**(8): p. 1891-905.

42. Morgan, D.O., *Regulation of the APC and the exit from mitosis.* Nat Cell Biol, 1999. **1**(2): p. E47-53.
43. Dutta, A. and S.P. Bell, *Initiation of DNA replication in eukaryotic cells.* Annu Rev Cell Dev Biol, 1997. **13**: p. 293-332.
44. Higgins, J.M., *Structure, function and evolution of haspin and haspin-related proteins, a distinctive group of eukaryotic protein kinases.* Cell Mol Life Sci, 2003. **60**(3): p. 446-62.
45. Song, S. and K.S. Lee, *A novel function of Saccharomyces cerevisiae CDC5 in cytokinesis.* J Cell Biol, 2001. **152**(3): p. 451-69.
46. Jensen, S. and L.H. Johnston, *Complexity of mitotic exit.* Cell Cycle, 2002. **1**(5): p. 300-3.
47. McShane, L.M., et al., *Methods for assessing reproducibility of clustering patterns observed in analyses of microarray data.* Bioinformatics, 2002. **18**(11): p. 1462-9.
48. Datta, S., *Comparisons and validation of statistical clustering techniques for microarray gene expression data.* Bioinformatics, 2003. **19**(4): p. 459-66.
49. Jakt, L.M., et al., *Assessing clusters and motifs from gene expression data.* Genome Res, 2001. **11**(1): p. 112-23.