Cluster and TreeView
Manual

Software and Manual written by Michael Eisen

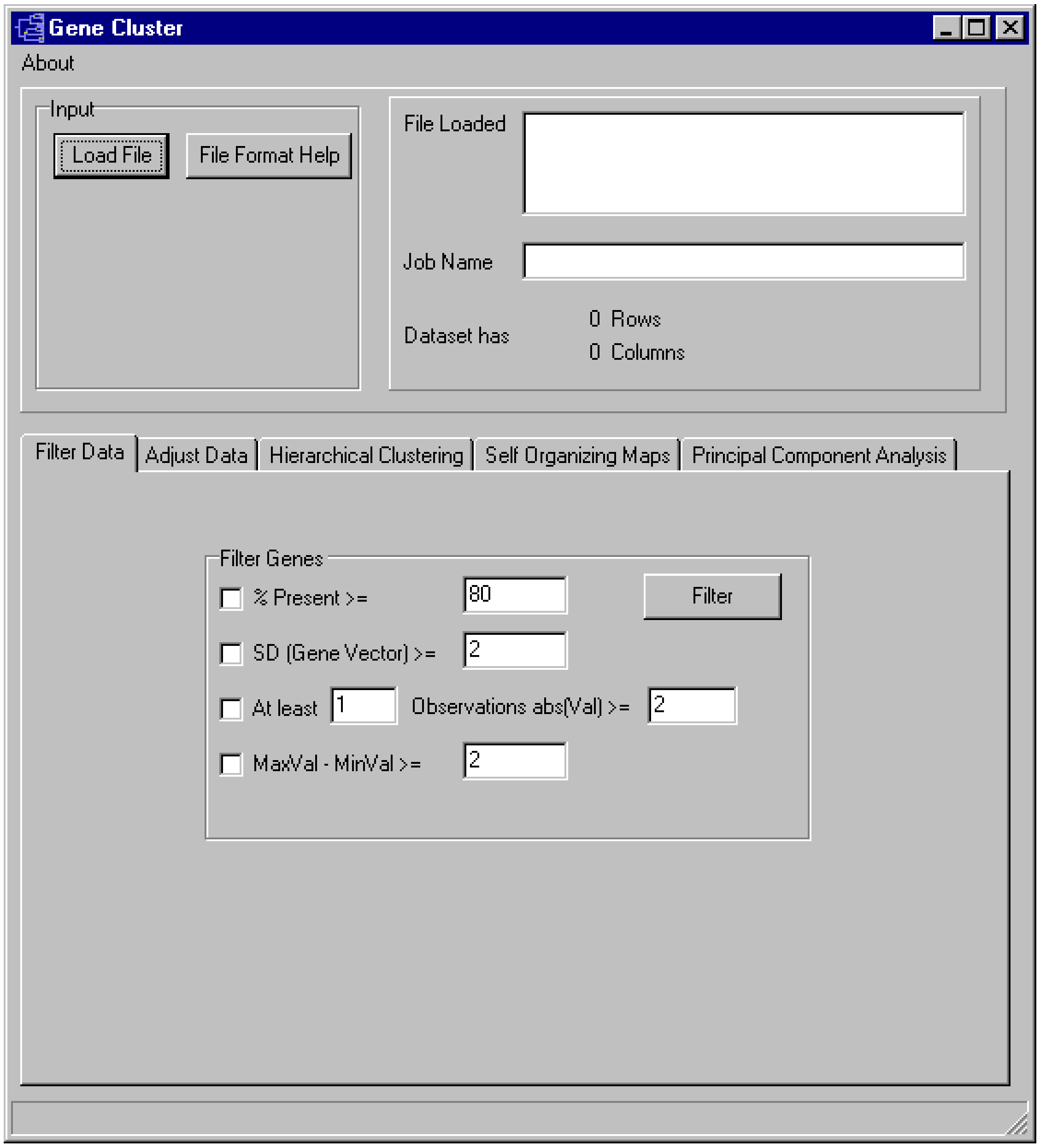
This manual is only partially complete and is a work in progress. A completed manual will be available by January 1,2000. Cluster and TreeView are Y2K Compliant because they are oblivious of date and time.

Introduction:

Cluster and TreeView are programs that provide a computational and graphical environment for analyzing data from DNA microarray experiments, or other genomic datasets. The program Cluster (which will soon be getting a new name) organizes and analyzes the data in a number of different ways. TreeView allows the organized data to be visualized and browsed. The next major release of this software (scheduled for early 2000) will integrate these two programs together into one application.

This manual is intended as a reference for using the software, and not as a comprehensive introduction to the methods employed. Many of the methods are drawn from standard statistical cluster analysis. There are excellent textbooks available on cluster analysis which are listed in the bibliography at the end. The bibliography also contains citations for recent publications in the biological sciences, especially genomics, that employ methods similar to those used here.

Cluster



- [ ] % Present >= 80
- [ ] SD (Gene Vector) >= 2
- [ ] At least 1 Observations abs(Val) >= 2
- [ ] MaxVal - MinVal >= 2

Loading Data: The first step in using Cluster is to import data. Currently, Cluster only reads tab-delimited text files in a particular format, described below. Such tab-delimited text files can be created and exported in any standard spreadsheet program, such as Microsoft Excel. An example datafile can be examined by pressing the *File Format Help* button on the Input panel of Cluster. This panel contains all the information you need for making a Cluster input file.

By convention, in Cluster input tables rows represent genes and columns represent samples or observations (e.g. a single microarray hybridization). For a simple timecourse, a *minimal* Cluster input file would look like this:

YORF	0 minutes	30 minutes	1 hour	2 hours	4 hours
YAL001C	1	1.3	2.4	5.8	2.4
YAL002W	0.9	0.8	0.7	0.5	0.2
YAL003W	0.8	2.1	4.2	10.1	10.1
YAL005C	1.1	1.3	0.8		0.4
YAL010C	1.2	1	1.1	4.5	8.3

Each row (gene) has an identifier (in green) that always goes in the first column. Here we are using yeast open reading frame codes. Each column (sample) has a label (in blue) that is always in the first row; here the labels describe the time at which a sample was taken. The first column of the first row contains a special field (in red) that tells the program what kind of objects are in each row. In this case, YORF stands for yeast open reading frame. This field can be any alpha-numeric value. It is used in TreeView to specify how rows are linked to external websites.

The remaining cells in the table contain data for the appropriate gene and sample. The "5.8" in row 2 column 4 means that the observed data value for gene YAL001C at 2 hours was 5.8. Missing values are acceptable and are designated by empty cells (e.g. YAL005C at 2 hours).

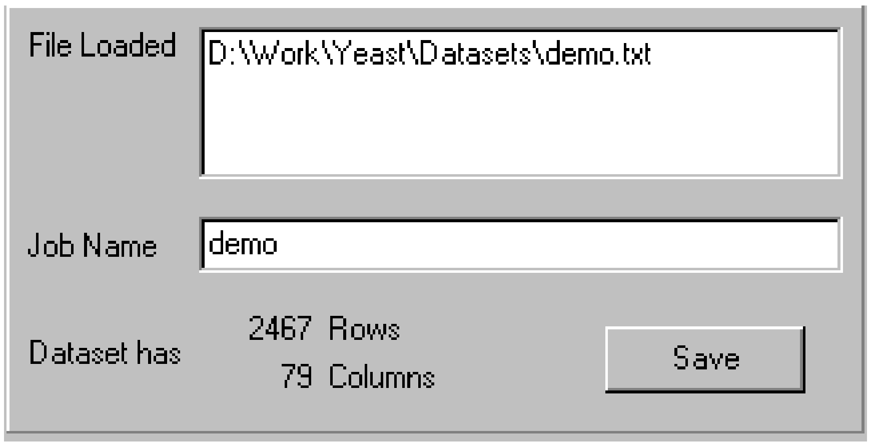
It is possible to have additional information in the input file. A maximal Cluster input file would look like this:

YORF	NAME	GWEIGHT	GORDER	0	30	1	2	4
EWEIGHT				1	1	1	1	0
EORDER				5	3	2	1	1
YAL001C	TFIIIC 138 KD SUBUNIT	1	1	1	1.3	2.4	5.8	2.4
YAL002W	UNKNOWN	0.4	3	0.9	0.8	0.7	0.5	0.2
YAL003W	ELONGATION FACTOR EF1-BETA	0.4	2	0.8	2.1	4.2	10.1	10.1
YAL005C	CYTOSOLIC HSP70	0.4	5	1.1	1.3	0.8		0.4

The yellow columns and rows are optional. By default, TreeView uses the ID in column 1 as a label for each gene. The NAME column allows you to specify a label for each gene that is distinct from the ID in column 1. The other rows and columns will be described later in this text.

Demo data: A demo datafile, which will be used in all of the examples here, is available at http://rana.stanford.edu/software/demo.txt. It contains yeast gene expression data described in Eisen et al. (1998) [see references at end]. Download this data and load it into Cluster.

Cluster will give you information about the loaded datafile.



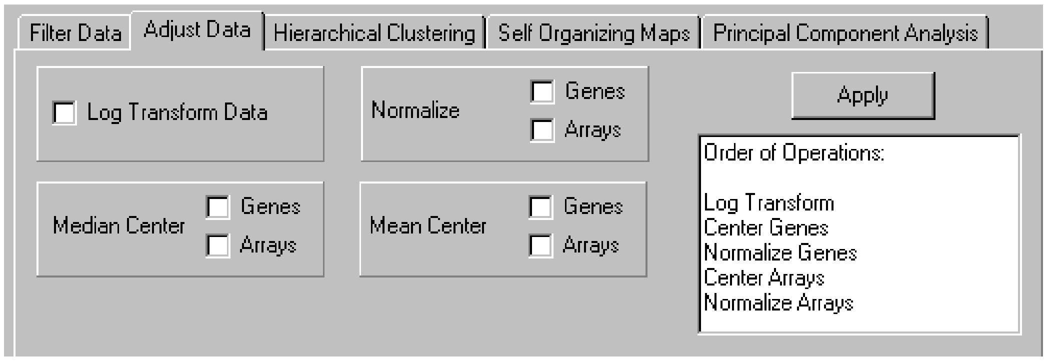
File Loaded: D:\Work\Yeast\Datasets\demo.txt

Job Name: demo

Dataset has 2467 Rows, 79 Columns

Adjusting and Filtering Data: Cluster provides a number of options for adjusting and filtering the data you have loaded. These functions are accessed via the *Filter Data* and *Adjust Data* tabs.

<u>Adjusting Data</u>:



Order of Operations:

Log Transform
Center Genes
Normalize Genes
Center Arrays
Normalize Arrays

From the Adjust Data tab, you can perform a number of operations that alter the underlying data in the imported table. These operations are

Log Transform Data: replace all data values X by $\log_2(X)$.

Normalize Genes and/or Arrays: Multiply all values in each row and/or column of data a scale factor S to so that the sum of the squares of the values is in each row and/or column is 1.0 (a separate S is computed for each row/column).

Mean Center Genes and/or Arrays: Subtract the row-wise or column-wise mean from the values in each row and/or column of data, so that mean value of each row and/or column is 0.

Median Center Genes and/or Arrays: Subtract the row-wise or column-wise mean from the values in each row and/or column of data, so that median value of each row and/or column is 0.

These operations are not associative, so the order in which these operations is applied is very important, and you should consider it carefully before you apply these operations.

The order of operations is (only checked operations are performed):

Log transform all values.
Mean center rows.
Median center rows.
Normalize rows.
Mean center columns.
Median center columns.
Normalize columns.

<u>When do you want to adjust data?</u>:

Log transformation: The results of many DNA microarray experiments are fluorescent ratios. Ratio measurements are most naturally processed in log space. Consider an experiment where you are looking at gene expression over time, and the results are relative expression levels compared to time 0. Assume at timepoint 1, a gene is unchanged, at timepoint 2 it is up 2-fold and at timepoint three is down 2-fold relative to time 0. The raw ratio values are 1.0, 2.0 and 0.5. In most applications, you want to think of 2-fold up and 2-fold down as being the same magnitude of change, but in an opposite direction. In raw ratio space, however, the difference between timepoint 1 and 2 is +1.0, while between timepoint 1 and 3 is -0.5. Thus mathematical operations that use the difference between values would think that the 2-fold up change was twice as significant as the 2-fold down change. Usually, you do not want this. In log space (we use log base 2 for simplicity) the data points become 0,1.0,-1.0.With these values, 2-fold up and 2-fold down are symmetric about 0. For most applications, we recommend you work in log space.

Mean/Median Centering: Consider a now common experimental design where you are looking at a large number of tumor samples all compared to a common reference sample made from a collection of cell-lines. For each gene, you have a series of ratio values that are relative to the expression level of that gene in the reference sample. Since the reference sample really has nothing to do with your experiment, you want your analysis to be independent of the amount of a gene present in the reference sample. This is achieved by adjusting the values of each gene to reflect their variation from some property of the series of observed values such as the mean or median. This is what mean and/or median centering of genes does. Centering makes less sense in experiments where the reference sample is part of the experiment, as it is many timecourses.

Centering the data for columns/arrays can also be used to remove certain types of biases. The results of many two-color fluorescent hybridization experiments are not corrected for systematic biases in ratios that are the result of differences in RNA amounts, labeling efficiency and image acquisition parameters. Such biases have the effect of multiplying ratios for all genes by a fixed scalar. Mean or median centering the data in log-space has the effect of correcting this bias, although it should be noted that an assumption is being made in correcting this bias, which is that the average gene in a given experiment is expected to have a ratio of 1.0 (or log-ratio of 0).

In general, I recommend the use of median rather than mean centering.

Normalization:

Normalization sets the magnitude (sum of the squares of the values) of a row/column vector to 1.0. Most of the distance metrics used by Cluster work with internally normalized data vectors, but the data are output as they were originally entered. If you want to output normalized vectors, you should select this option.

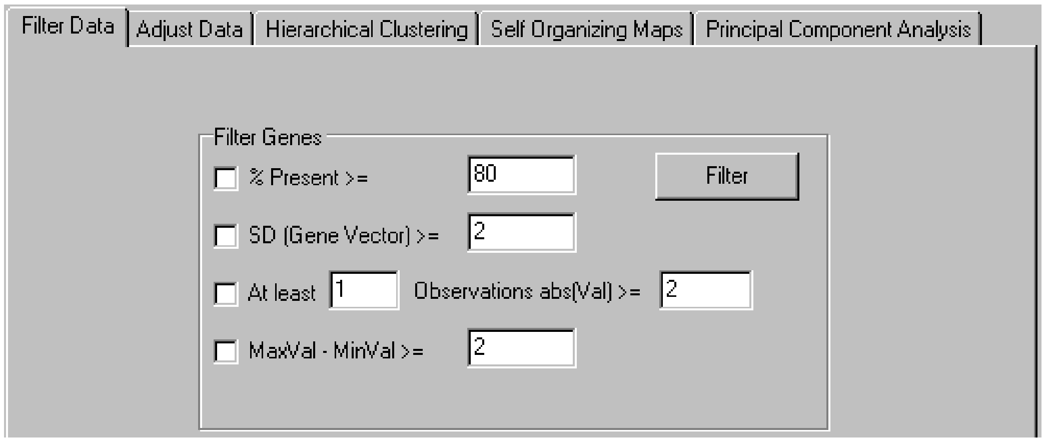
A sample series of operations for raw data would be:

Adjust Cycle 1) log transform
Adjust Cycle 2) median center genes and arrays
repeat (2) five to ten times
Adjust Cycle 3) normalize genes and arrays
repeat (3) five to ten times

This results in a log-transformed, median polished (i.e. all row-wise and column-wise median values are close to zero) and normal (i.e. all row and column magnitudes are close to 1.0) dataset.

After performing these operations you should save the dataset.

Filtering Data:



☐ % Present >= 80

☐ SD (Gene Vector) >= 2

☐ At least 1 Observations abs(Val) >= 2

☐ MaxVal - MinVal >= 2

The *Filter Data* tab allows you to remove genes that do not have certain desired properties from you dataset. The currently available properties that can be used to filter data are

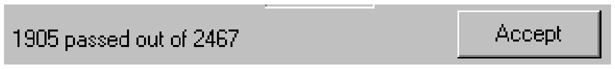
% Present >= X. This removes all genes that have missing values in greater than (100-X) percent of the columns.

SD (Gene Vector) >= X. This removed all genes that have standard deviations of observed values less than X.

At least X Observations abs(Val)>= Y. This removes all genes that do not have at least X observations with absolute values greater than Y.

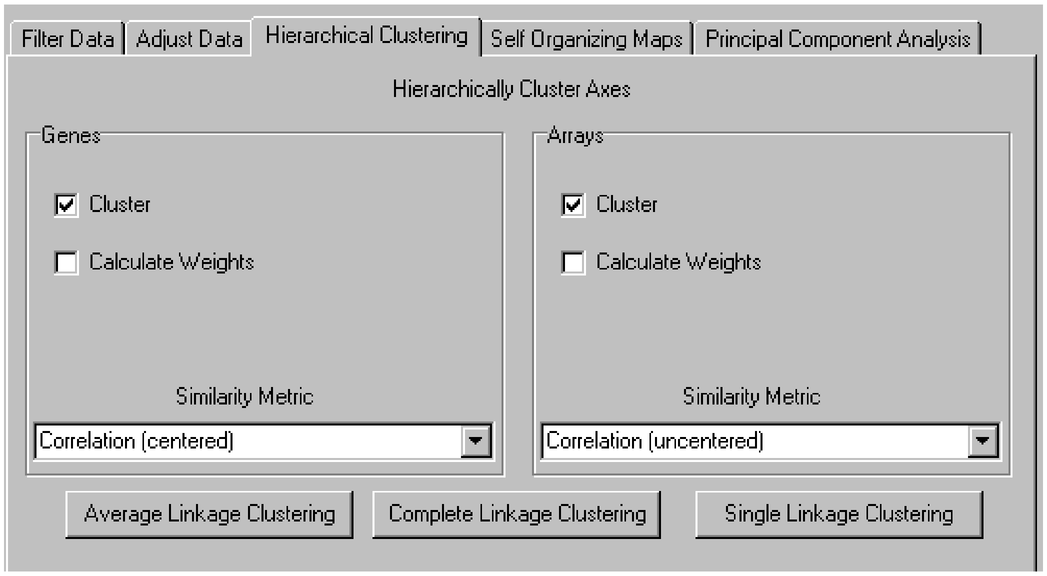
MaxVal-MinVal >=X. This removes all genes whose maximum minus minimum values are less than X.

These are fairly self-explanatory. When you press filter, the filters are not immediately applied to the dataset. You are first told how many genes would have passed the filter. If you want to accept the filter, you press Accept, otherwise no changes are made.



1905 passed out of 2467

Hierarchical Clustering:


Filter Data
Adjust Data
Hierarchical Clustering
Self Organizing Maps
Principal Component Analysis
Hierarchically Cluster Axes
Average Linkage Clustering
Complete Linkage Clustering
Single Linkage Clustering

Genes

✓ Cluster

☐ Calculate Weights

Similarity Metric: Correlation (centered)

Arrays

✓ Cluster

☐ Calculate Weights

Similarity Metric: Correlation (uncentered)

The *Hierarchical Clustering* tab allows you to perform hierarchical clustering on your data. This is an incredibly powerful and useful method for analyzing all sorts of large genomic datasets. Many published applications of this analysis are given in the references section at the end.

Cluster currently performs three types of binary, agglomerative, hierarchical clustering. The basic idea is to assemble a set of items (genes or arrays) into a tree, where items are joined by very short branches if they are very similar to each other, and by increasingly longer branches as their similarity decreases.

<u>Similarities/Distances</u>:

The first choice that must be made is how “similarity” is to be defined. There are many ways to compute how similar two series of numbers are, and Cluster provides a small number of options. The most commonly used similarity metrics are based on Pearson correlation. The Pearson correlation coefficient between any two series of number X={ $X_1, X_2, \ldots, X_N$ } and Y={ $Y_1, Y_2, \ldots, Y_N$ } is defined as

$$r = \frac{1}{N} \sum_{i=1,N} \left(\frac{X_i - \overline{X}}{\sigma_X} \right) \left(\frac{Y_i - \overline{Y}}{\sigma_Y} \right)$$

where $\overline{X}$ is the average of values in X, and σ_X is the standard deviation of these values.

There are many ways of conceptualizing the correlation coefficient. If you were to make a scatterplot of the values of X against Y (pairing X1 with Y1, X2 with Y2 etc…), then r reports how well you can fit a line to the values. If instead you think of X and Y as vectors in N dimensional space that pass through the origin, r tells you how large is the angle between them. The simplest way to think about the correlation coefficient is to plot X and Y as curves, with r telling you how similar the shapes of the two curves are. The Pearson correlation coefficient is always between -1 and 1, with 1 meaning that the two series are identical, 0 meaning they are completely independent, and -1 meaning they are perfect opposites. The correlation coefficient is invariant under scalar transformation of the data (that is, if you multiply all the values in Y by 2, the correlation between X and Y will be unchanged). Thus, two curves that have “identical” shape, but different magnitude, will still have a correlation of 1.

Cluster actually uses four different flavors of the Pearson correlation. The textbook Pearson correlation coefficient, given by the formula above, is used if you select *Correlation (centered)* in the Similarity Metric dialog box.

Correlation (uncentered) uses the following modified equation

$$r = \frac{1}{N} \sum_{i=i}^{N} \left(\frac{X_i}{\sqrt{\frac{1}{N} \sum_{i=1}^{N} (X_i)^2}} \right) \left(\frac{Y_i}{\sqrt{\frac{1}{N} \sum_{i=1}^{N} (Y_i)^2}} \right)$$

which is basically the same function, except that it assumes the mean is 0, even when it is not. The difference is that, if you have two vectors X and Y with identical shape, but which are offset relative to each other by a fixed value, they will have a standard Pearson correlation (centered correlation) of 1 but will not have an uncentered correlation of 1.

Cluster provides two similarity metrics that are the absolute value of these two correlation functions, which consider two items to be similar if they have opposite expression patterns; the standard correlation coefficients consider opposite genes are being very distant.

Two additional metrics are non-paramteric versions of Pearson correlation coefficients, which are described in

http://www.ulib.org/webRoot/Books/Numerical_Recipes/bookcpdf/c14-6.pdf

When either X or Y has missing values, only observations present for both X and Y are used in computing similarities.

Clustering:

With any specified metric, the first step in the clustering process is to compute the distance (the opposite of similarity; for all correlation metrics distance = 1.0 - correlation) between of all pairs of items to be clustered (e.g. the set of genes in the current dataset). This can often be time consuming, and, with the current implementation of the program, memory intensive (the maximum amount of memory required is 4*N*N bytes, where N is the number of items being clustered). The program updates you on its progress in computing distances.

Once this matrix of distances is computed, the clustering begins. The process used by Cluster is agglomerative hierarchical processing, which consists of repeated cycles where the two closest remaining items (those with the smallest distance) are joined by a node/branch of a tree, with the length of the branch set to the distance between the joined items. The two joined items are removed from list of items being processed replaced by a item that represents the new branch. The distances between this new item and all other remaining items are computed, and the process is repeated until only one item remains. Note that once clustering commences, we are working with items that are true items (e.g. a single gene) and items that are pseudo-items that contain a number of true items. There are a variety of ways to compute distances when we are dealing with pseudo-items, and Cluster currently provides three choices.

If you click *Average Linkage Clustering*, a vector is assigned to each pseudo-item, and this vector is used to compute the distances between this pseudo-item and all remaining items or pseudo-items using the same similarity metric as was used to calculate the initial similarity matrix. The vector is the average of the vectors of all actual items (e.g. genes) contained within the pseudo-item. Thus, when a new branch of the tree is formed joining together a branch with 5 items and an actual item, the new pseudo-item is assigned a vector that is the average of the 6 vectors it contains, and not the average of the two joined items (note that missing values are not used in the average, and a pseudo-item can have a missing value if all of the items it contains are missing values in the corresponding row/column). Note to people familiar with clustering algorithms. This is really a variant of true average linkage clustering, where the distance between two items X and Y is the mean of all pairwise distances between items contained in X and Y.

In *Single Linkage Clustering* the distance between two items X and Y is the minimum of all pairwise distances between items contained in X and Y.

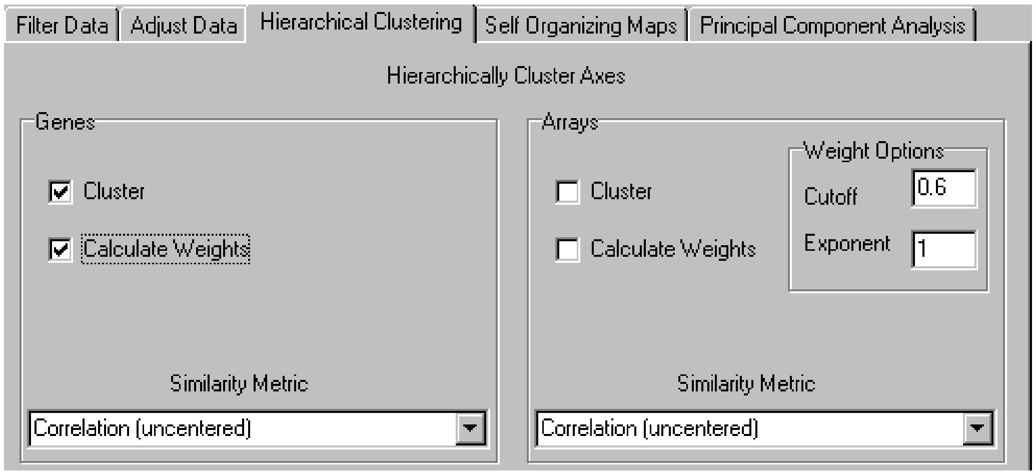
In *Complete Linkage Clustering* the distance between two items X and Y is the minimum of all pairwise distances between items contained in X and Y.

Weighting: By default, all of the observations for a given item are treated equally. In some cases you may want to give some observations more weight than others. For example, if you have duplicate copies of a gene on your array, you might want to downweight each individual copy when computing distances between arrays. You can specify weights using the GWEIGHT (gene weight) and EWEIGHT (experiment weight) parameters in the input file. By default all weights are set to 1.0. Thus, the actual formula, with weights included, for the Pearson correlation of X={ $X_1, X_2, \ldots, X_N$ } and Y={ $Y_1, Y_2, \ldots, Y_N$ } with observation weights of { $W_1, W_2, \ldots, W_N$ } is:

$$r = \frac{1}{\sum_{i=1}^{N} w_i} \sum_{i=1}^{N} w_i \left(\frac{X_i - \overline{X}}{\sigma_X} \right) \left(\frac{Y_i - \overline{Y}}{\sigma_Y} \right)$$

Note that when you are clustering rows (genes), you are using column (array) weights.

It is possible to compute weights as well based on a not entirely well understood function. If you want to compute weights for clustering genes, select the check box in the *Genes* panel of the *Hierarchical Clustering* tab



Genes: Cluster [✓], Calculate Weights [✓]
Arrays: Cluster [ ], Calculate Weights [ ]
Cutoff 0.6
Exponent 1
Similarity Metric: Correlation (uncentered)
Similarity Metric: Correlation (uncentered)

This will expose a *Weight Options* dialog box in the *Arrays* panel (I realize this placement is a bit counterintuitive, but it makes sense as you will see below).

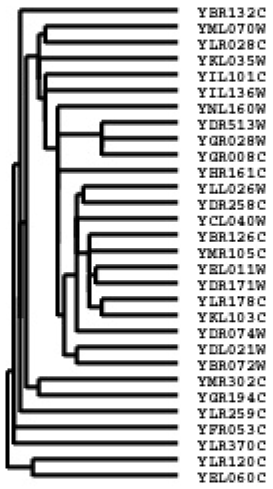
The idea behind the *Calculate Weights* option is to weight each row (the same idea applies to columns as well) based on the local density of row vectors in its vicinity, with a high density vicinity resulting in a low weight and a low density vicinity resulting in a higher weight. This is implemented by assigning a local density score *L(i)* to each row *i*.

$$L(i) = \sum_{\substack{\text{all rows j} \\ \text{where d(i,j)} < \text{k}}} \left(\frac{k - d(i,j)}{k} \right)^n$$

where k (cutoff) and n (exponent) are user supplied parameters. The weight for each row is $\frac{1}{L}$. Note that L(i) is always at least 1, since d(i,i) = 0. Each other row that is within the distance k of row i increases L(i) and decreases the weight. The larger d(i,j), the less L(i) is increased. Values of n greater than 1 mean that the contribution to L(i) drops off exponentially as d(i,j) increases.

<u>Ordering of Output File</u>:

The result of a clustering run is a tree or pair of trees (one for genes one for arrays). However, to visualize the results in *TreeView*, it is necessary to use this tree to reorder the rows and/or columns in the initial datatable. Note that if you simply draw all of the node in the tree in the following manner, a natural ordering of items emerges:



Thus, any tree can be used to generate an ordering. However, the ordering for any given tree is not unique. There is a family of 2^{N-1} ordering consistent with any tree of N items; you can flip any node on the tree (exchange the bottom and top branches) and you will get a new ordering that is equally consistent with the tree. By default, when Cluster joins two items, it randomly places one item on the top branch and the other on the bottom branch. It is possible to guide this process to generate the "best" ordering consistent with a given tree. This is done by using the GORDER (gene order) and EORDER (experiment order) parameters in the input file, or by running a self-organizing map (see section below) prior to clustering. By default, Cluster sets the order parameter for each row/column to 1. When a new node is created, Cluster compares the order parameters of the two joined items, and places the item with the smaller order value on the top branch. The order parameter for a node is the average of the order parameters of its members. Thus, if you want the gene order produced by Cluster to be as close as possible (without violating the structure of the tree) to the order in your input file, you use the GORDER column, and assign a value that increases for each row. Note that order parameters do not have to be unique.

Output Files:

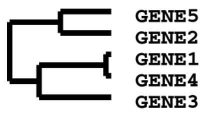
Cluster writes up to three output files for each hierarchical clustering run. The root filename of each file is whatever text you enter into the *Job Name* dialog box. When you load a file, *Job Name* is set to the root filename of the input file. The three output files are

JobName.cdt, JobName.gtr, JobName.atr

The .cdt (for clustered data table) file contains the original data with the rows and columns reordered based on the clustering result. It is the same format as the input files, except that an additional column and/or row is added if clustering is performed on genes and/or arrays. This additional column/row contains a unique identifier for each row/column that is linked to the description of the tree structure in the .gtr and .atr files.

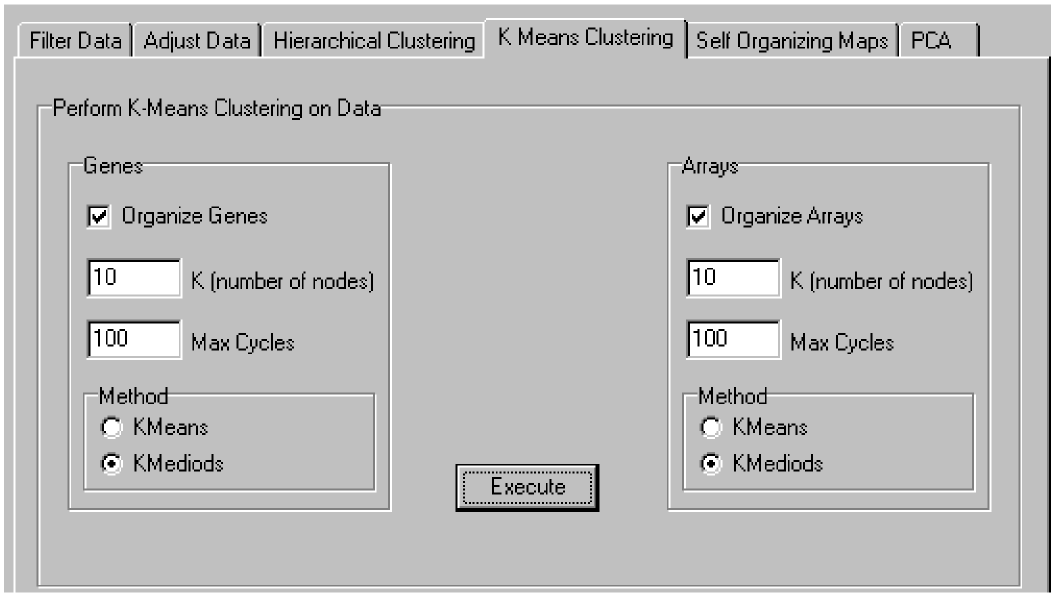
The .gtr (gene tree) and .atr (array tree) files are tab-delimited text files that report on the history of node joining in the gene or array clustering (note that these files are produced only when clustering is performed on the corresponding axis). When clustering begins each item to be clustered is assigned a unique identifier (e.g. GENE1X or ARRY42X- the X is a relic from the days when this was written in Perl and substring searches were used). These identifiers are added to the .cdt file. As each node is generated, it receives a unique identifier as well, starting is NODE1X, NODE2X, etc… Each joining event is stored in the .gtr or .atr file as a row with the node identifier, the identifiers of the two joined elements, and the similarity score for the two joined elements. These files look like:

```
NODE1X    GENE1X    GENE4X    0.98
NODE2X    GENE5X    GENE2X    0.80
NODE3X    NODE1X    GENE3X    0.72
NODE4X    NODE2X    NODE3X    0.60
```



The .gtr and/or .atr files are automatically read in TreeView when you open the corresponding .cdt file.

K-mean Clustering (new):



Perform K-Means Clustering on Data

Genes

[✓] Organize Genes

10 K (number of nodes)

100 Max Cycles

Method

○ KMeans

● KMediods

Arrays

[✓] Organize Arrays

10 K (number of nodes)

100 Max Cycles

Method

○ KMeans

● KMediods

K-means clustering is a simple, but popular, form of cluster analysis. The basic idea is that you start with a collection of items (e.g. genes) and some chosen number of clusters (k) you want to find. The items are initially randomly assigned to a cluster. K-means clustering proceeds by repeated application of a two-step process where:

1) the mean vector for all items in each cluster is computed
2) items are reassigned to the cluster whose center is closest to the item

The parameters that control k-means clustering are

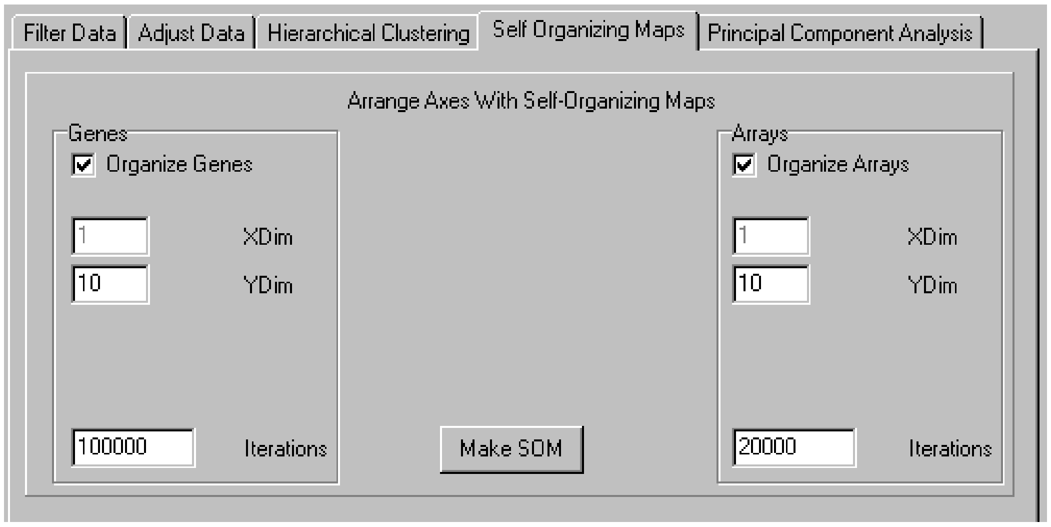
1) the number of clusters (K)
2) the maximum number of cycles

The output is simply an assignment of items to a cluster. The implementation here simply rearranges the rows and/or columns based on which cluster they were assigned to in the final cycle. The output filename is $\mathit{JobName\_K\_GK}_g$ $\mathit{\_AK}_a\mathit{.txt}$, where GK_g is included if genes were organized, and AK_g is included if arrays were organized.

Cluster also implements a slight variation on k-means clustering known as k-mediod clustering in which the median instead of the mean of items in a node are used.

The next version of Cluster will have a more sophisticated interface for K-means clustering.

Self-Organizing Maps:



Arrange Axes With Self-Organizing Maps

Genes
[✓] Organize Genes
1 XDim
10 YDim
100000 Iterations

Arrays
[✓] Organize Arrays
1 XDim
10 YDim
20000 Iterations

Self-Organizing Maps (SOMs) is a method of cluster analysis that are somewhat related to k-means clusterins. SOMs were invented in by Teuvo Kohonen in the early 1980s, and have recently been used in genomic analysis (see Chu 1998, Tamayo 1999 and Golub 1999 in references). The Tamayo paper contains a simple explanation of the methods. A more detailed description is available in the book by Kohonen, Self-Organizing Maps, 1997.

The current implementation varies slightly from that of Tamayo et al., in that it restricts the analysis one-dimensional SOMs along each axis, as opposed to a two-dimensional network. The one-dimensional SOM is used to reorder the elements on whichever axes are selected. The result is similar to the result of k-means clustering, except that, unlike k-means, the nodes in a SOM are ordered. This tends to result in a relatively smooth transition between groups.

The options for SOMs are 1) whether or not you will organize each axis, 2) the number of nodes for each axis (the default is the square-root of the number of items) and the number of iterations to be run.

The output file is of the form *JobName_SOM_G*X_g-Y_g*_A*X_a-Y_a*.txt*, where GX_g-Y_g is included if genes were organized, and AX_g-Y_g is included if arrays were organized. *X* and *Y* represent the dimensions of the corresponding SOM; note that in this version X is always 1. Up to two additional files (.gnf and .anf) are written containing the vectors for the SOM nodes.

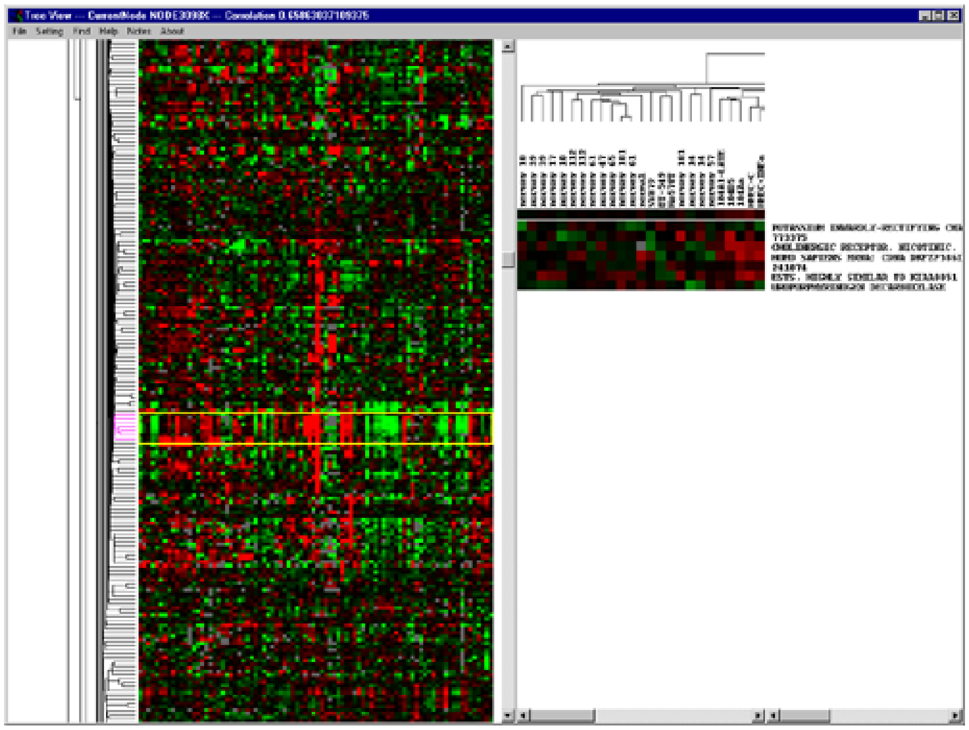
In the next version of the clustering software, two-dimensional SOMs will be supported and will have their own visualization methods.

SOMs and hierarchical clustering: Our original use of SOMs (see Chu et al., 1998) was motivated by the desire to take advantage of the properties of both SOMs and hierarchical clustering. This was accomplished by first computing a one dimensional SOM, and using the ordering from the SOM to guide the flipping of nodes in the hierarchical tree. In *Cluster*, after a SOM is run on a dataset, the GORDER and/or EORDER fields are set to the ordering from the SOM so that, for subsequent hierarchical clustering runs, the output ordering will come as close as possible to the ordering in the SOM without violating the structure of the tree.

Principal Component Analysis:

Cluster will perform principal component analysis on data. The output is very simple in this version and consists of two files: *JobName_svv.txt* that contains the principal components and *JobName_svu.txt* that contains the loadings of each gene on the principal components. A more sophisticated set of principal component based tools is being prepared in the next version of *Cluster*.

TreeView


Tree View -- CurrentNode NODE3098X -- Correlation 0.65863037109375
File Setting Find Help Notes About

TreeView is a program that allows interactive graphical analysis of the results from Cluster. It is fairly straightforward, but a manual is being prepared and will be available by January 1, 2000.

Bibliography

Brown, P. O., and Botstein, D. (1999). Exploring the new world of the genome with DNA microarrays. Nat Genet *21*, 33-7.

Chu, S., DeRisi, J., Eisen, M., Mulholland, J., Botstein, D., Brown, P. O., and Herskowitz, I. (1998). The transcriptional program of sporulation in budding yeast [published erratum appears in Science 1998 Nov 20;282(5393):1421]. Science *282*, 699-705.

Eisen, M. B., Spellman, P. T., Brown, P. O., and Botstein, D. (1998). Cluster analysis and display of genome-wide expression patterns. Proc Natl Acad Sci U S A *95*, 14863-8.

Hartigan, J. A. (1975). Clustering algorithms (New York,: Wiley).

Jain, A. K., and Dubes, R. C. (1988). Algorithms for clustering data (Englewood Cliffs, N.J.: Prentice Hall).

Jardine, N., and Sibson, R. (1971). Mathematical taxonomy (London, New York,: Wiley).

Kohonen, T. (1997). Self-organizing maps, 2nd Edition (Berlin ; New York: Springer).

Sneath, P. H. A., and Sokal, R. R. (1973). Numerical taxonomy; the principles and practice of numerical classification (San Francisco,: W. H. Freeman).

Sokal, R. R., and Sneath, P. H. A. (1963). Principles of numerical taxonomy (San Francisco,: W. H. Freeman).

Tryon, R. C., and Bailey, D. E. (1970). Cluster analysis (New York,: McGraw-Hill).

Tukey, J. W. (1977). Exploratory data analysis (Reading, Mass.: Addison-Wesley Pub. Co.).

Weinstein, J. N., Myers, T. G., O'Connor, P. M., Friend, S. H., Fornace, A. J., Jr., Kohn, K. W., Fojo, T., Bates, S. E., Rubinstein, L. V., Anderson, N. L., Buolamwini, J. K., van Osdol, W. W., Monks, A. P., Scudiero, D. A., Sausville, E. A., Zaharevitz, D. W., Bunow, B., Viswanadhan, V. N., Johnson, G. S., Wittes, R. E., and Paull, K. D. (1997). An information-intensive approach to the molecular pharmacology of cancer. Science *275*, 343-9.

Wen, X., Fuhrman, S., Michaels, G. S., Carr, D. B., Smith, S., Barker, J. L., and Somogyi, R. (1998). Large-scale temporal gene expression mapping of central nervous system development. Proc Natl Acad Sci U S A *95*, 334-9.