User Manual for ClusterProject Version 1.0

Software and Manual written by Haiyan Pan, Lei Wan & Jun Zhu

Zhejiang University

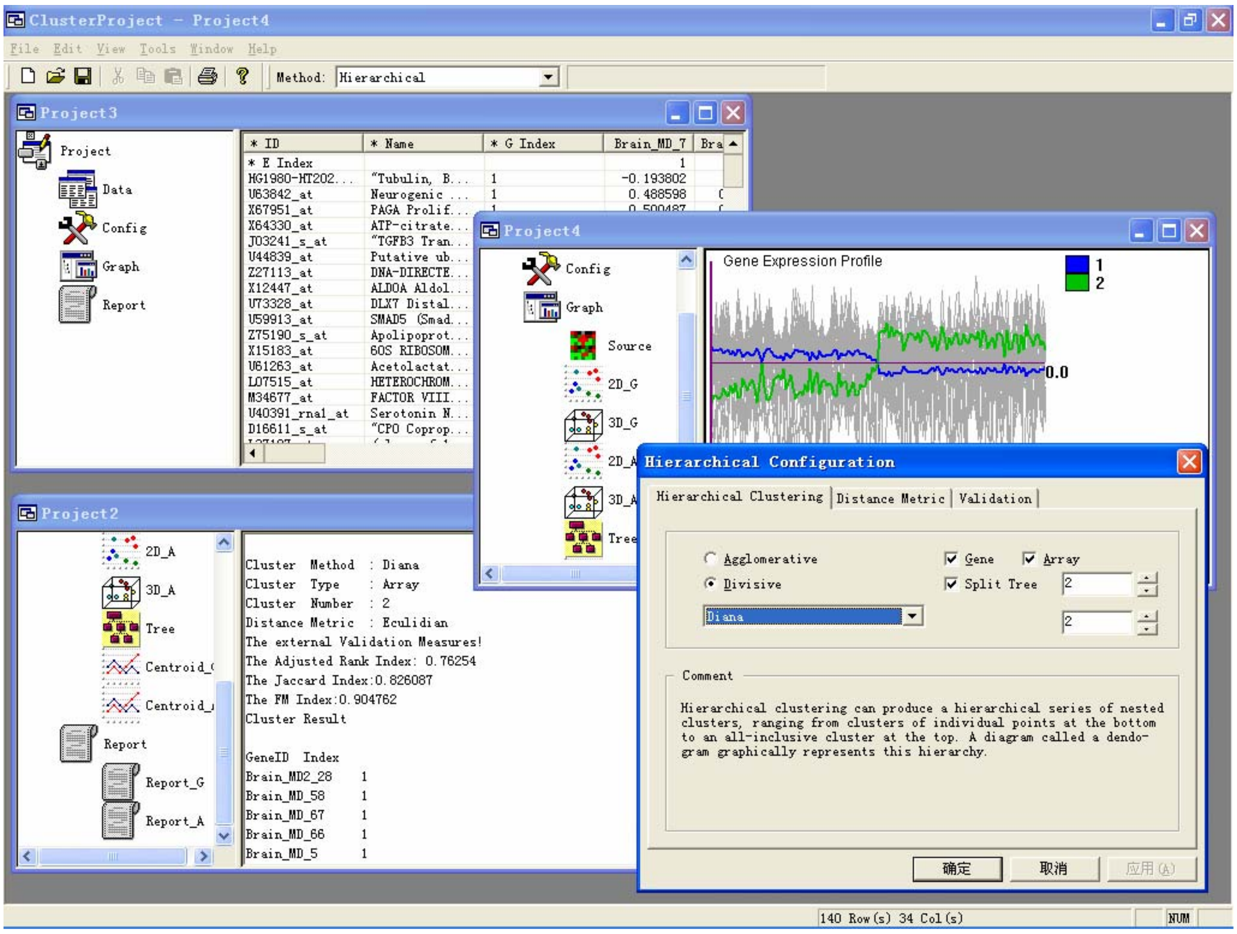
Hangzhou, China

June 2004

1. About ClusterProject 1.0

ClusterProject is a program that provides a computational and graphical environment for analyzing data from DNA microarray experiments, or other corresponding cluster datasets. The program ClusterProject organizes and analyzes the data in various ways and allows the organized data to be visualized.

This manual is intended as a reference for using the software, meanwhile 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.



Method: Hierarchical

* ID	* Name	* G Index	Brain_MD_7
* E Index			1
HG1980-HT202...	"Tubulin, B...	1	-0.193802
U63842_at	Neurogenic ...	1	0.488598
X67951_at	PAGA Prolif...	1	0.500487
X64330_at	ATP-citrate...		
J03241_s_at	"TGFB3 Tran...		
U44839_at	Putative ub...		
Z27113_at	DNA-DIRECTE...		
X12447_at	ALDOA Aldol...		
U73328_at	DLX7 Distal...		
U59913_at	SMAD5 (Smad...		
Z75190_s_at	Apolipoprot...		
X15183_at	60S RIBOSOM...		
U61263_at	Acetolactat...		
L07515_at	HETEROCHROM...		
M34677_at	FACTOR VIII...		
U40391_rna1_at	Serotonin N...		
D16611_s_at	"CPO Coprop...		

```
Cluster  Method  : Diana
Cluster  Type    : Array
Cluster  Number  : 2
Distance Metric  : Eculidian
The external Validation Measures!
The Adjusted Rank Index: 0.76254
The Jaccard Index:0.826087
The FM Index:0.904762
Cluster Result

GeneID  Index
Brain_MD2_28    1
Brain_MD_58     1
Brain_MD_67     1
Brain_MD_66     1
Brain_MD_5      1
```

Hierarchical Configuration

Hierarchical Clustering | Distance Metric | Validation

Agglomerative [✓ Divisive] — Diana

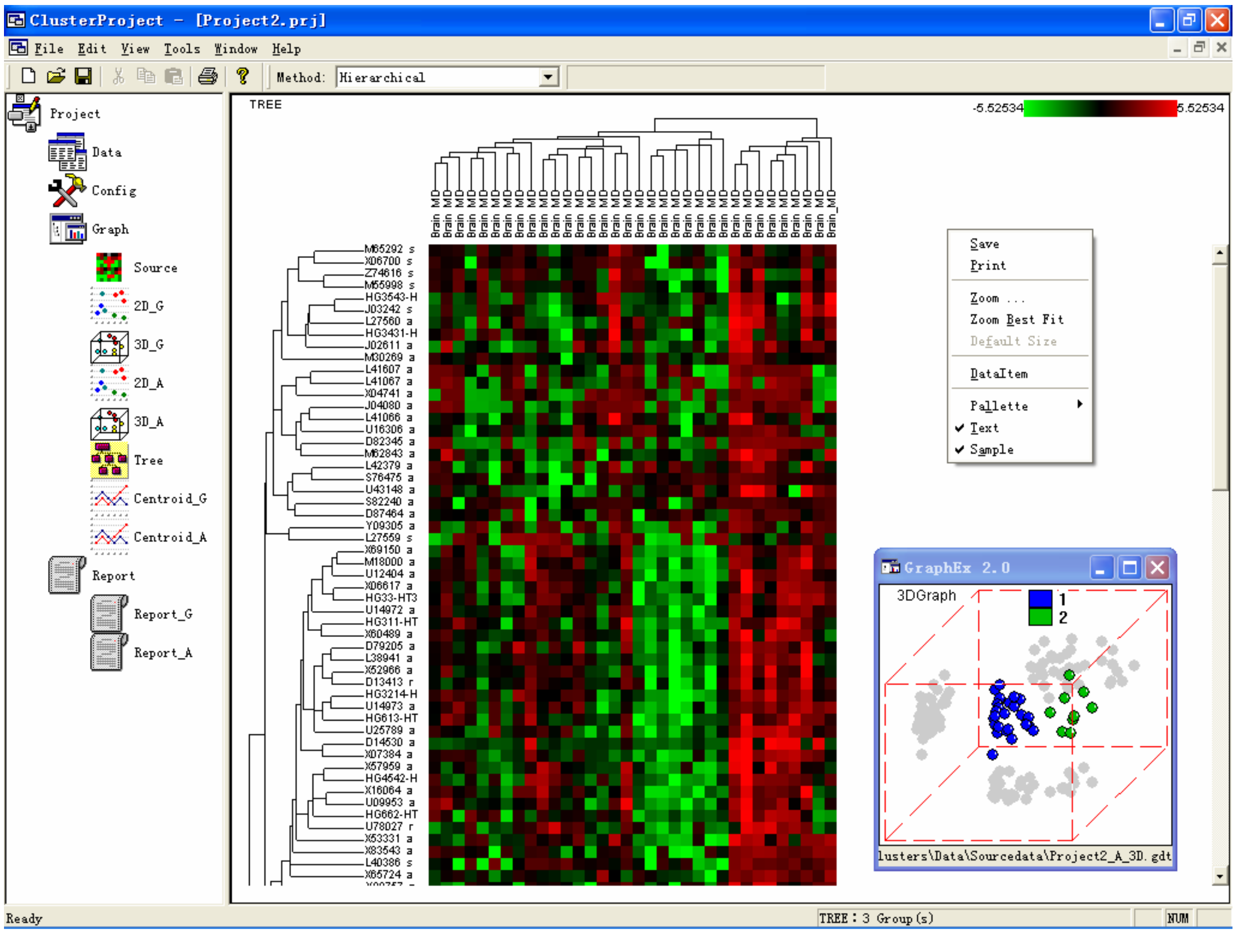
[✓] Gene [✓] Array [✓] Split Tree 2 / 2

Comment

Hierarchical clustering can produce a hierarchical series of nested clusters, ranging from clusters of individual points at the bottom to an all-inclusive cluster at the top. A diagram called a dendogram graphically represents this hierarchy.

140 Row(s) 34 Col(s)

- ✧ MDI (Multi-Document Interface) framework enables the user to work with more than one document at the same time.
- ✧ The various clustering algorithms are available with custom-preferences.



✧ Project manages the all resources under four types: data, preference, graph and report with a tree browser.

✧ The various graphical modes include data table, text report, 2D distributing, 3D distributing, grid matrix, tree, and centroid curve. These graphical modes can be saved as BMP or PDF documents.

2. System Requirement

- **Hardware:** PC 586 or higher; 64MB RAM or more; at least 10 MB free space on hard disk.
- **Operating System:** Microsoft Window 95/98, Windows 2000 or Windows XP.
- **Recommended Operating System:** Microsoft Window 2000/XP.

3. Install and Update

All related files for ClusterProject 1.0 are compressed into ClusterProject.exe.

Installing ClusterProject

1) Download the ClusterProject.exe file to a temporary directory.
2) Click ClusterProject.exe to install this program.
3) Follow the installation wizard to finish the installation.

ClusterProject can be installed on your computer in the folder which you nominate during the installation process. The defaulted installation path is: C:\Program Files\ClusterProject\.

Uninstalling ClusterProject

To remove ClusterProject, one way is to open the Windows Control Panel, double click Add/Remove Programs, select ClusterProject, and then click Add/Remove. Another way is to click "Uninstall ClusterProject" in Start/Programes Menu.

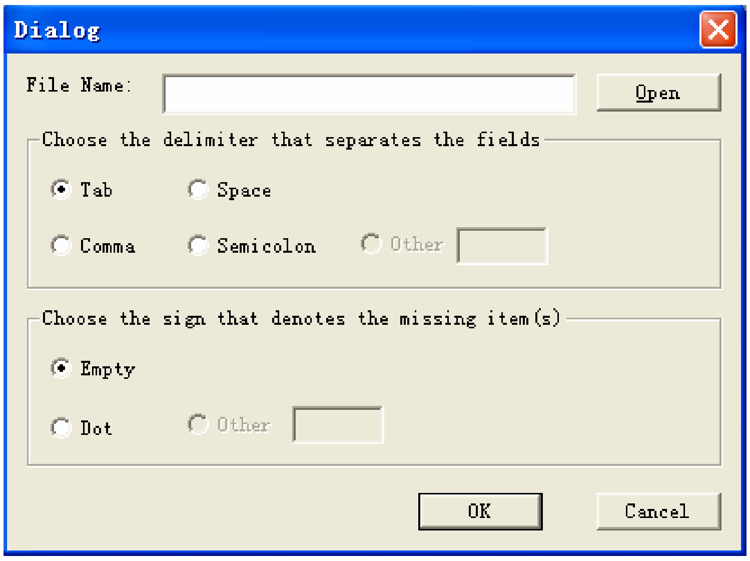
Update new version

You can get the latest version from http://www.cab.zju.edu.cn/english/ics/faculty/zhujun.htm or send E-mail to Authors: hypan@scbit.org or jzhu@zju.edu.cn.

4. Basic Operation

4.1 Inserting Data

- Click ***File/New*** menu or the first button on the **Toolbar**. A dialog will be popped up.


Dialog

File Name: [ ]

Choose the delimiter that separates the fields
[✓ Tab] Space
Comma Semicolon Other

Choose the sign that denotes the missing item(s)
[✓ Empty]
Dot Other

Currently, ClusterProject reads two types of text files in particular formats described below. It offers several the delimiters to separate the text, meanwhile, missing values are acceptable and are designated by several signs. The default postfix of original data files is ***.sdt** and other postfixes such as ***.txt** can be acceptable. Such text files can be created and exported in any standard spreadsheet program, such as Microsoft Excel.

Type 1: The first standard format of input file is described as follow. In this standard format, rows represent genes and columns represent samples or treatments or time points in a biological process. For a simple time course, a minimal 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	1.4	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, and each column (sample) has a label (in blue) that is always in the first row. The remaining cells in the table contain data for the appropriate gene and sample. The "5.8" in row 2 and column 4 means that the observed data value for gene YAL001C at 2 hours is 5.8.

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

YORF	**NAME**	**GINDEX**	0 min	30 min	1 h	2 h	4 h
EINDEX			1	1	1	1	1
YAL001C	TFIIIC 138 KD SUBUNIT	1	1.3	2.4	5.8	2.4	1
YAL002W	UNKNOWN	1	0.9	0.8	0.7	0.5	0.2
YAL003W	ELONGATION FACTOR EF1-BETA	1	0.8	2.1	4.2	10.1	10.1
YAL005C	CYTOSOLIC HSP70	1	1.1	1.3	0.8		0.4

The yellow cells are optional. By default, ClusterProject 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 GINDEX column allows user to define the class label of each gene and the EINDEX row allows user to define the class label of each sample.

Type 2: Another standard format of original file would look like this:

[TYPE1.0]			
GeneID	**TimePoint**	**Rep**	**Trait**
YAL001C	0 min	1	-1.0
YAL001C	30 min	1	2.0
YAL001C	1 h	1	3.0
YAL002W	0 min	1	-0.6
YAL002W	30 min	1	0.12
YAL002W	1 h	1	-0.36

The red cell **[TYPE1.0]** is necessary and the input should be accordant to this flag. The column of **Rep** is indispensable whether the experiment has replication or not. If there is no replication, all values of this column are set to one. It can have additional factors in the input file such as dye, treatment or array *et al*.

4.2 Cluster Methods

- Click the button (**Figure 1**) on the **Toolbar**. In current, ClusterProject contains two common kind methods: **hierarchical** and **partition** clustering methods.

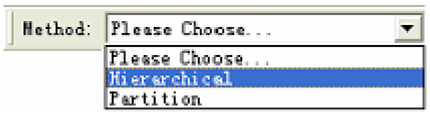
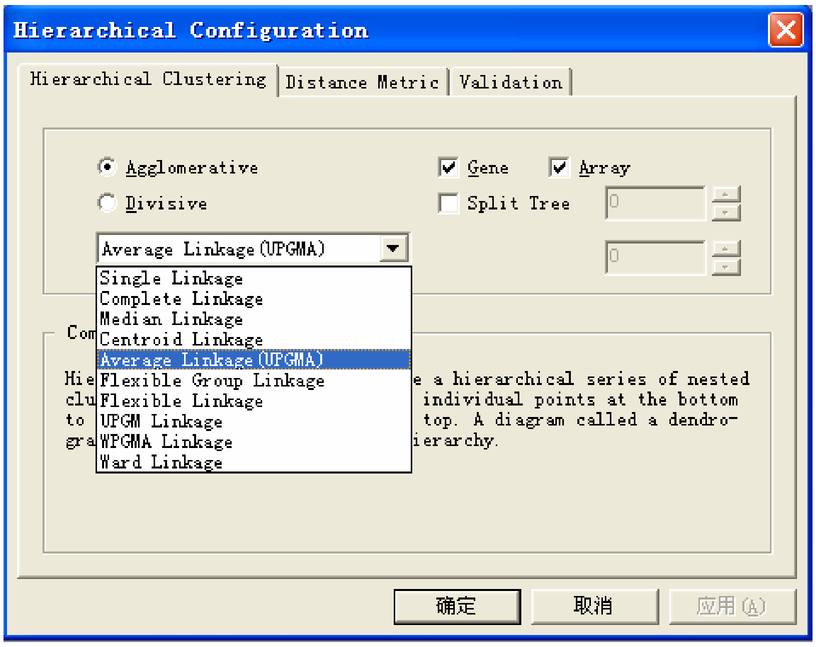


Figure 1

Cluster analysis groups objects based on the information found in the data to describe their relationships. The goal of cluster analysis makes that the objects in a group will be similar (or related) to each other and different from (or unrelated to) the objects in other groups. There are abundant clustering methods available in many literatures.

4.2.1 Hierarchical Clustering

- Click ***Hierarchical Clustering*** attribute, you can perform hierarchical clustering methods on your data to obtain meaningful results. Meanwhile you can select ***Gene*** or ***Array*** or both to perform cluster analysis. ClusterProject currently implements ten types of **agglomerative** hierarchical and one **divisive** hierarchical clustering (*Diana*). It is notable that the generic term "**array**" refers to tissue types, treatments or time points in a biological process. The meaning of this term is same in other interfaces.

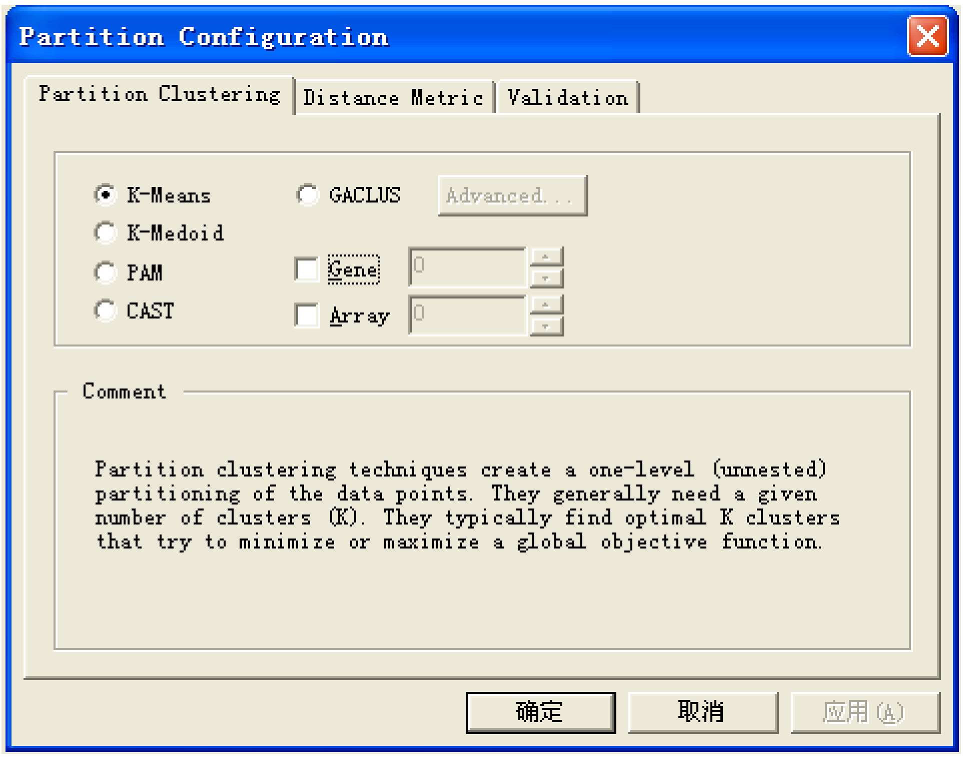


If you click ***Split Tree***, you can split the hierarchical tree at a desirable height level and then obtain several similar groups.

See also: Hierarchical Clustering

4.2.2 Partition Clustering

- Click ***Partition Clustering*** attribute, you can perform partition clustering methods on your data. Meanwhile you should input the ***cluster number*** advanced. ClusterProject currently performs five types of partition clustering: K-Means, K-Medoid, PAM, CAST and GACLUS (which is based on genetic algorithms, See also: Genetic Algorithm).



(● K-Means) ○ K-Medoid ○ PAM ○ CAST ○ GACLUS

☐ Gene 0
☐ Array 0

Comment

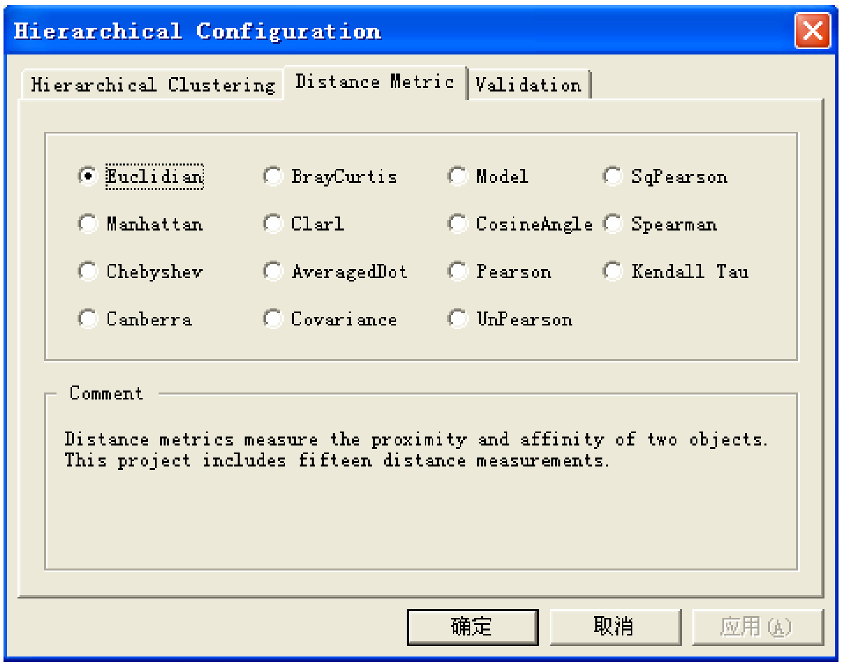
Partition clustering techniques create a one-level (unnested) partitioning of the data points. They generally need a given number of clusters (K). They typically find optimal K clusters that try to minimize or maximize a global objective function.

If you click ***GACLUS***, you can configure the attributes of GACLUS such as selection, crossover, mutation, and reinsertion etc.

See also: Partition Clustering

4.3 Distance Methods

- Click ***Distance Metric*** attribute, you can select various distance metrics to combine with a particular clustering method.



[✓ Euclidian] BrayCurtis Model SqPearson Manhattan Clarl CosineAngle Spearman Chebyshev AveragedDot Pearson Kendall Tau Canberra Covariance UnPearson

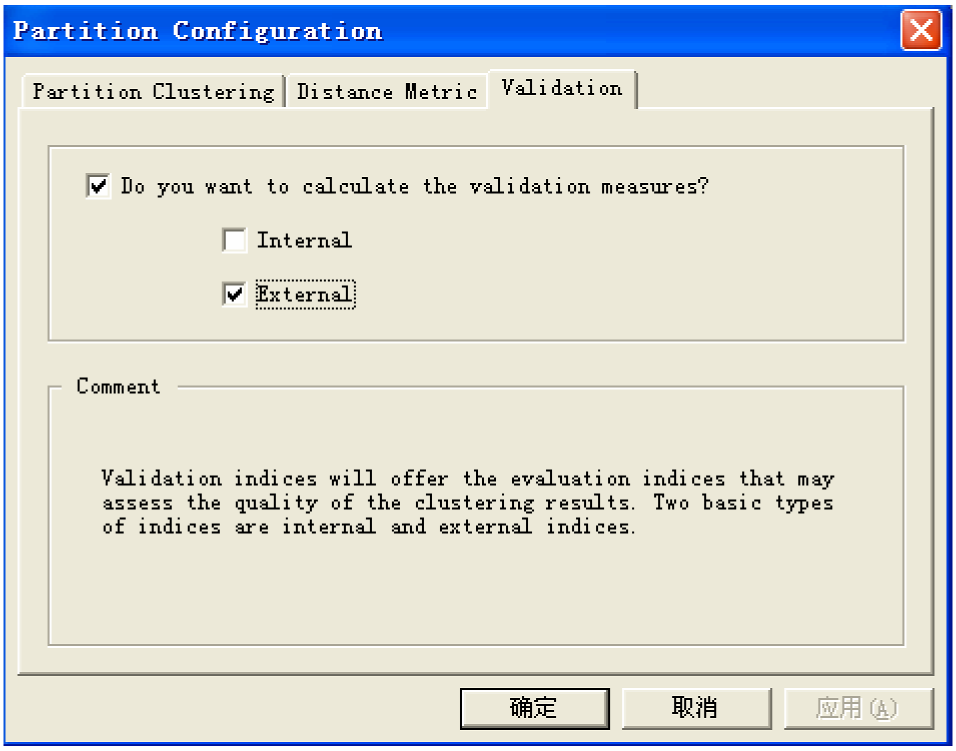
Comment

Distance metrics measure the proximity and affinity of two objects.
This project includes fifteen distance measurements.

All clustering algorithms use a dissimilarity or similarity measurement between samples or genes to compare patterns of expression. In current, this software contains fifteen distance metrics. Among these metrics, six metrics are correlation coefficients (cosine-angle correlation, Pearson correlation, Uncentered Pearson correlation, Squared Pearson correlation, Spearman rank-order correlation, and Kendall's Tau correlation).

4.4 Validation Methods

- Click ***Validation*** attribute, you can select two evaluation metrics to combine with a particular clustering method.



[✓] Do you want to calculate the validation measures?

[ ] Internal

[✓] External

Comment

Validation indices will offer the evaluation indices that may assess the quality of the clustering results. Two basic types of indices are internal and external indices.

In current, ClusterProject contains three types of external indices: **Adjusted rank index**, **Jaccard index**, and **FM index** and four types of internal indices: **homogeneity** and **separation**, **VRC criterion**, and **silhouette width**.

See also: Evaluation Indices

4.5 Output Files

ClusterProject writes up to several output files for each clustering run. The root filename of each file is whatever text you name the project, and the project file will save configurations and source data filename.

The output filename extension specifies the type of output file. It can be one of the following names: *.sdt， *.gdt, *.rpt.

- ***.sdt** Document of original data, which are tab-delimited text file. In fact, it is the copy of the file which you insert into this project.

  See also: Data

- ***.gdt** Document of graph results, it can be one of the following types: 2D/3D distribution (***_2D.gdt** or ***_3D.gdt**), hierarchical tree (***_T.gdt**), expression profile (***_C.gdt** or ***_M.gdt**). If clustering for genes, the filename will add **_G**. For example, if it is 2D distribution for genes, the document is ***_G_2D.gdt**. If clustering for arrays, the filename will add **_A**. It is same for other graph results and text report. These files are automatically read in ClusterProject when you open the corresponding ***.prj** file. You can browse them by a graph viewer embedded in ClusterProject.

  See also: Graph

- ***.rpt** Document of text report, it contains the configurations which you select and the cluster results of each run.

4.6 Graphical Representation

The raw data and cluster results obtained by various cluster analysis can be visualized by several types ways and assist the biologists to obtain meaningful conclusions. Click ***Graph*** menue of the left view, you can see several graph types. The main types of visualizations are described as follow:

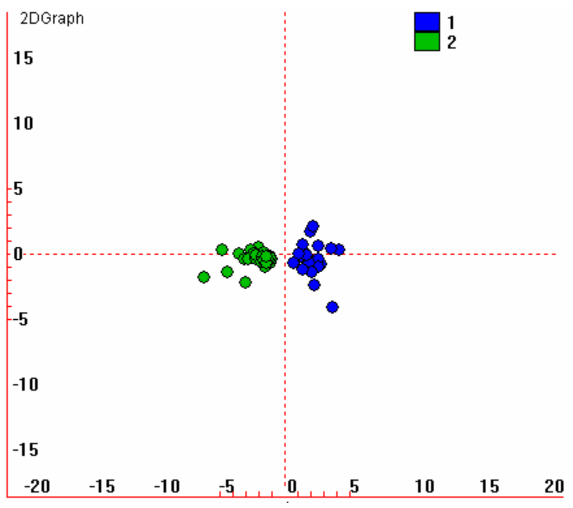


Fig.1. After PCA, the first two PCs are extracted. The expression pattern can be represented by 2-dimensional graph. The *x*-axis represents PC1 and *y*-axis PC2.

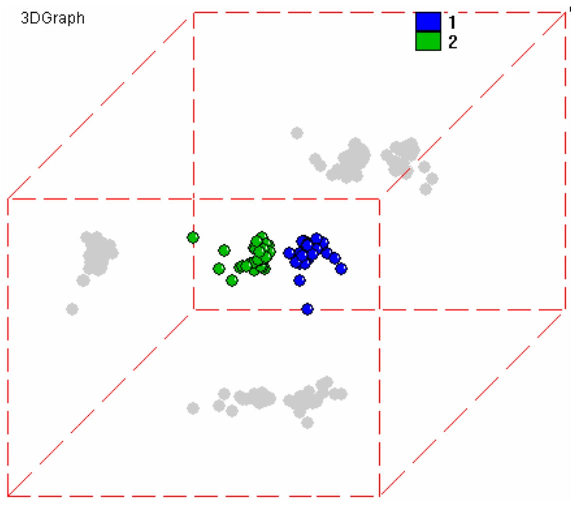


Fig.2. After PCA, the first two PCs are extracted. The expression pattern can be represented by 2-dimensional graph. The *x*-axis represents PC1, *y*-axis PC2 and *z*-axis PC3.

The primary data is combined with a graphical representation by representing each data point with a color that quantitatively reflects the original experimental observation. Color scales range usually from saturated green (negative max absolute value) to saturated red (positive max)

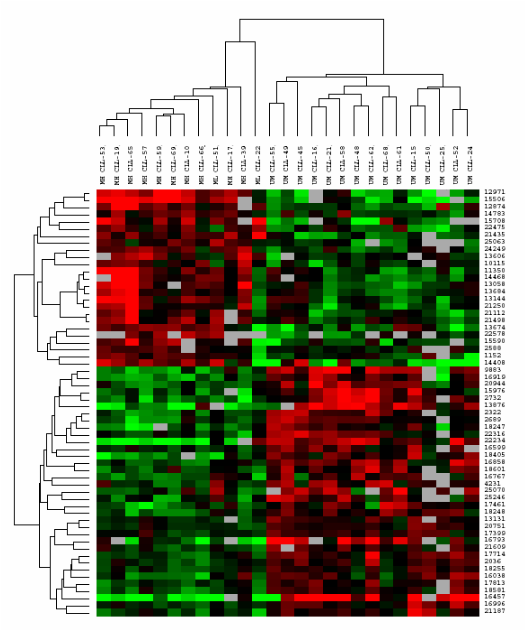


Fig.3. The dendrogram of cluster groups using Hierarchical Clustering methods.

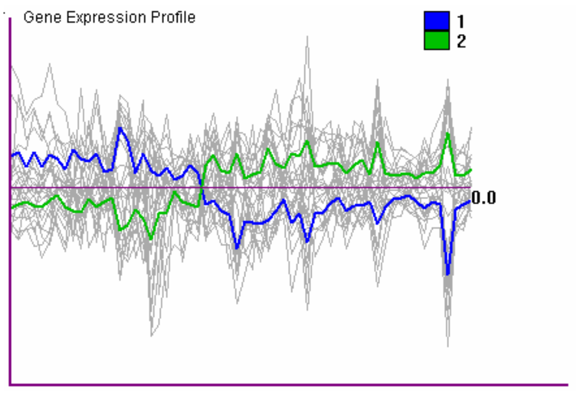


Fig.4. Expression patterns of all genes or samples and the mean profiles of groups.

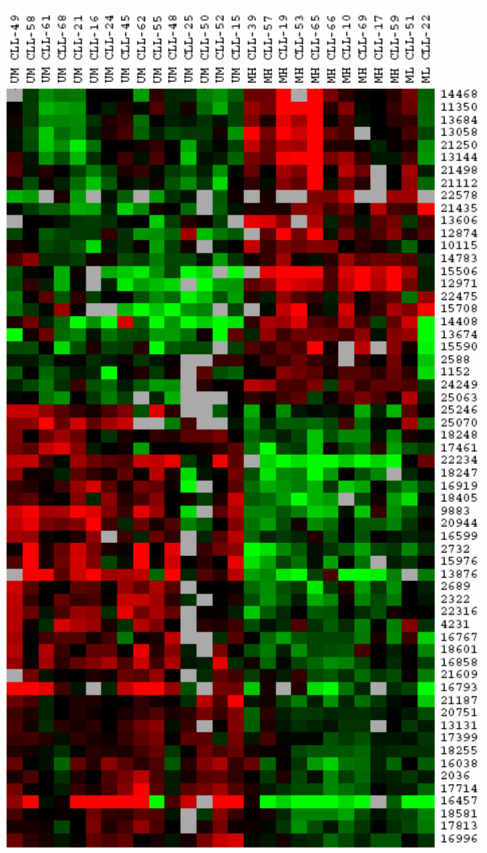


Fig.5. The expression profile of raw gene expression data.

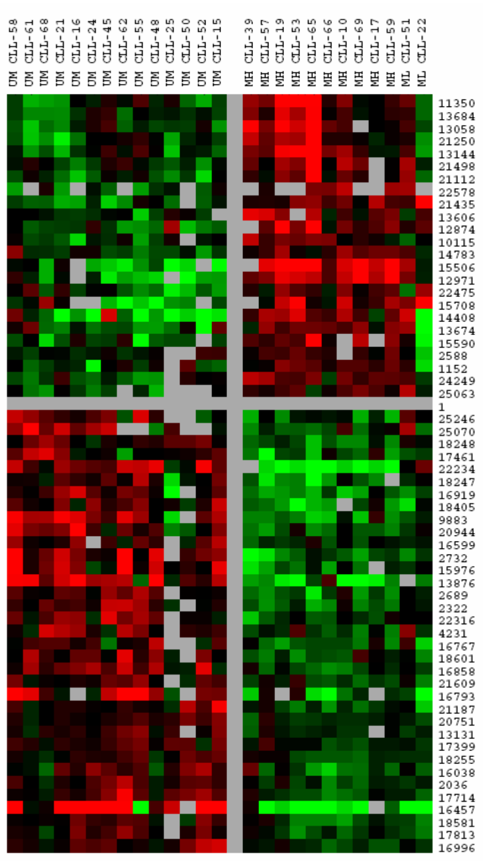
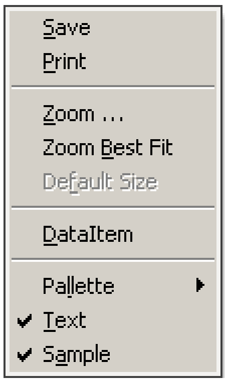


Fig.6. The expression profile of clusters using Partition Clustering Methods.

4.7 Context Menu

- Click the right of mouse on a graph, a particular menu will be popped up. The detailed context of this menu is as follow:



Save	Save as Windows Bitmap file or PostScript file. You can export your graph results and share them with others.
Print	Print graph results.
Zoom	Show a pop-up dialog with a slider. You can move the slider to zoon in/out the graph.
Zoom Best Fit	Show the graph as the “best” size for the window.
DataItem	Show or hide the components in the graph.
Palette	Change the palette for profile graph.
Text	Show or hide the topic of the graph.
Sample	Show or hide the color sample of the graph.
Multicolor	Change the color setting: Multicolor or Monochrome

4.8 Graph Viewer

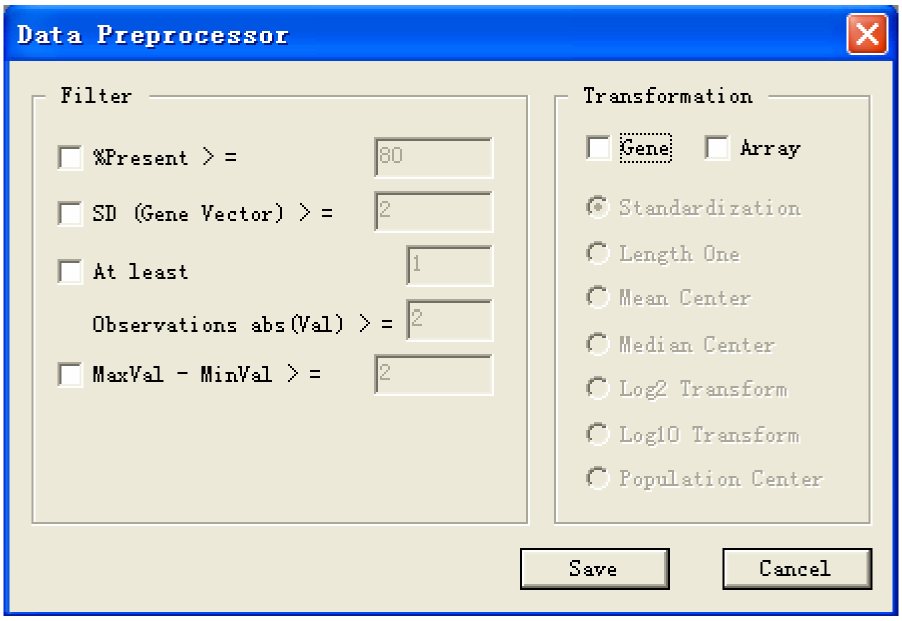
Click ***Tools/Graph viewer*** menu, you can take a quick view for your graph file without open a project by an embedded graph viewer. When you start this menu, a file-open dialog will popped up, and you can input a ***.gdt** file, **Graph Viewer** will show it.

4.9 Data Browser

Click ***Tools/Data Browser*** menu, you can take a quick view for the original data without open a project or insert it into a new project. When you click this menu, a file-open dialog will be popped up, and you can input an existing ***.sdt** file, **Data Browser** will show it in a grid table.

4.10 Data Preprocessor

- Click ***Tools/Preprocessor*** menu. You can filter and transform the data inputted.



Data Preprocessor

Filter

☐ %Present > = 80

☐ SD (Gene Vector) > = 2

☐ At least 1

Observations abs(Val) > = 2

☐ MaxVal - MinVal > = 2

Transformation

☐ Gene ☐ Array

● Standardization

○ Length One

○ Mean Center

○ Median Center

○ Log2 Transform

○ Log10 Transform

○ Population Center

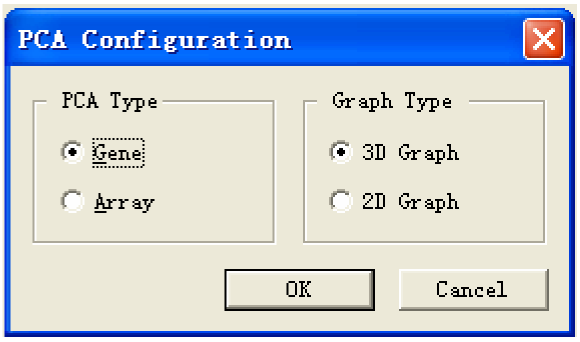
Save Cancel

The ***Filter*** menu allows you to remove genes that do not have certain desired properties from your dataset, and the ***Transformation*** menu, you can perform a number of operations that alter the underlying data in the imported table.

See also: Preprocessing

4.11 Principle Component Analysis (PCA)

- Click ***Tools/PCA*** menu allows you to perform the PCA process on your data.



PCA Type: [✓ Gene] Array

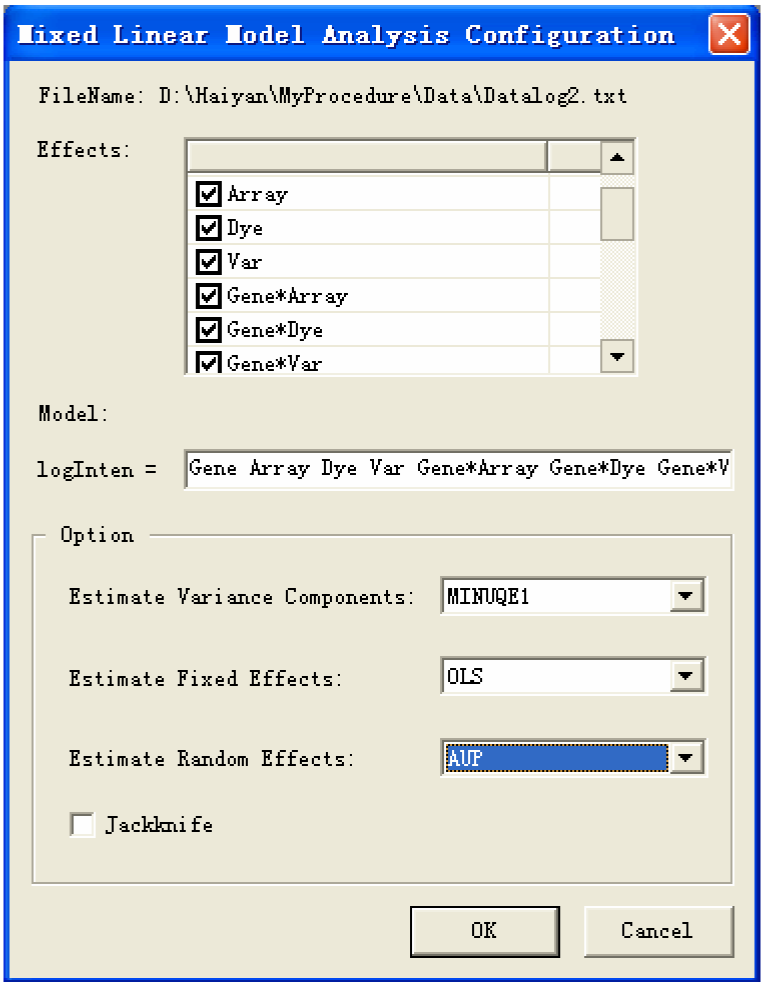
Graph Type: [✓ 3D Graph] 2D Graph

It can extract the first *p* characteristic principal components which are mutually uncorrelated and orthogonal. Each variation of an object can be written exactly as a linear combination of these *p* characteristic principal components. The result is visualized by 2-dimensional or 3-dimesional graph.

See also: PCA

4.12 Mixed Model Approach

- Click **Tools/Mixed Model** menu, you can implement the mixed model analysis on your data. It allows you to input the **expression** of the model used.



FileName: D:\Haiyan\MyProcedure\Data\Datalog2.txt

Effects:

- ✓ Array
- ✓ Dye
- ✓ Var
- ✓ Gene*Array
- ✓ Gene*Dye
- ✓ Gene*Var

Model:

logInten = Gene Array Dye Var Gene*Array Gene*Dye Gene*V

Estimate Variance Components: MINUQE1

Estimate Fixed Effects: OLS

Estimate Random Effects: AUP

Jackknife

Option allows you to estimate variance components and fixed effects, and predict random effects. It offers different methods for estimating the variance components (**MINQUE0**, **MINQUE1** and **REML**) and predicting the random effects (**LUP** and **AUP**) and fixed effects (**OLS** and **GLS**). If selecting **Jackknife** checkbox, it can test for the significance of the estimated variance components and effects.

The standard format of the original data looks like this:

GeneID	TimePoint	Rep	Trait
YAL001C	0 min	1	-1.0
YAL001C	30 min	1	2.0
YAL001C	1 h	1	3.0
YAL002W	0 min	1	-0.6
YAL002W	30 min	1	0.12
YAL002W	1 h	1	-0.36

The column of **Rep** is indispensable whether the experiment has replication or not. If there is no replication, all values of this column are set to one. It can have additional factors in the input file such as dye, treatment or array *et al*. This is tab-delimited text file.

Mixed model approaches are widely used to partition observed phenotypes into various sources of variation. They have the flexibility to handle a wide variety of experimental designs and data shapes including balanced and unbalanced data, and to be easily extended to more complicated biological models.The detailed descriptions of this method can be seen from the literatures (Miler, 1974; Searle, 1992; Zhu, 1994; Zhu and Weir, 1996).

5. Knowledge Base

5.1 Data Preprocessor

5.1.1 Transformation

5.1.1.1 Logarithmic transformation

In the log transformation, the logarithm transformation function is applied to each expression level in the data.

$$y_{ij} = \log_2 x_{ij}$$

or

$$y_{ij} = \log_{10} x_{ij}$$

5.1.1.2 Mean/Median centering

Here the mean or median will be subtracted from each data element:

$$y_{ij} = x_{ij} - \overline{x}_i \qquad \text{or} \qquad y_{ij} = x_{ij} - x_{med\ j}$$

$$\overline{x}_i = \frac{1}{n}\sum_{j=1}^{n} x_{ij} \qquad\qquad x_{med\ j} = \begin{cases} x'_{i(n+1)/2} & n \text{ odd} \\ \frac{1}{2}(x'_{i(n/2)} + x'_{i(n/2)+1}) & n \text{ even} \end{cases}$$

where $\overline{x}_i$ is mean of the vector **x**$_i$ and $x_{med\ j}$ is median of the ascending sorted vector **x**'$_i$.

5.1.1.3 Standardization

The expression vectors are standardized to have mean 0 and standard deviation 1(by subtracting the mean of each row in the data, and then dividing by the standard deviation of the row).

$$y_{ij} = \frac{x_{ij} - \overline{x}_i}{\hat{\sigma}_i}$$

where $\overline{x}_i = \frac{1}{n}\sum_{j=1}^{n} x_{ij}$ and $\hat{\sigma}_i = \sqrt{\frac{1}{n-1}\sum_{j=1}^{n}(x_{ij} - x_i)^2}$

5.1.1.4 Length One

The expression vectors are transformed to have mean 0 and length 1.

$$y_{ij} = \frac{x_{ij} - \overline{x}_i}{\hat{\sigma}_i}$$

where $\bar{x}_i = \frac{1}{n}\sum_{j=1}^{n} x_{ij}$ and $\hat{\sigma}_i = \sqrt{\sum_{j=1}^{n}(x_{ij} - x_i)^2}$

5.1.1.5 Population Center

Here the population mean will be subtracted from each data element:

$$y_{ij} = x_{ij} - \bar{x}$$

$$\bar{x} = \frac{1}{n*m}\sum_{i=1}^{m}\sum_{j=1}^{n} x_{ij}$$

where $\bar{x}$ is the total mean of the population.

5.1.2 Filtering

Due to the large number of genes, the full gene expression data set is usually filtered to reduce the size of the data. Filtering removes genes that do not vary significantly across the experiments and do not have certain desired properties from your dataset. Filtering also facilitates biological interpretation because genes that do not vary significantly across the experiments are usually not of great interest to the biologist. The currently 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 removes all genes that have standard deviation 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.

✧ Max Val−MinVal >= X. This removes all genes whose maximum minus minimum values are less than X.

5.2 Distance Metrics

All clustering algorithms use a dissimilarity or similarity measurement between samples or genes to compare patterns of expression. Let $\mathbf{x}_i = \{x_{i1}, x_{i2}, \cdots, x_{ik}, \cdots, x_{in}\}$ and $\mathbf{x}_j = \{x_{j1}, x_{j2}, \cdots, x_{jk}, \cdots, x_{jn}\}$ be two vectors of expression values for object *i* and object *j*.

5.2.1 Euclidian distance

Euclidian distance is a very commonly used distance measurement. It is basically just the sum of the squared distances of two vector values (x_{ik}, x_{jk}). The Euclidian distance is given by the formula:

$$D(i, j) = \sqrt{\sum_{k=1}^{n}(x_{ik} - x_{jk})^2}$$

Euclidian distance will be variant to both adding and multiplying all components with a constant factor, it is necessary to normalization the dataset to remove the effect of scales before using Euclidian distance. It is also variant to the dimension of the vectors, for example if missing values reduces the dimension of certain vectors.

5.2.2 Manhattan distance

Manhattan distance is the sum of the absolute distances of two vector values (x_{ik}, x_{jk}) on a compound basis.

$$D(i,j)=\sum_{k=1}^{n}|x_{ik}-x_{jk}|$$

Basically this is a linear version of the Euclidian distance, with the same advantages and disadvantages.

5.2.3 Chebyshev distance

Chebyshev distance is the maximum of the absolute distances of two vector values (x_{ik}, x_{jk}).

$$D(i,j)=\max_{1\le k\le n}|x_{ik}-x_{jk}|$$

5.2.4 Canberra distance

$$D(i,j)=\sum_{k=1}^{n}\frac{|x_{ik}-x_{jk}|}{|x_{ik}+x_{jk}|}$$

5.2.5 Bray-Curtis distance

$$D(i,j)=\frac{\sum_{k=1}^{n}|x_{ik}-x_{jk}|}{\sum_{k=1}^{n}|x_{ik}+x_{jk}|}$$

5.2.6 Clarl distance

$$D(i,j)=\sqrt{\frac{1}{n}\sum_{k=1}^{n}\left(\frac{x_{ik}-x_{jk}}{x_{ik}+x_{jk}}\right)^2}$$

5.2.7 Averaged Dot

The simplest measurement of association between two vectors is the inner product. The inner product between two points is defined as the sum of products of components and can be modified by defining an adjusted or ‘average’ value;

$$D(i,j) = \frac{1}{n}\sum_{k=1}^{n} x_{ik}x_{jk}$$

which is independent of the sample size *n*.

5.2.8 Covariance

The covariance distance between two points is defined as follow:

$$Cov(i,j) = \frac{\sum_{k=1}^{n}(x_{ik}-\overline{x}_i)(x_{jk}-\overline{x}_j)}{n-1}$$

Adding or subtracting constant numbers to the vectors does not alter the magnitude of the covariance, but multiplying by constants does. Covariance values are also independent of the sample size *n*, since the numerator is divided by *n* −1.

5.2.9 Distance based on mixed-model

The phenotype value of the *j*th feature for the *i*th object can be expressed by a mixed linear model,

$$y_{ijk} = \mu_i + \alpha_j + \tau_{ij} + \varepsilon_{ijk}$$

where y_{ijk} is the observed value of feature *j* for object *i* in *k* replication; μ_i is the mean expression level of object *i*, fixed effect; $\alpha_j \sim (0,\sigma_\alpha^2)$ is the random effect of feature *j*; $\tau_{ij} \sim (0,\sigma_\tau^2)$ is the interaction between the *i*-th object and the *j*-th feature and the random error term $\varepsilon_{ijk} \sim (0,\sigma_\varepsilon^2)$ is the residual effect.

For any pairs of objects, the similarity between object *i* and object *j* is defined as:

$$\begin{aligned} D^2{}_{(i,j)} &= \frac{1}{2}(\mu_i - \mu_j)^2 + \sigma^2_{\tau(i,j)} \\ &= d^2_{(i,j)} + \sigma^2_{\tau(i,j)} \end{aligned}$$

where $d^2_{(i,j)}$ is half of the squared difference of feature means between object *i* and object *j*, and $\sigma^2_{\tau(i,j)}$ is the object $\times$feature interaction variance for object *i* and object *j*. The squared difference $d^2_{(i,j)}$ is a parameter of marginal means difference and the interaction variation, $\sigma^2_{\tau(i,j)}$ is of the response ‘shape’. $\sigma^2_{\tau(i,j)}$ can be obtained by the ANOVA method for balanced data. For the missing data, $\sigma^2_{\tau(i,j)}$ can be estimated by restricted maximum likelihood estimation (REML) method (Searle *et al.*, 1992) or Minimum Norm Quadratic Unbiased Estimation (MINUQE) method (Rao, 1970).

The similarity between two objects is then estimated by

$$\hat{D}^2_{(i,j)} = \frac{1}{2}(\overline{y}_{i..} - \overline{y}_{j..})^2 + \hat{\sigma}^2_{\tau(i,j)}$$

5.2.10 CosineAngle Correlation

In term of vector components, $\cos\theta$ can be expressed as:

$$\cos\theta = \frac{\sum_{k=1}^{n} x_{ik} x_{jk}}{\sqrt{\sum_{k=1}^{n} x_{ik}^2}\sqrt{\sum_{k=1}^{n} x_{jk}^2}}$$

When $\theta = 0$ both **x*i*** and **x*j*** lie on the same straight line and are thus linearly dependent. For $\theta = 90^\circ$, the vectors are orthogonal, and in general -1≤ $\cos\theta$ ≤1.

5.2.11 Pearson Correlation

Pearson correlation coefficient is a widely used measurement of association between two vectors. Pearson correlation *r* is given by the formula:

$$R(i,j) = \frac{\sum_{k=1}^{n} (x_{ik} - \overline{x}_i)(x_{jk} - \overline{x}_j)}{\sqrt{\sum_{k=1}^{n} (x_{ik} - \overline{x}_i)^2}\sqrt{\sum_{k=1}^{n} (x_{jk} - \overline{x}_j)^2}}$$

where $\overline{x}_i$ is the mean of the vector **x*i*** and $\overline{x}_j$ is the mean of the vector **x*j***. The correlation coefficient is invariant under scalar transformation of the data (adding, subtracting or multiplying the vectors with a constant factor).

5.2.12 Uncentered Pearson correlation

This is basically the same formula as above, except the mean is expected to be 0.

$$R(i,j) = \frac{\sum_{k=1}^{n} x_{ik} x_{jk}}{\sqrt{\sum_{k=1}^{n} (x_{ik} - \overline{x}_i)^2}\sqrt{\sum_{k=1}^{n} (x_{jk} - \overline{x}_j)^2}}$$

Curves with “identical” shape, but different magnitude have an uncentered correlation coefficient < 1, whereas the Pearson correlation coefficient in this case would be 1. If just the magnitude of the curve is important, then the uncentered correlation value is better.

5.2.13 Squared Pearson correlation distance

The squared Pearson correlation coefficient calculates the square of the Pearson correlation coefficient, so that negative values become positive.

$$R(i,j)=\left(\frac{\sum_{k=1}^{n}(x_{ik}-\overline{x}_i)(x_{jk}-\overline{x}_j)}{\sqrt{\sum_{k=1}^{n}(x_{ik}-\overline{x}_i)^2}\sqrt{\sum_{k=1}^{n}(x_{jk}-\overline{x}_j)^2}}\right)^2$$

If the correlation approaches –1, the squared Pearson will approach 1, i.e. a perfect match. In this way, samples or genes with an opposite behavior will be seen as identical. This is especially good for identifying reciprocal expression profiles (e.g. regulatory or inhibitory mechanisms).

5.2.14 Spearman Rank-Order correlation distance

Spearman rank correlation is nonparametric or rank correlations. The key concept of nonparametric correlation is this: If we replace the value of each x_{ik} and x_{jk} by the value of its rank among all the other $\mathbf{x}_i$'s ($\mathbf{x}_j$'s) in the object, that is, 1, 2,. . . , n, then the resulting list of numbers will be drawn from a perfectly known distribution function, namely uniformly from the integers between 1 and n.

Let R_i be the rank of $\mathbf{x}_i$ among the other $\mathbf{x}_i$'s, S_i be the rank of $\mathbf{x}_j$ among the other $\mathbf{x}_j$'s, the *sum squared difference of ranks* is defined as:

$$D=\sum_{i=1}^{n}(R_i-S_i)^2$$

When there are no ties in the data, then the exact relation between D and r_s is

$$R_s(i,j)=1-\frac{6D}{n^3-n}$$

When there are ties, then the exact relation is slightly more complicated: Let f_k be the number of ties in the kth group of ties among the R_i's, and let g_m be the number of ties in the mth group of ties among the S_i's. Then the exact relation between D and r_s is

$$R_s(i,j)=\frac{1-\frac{6}{n^3-n}\left[D+\frac{1}{12}\sum_k(f_k^3-f_k)+\frac{1}{12}\sum_m(g_m^3-g_m)\right]}{\left[1-\frac{\sum_k(f_k^3-f_k)}{n^3-n}\right]^{1/2}\left[1-\frac{\sum_m(g_m^3-g_m)}{n^3-n}\right]^{1/2}}$$

Notice that if all the f_k and all the g_m are equal to one, meaning that there are no ties, the equation (2) reduces to equation (1).

5.2.15 Kendall's Tau correlation distance

Kendall's Tau is even more nonparametric than Spearman's Rank correlation. Instead of using the numerical difference of ranks, it uses only the relative ordering of ranks: higher in rank, lower in rank, or the same in rank. Considering all 1/2n(n-1) pairs of data points, a pair is called as *concordant* if the relative ordering of the ranks of the two $\mathbf{x}_i$'s (or for that matter the two $\mathbf{x}_i$'s themselves) is the same as the relative ordering of the ranks of the two $\mathbf{x}_j$'s (or for that matter the two $\mathbf{x}_j$'s themselves), a pair is called as *discordant* if the relative ordering of the ranks of the two $\mathbf{x}_i$'s is opposite from the relative ordering of the ranks of the two $\mathbf{x}_j$'s. If there is a tie in either the ranks of the two $\mathbf{x}_i$'s or the ranks of the two $\mathbf{x}_j$'s, then the pair is not called as either *concordant* or *discordant*. If the tie is in the $\mathbf{x}_i$'s, the pair is defined as an "extra *y* pair." If the tie is in the *y*'s, the pair is defined as an "extra *x* pair." If the tie is in both the *x*'s and the *y*'s, we don't call the pair anything at all. Kendall'sτ is now the following simple combination of these various counts:

$$\tau(i,j)=\frac{concordant-discordant}{\sqrt{concordant+discordant+extra-y}\sqrt{concordant+discordant+extra-x}}$$

5.3 Clustering Algorithms

Cluster analysis is a useful exploratory technique, which is essential in data mining process for exploring natural structure and identifying interesting distributions and patterns in underlying data. Cluster analysis groups objects based on the information found in the data describing the objects or their relationships. The objective of cluster analysis is to find groups in a given data set so that objects in the same group are more similar to each other and different from the objects in other groups.

5.3.1 Hierarchical clustering

Hierarchical clustering proceeds successively by either merging smaller clusters into larger ones, or by splitting larger clusters. The result of the algorithm is to produce a tree of clusters, ranging from clusters of individual points at the bottom to an all-inclusive cluster at the top. A diagram called a dendrogram which shows how the clusters are related and describes the order in which points are merged (bottom-up view) or clusters are split (top-down view). According to the methods producing the hierarchical tree, hierarchical clustering algorithms can be further divided into agglomerative algorithms and divisive algorithms.

One of the attractions of hierarchical techniques is that they correspond to taxonomies that are very common in the biological sciences, e.g., kingdom, phylum, genus, species etc. Another attractive feature is that hierarchical techniques do not assume any particular number of clusters. Instead any desired number of clusters can be obtained by "cutting" the dendrogram at the proper level and then a clustering of the data items into disjoint groups is obtained. Finally, hierarchical techniques are thought to produce better quality clusters.

5.3.1.1 Agglomerative clustering

Agglomerative clustering starts with the points as individual clusters and iteratively reduces the number of clusters by merging the two most similar objects or clusters, respectively, until only one cluster is remaining. This requires defining the notion of cluster proximity.

The procedures of many hierarchical agglomerative clustering methods can be expressed by the following algorithms, which are known as the Lance-Williams algorithm.

Basic Agglomerative hierarchical Clustering Algorithm

(1) Calculate the distance between all objects and construct the similarity distance matrix. Each object represents its own cluster.

(2) Find the two clusters with the minimum distance to each other and merge.

(3) Update the distance matrix to reflect the distance between the new cluster and the original clusters.

(4) Repeat step (2) and (3) until only a single cluster remains.

The key step of the previous algorithm is the calculation of the distance between two clusters, and this is where the various agglomerative hierarchical techniques differ. Most of the cluster proximities that in the ClusterProject can be viewed as a choice of different parameters (in the Lance_Williams formula) for the distance between clusters *k* and *r*, where *r* is formed by merging cluster *p* and *q*.

$$D_{kr} = \alpha_p D_{kp} + \alpha_q D_{kq} + \beta D_{pq} + \gamma \left| D_{kp} - D_{kq} \right|$$

Table 1 The parameters for four agglomerative clustering algorithms.

Cluster Method	α_p	α_q	β	γ
Single Linkage	1/2	1/2	0	-1/2
Complete Linkage	1/2	1/2	0	1/2
WPGMA Linkage	1/2	1/2	0	0
Average Linkage	n_p / n_r	n_q / n_r	0	0

$$D_{kr}^2 = \alpha_p D_{kp}^2 + \alpha_q D_{kq}^2 + \beta D_{pq}^2 + \gamma \left| D_{kp}^2 - D_{kq}^2 \right|$$

Table 2 The parameters for five agglomerative clustering algorithms.

Cluster Method	α_p	α_q	β	γ
Median Linkage	1/2	1/2	-1/4	0
Centroid Linkage	n_p/n_r	n_q/n_r	$-n_p n_q/n_r^2$	0
Flexible Group Linkage	$(1-\lambda)n_p/n_r$	$(1-\lambda)n_q/n_r$	<1	0
Flexible Linkage	$(1-\lambda)/2$	$(1-\lambda)/2$	<1	0
Ward Linkage	$(n_p+n_k)/(n_r+n_k)$	$(n_p+n_k)/(n_r+n_k)$	$-n_k/(n_r+n_k)$	0

The following are the detail of some of the most commonly used.

Single linkage (nearest neighbor)

In this method, the distance between two clusters is defined to be minimum of the distance between the two points in the different clusters (i.e., by the “nearest neighbors”). If there are several equal minimum distances between clusters during merging, single linkage is the only well defined procedure. Its greatest drawback is the tendency for chain building: Only one (random) small distance is enough to enforce the amalgamation of two otherwise very different clusters. Therefore, the resulting clusters tend to represent long "chains". Single linkage is good at handling non-elliptical shapes, but is sensitive to noise and outliers.

Complete linkage (furthest neighbor)

In this method, the distance between two clusters is determined by the maximum distance between any two points in the different clusters (i.e., by the "furthest neighbors"). Complete linkage usually performs quite well in cases when the objects actually form naturally distinct data clouds in the multidimensional space. If the clusters tend to be somehow elongated or to be a "chain" type nature, then this method is inappropriate. Since only one (random) large distance is enough to merge two clusters, clusters tend to be small. Complete linkage is less susceptible to noise and outliers, but can break large clusters, and has trouble with convex shapes.

Unweighted pair-group average linkage

In this method, the distance between two clusters is calculated as the average distance of the pair-wise distances between all pairs of points in the two different clusters. This method is very efficient when the objects form natural distinct "clumps," however, it performs equally well with elongated, "chain" type clusters. Since the distance between two clusters lies between the minimum formation of single linkage and the maximum formation of complete linkage, this procedure empirically shows no tendencies to either extreme described above, and is therefore more stable to unknown data point distributions. Admittedly, if there are several equal distances, the sequence of the amalgamation is critical. Note that the abbreviation UPGMA is used as well to refer to this method as unweighted pair-group method using arithmetic averages.

Weighted pair-group average linkage

This method is identical to the unweighted pair-group average method, except that in the computations, the size of the respective clusters (i.e., the number of objects contained in them) is used as a weight. Thus, this method (rather than the previous method) should be used when the cluster sizes are suspected to be greatly uneven.

Centroid linkage

In this method, the proximity between two clusters is calculated as the distance between two centroids of two clusters. Centroid linkage has a characteristic—often considered bad that other hierarchical clustering techniques don't posses: the possibility of inversions. In other words, two clusters that are merged may be more similar that the pair of clusters that were merged in a previous step. For other methods, the similarity of clusters merged decreases (the distance between merged clusters increases) as proceeding from singleton clusters to one inclusive cluster.

Ward's method

This method is distinct from all other methods because it uses an analysis of a variance approach to evaluate the distances between clusters. This method attempts to minimize the Sum of Squares of any two (hypothetical) clusters that can be formed at each step. In general, this method is regarded as very efficient and tends to create equally sized clusters of small size.

Time and Space Complexity

Hierarchical clustering techniques typically use a proximity matrix. This requires the computation and storage of m^2 proximities, a factor that limits the size of data sets that can be processed. Once the distance matrix is available, the time required for hierarchical clustering is $O(m^2)$.

Limitations and problems

We summarize the problems with agglomerative hierarchical clustering:

- ✧ No global objective function is being optimized.
- ✧ An incorrect merging of clusters in an early stage often yields results, which are far away from

the real cluster structure.

- ✧ Good local merging decisions may not result in good global results.
- ✧ Agglomerative hierarchical clustering techniques have trouble with one of more of the following: Noise and outliers, non-convex shapes, and a tendency to break large clusters.

5.3.1.2 Divisive clustering

Divisive clustering starts with one, all-inclusive cluster and, at each step, a biggest cluster is split into two smaller clusters until each cluster contains only a single sample. In this case, one should decide which cluster to split and how to split the bigger one into two smaller ones at each step.

In divisive procedures, fundamentally all subsets have to be analyzed so that divisive procedures have an algorithmic complexity in the magnitude of O (2^n). Divisive procedures immediately start with interesting cluster arrangements and are much more robust.

Here we introduce a robust divisive clustering algorithm—*Diana*.

Basic *Diana* Clustering Algorithm

1) First time the selected cluster C is the whole data. To divide the selected cluster C with cardinality *n*(C) into two subgroups, the algorithm first looks for its most disparate observation (i.e., which has the largest average dissimilarity to the other observations of the selected cluster). That is, for each member $x_1 \in$C, one computes the 'distance' from the rest of C by

$$D_{1,x_1} = (n(C)-1)^{-1} \sum_{y \in C \setminus \{x_1\}} d(x_1, y)$$

and identifies most disparate observation x_1^* which has largest D_1,x_1.

2) This disparate observation initiates the "splinter group". In subsequent steps, the algorithm reassigns observations that are closer to the "splinter group" than to the "old party", the observations in the "splinter group" are noted as $x_1^*, \ldots, x_{k+1}^*$. This procedure continues iteratively till $D_{k+1}, x_{k+1}^* < 0$ where

$$D_{k+1,x_{k+1}} = (n(c)-k-1)^{-1} \sum_{y \in C \setminus \{x_1^*, \ldots, x_k^*\}} d(x_{k+1}, y) - k^{-1} \sum_{i=1}^{k} d(x_i^*, x_{k+1})$$

and x_{k+1}^* maximizes D_{k+1}, x_{k+1}. The result is a division of the selected cluster C into two new clusters $\{x_1^*, \ldots, x_{k+1}^*\}$ and C$\setminus\{x_1^*, \ldots, x_{k+1}^*\}$.

3) The cluster with the largest diameter is selected. The diameter of a cluster is the largest dissimilarity between any two of its observations. The diameter is defined as:

$$Diameter(s) = \max_{x,y \in S}\{d(x,y)\}$$

4) Continue Step 1, 2 and 3 until each cluster contains only a single observation.

5.3.2 Partition clustering

Partition clustering attempts to directly decompose the data set into a set of disjoint clusters. More specifically, they attempt to determine an integer number of partitions that optimize a certain objective function. The objective function may emphasize the local or global of the data and its optimization is an iterative procedure.

5.3.2.1 K-Means

The K-Means is a very simple clustering method because it is based on a very simple principle and provides good results.

Basic K-Means Algorithm for finding K clusters

1) Select k points as the initial centroids.
2) Assign all points to the closest centroid.
3) Re-compute the centroid of each cluster.
4) Repeat step 2 and 3 until the centroids don't change.

The k-means algorithm has the following important properties:

1) It is efficient in processing large data sets, due to the fact that the computational complexity of the algorithm is O (tkmn), where n is the number of points, m is the dimension of the points **x**, k is the number of clusters and t is the number of iterations over the whole data set. Usually, $k, m, t << n$. It takes usually just a few seconds to calculate even datasets with 10000 elements and more, making it a valuable tool for the investigation of datasets that are too big for hierarchical clustering for instance.
2) Another big advantage is the moderate memory requirement for k-means clustering. Since there is no similarity matrix to calculate the memory requirements rise with $O(n)$.

The k-means algorithm has the following limitations and problems:

1) It is difficult to discover correct clusters with non-convex shapes and wide different sizes
2) It often terminates at a local optimum. The K-Means objective function is minimized by globular clusters of equal size or by clusters that are well separated.
3) The k-Means algorithm is also normally restricted to data in Euclidean spaces because in many cases the required means and medians do not make sense.
4) The major drawback of the k-means algorithm is that the number of clusters has to be specified in advance and seriously affected by the choice of the initial centroids.

5.3.2.2 K-Medoid

In this project, K-Medoid is implemented a slight variation on K-Means clustering in which the median instead of the centroid of items are used. Obviously, K-Medoid has similar properties and problems. It computes more expensively than K-Means.

5.3.2.3 PAM (Partitioning Around Medoids)

The objective of PAM clustering is to find a non-overlapping set of clusters such that each cluster has a most representative point, i.e., appoint that is most centrally located with respect to some measure. These representative points are called medoids.

Basic PAM Algorithm for finding K clusters

(1) Select K initial points. These points are the candidate medoids and are intended to be the most central points of their clusters.

(2) Consider the effect of replacing one of the selected objects (medoids) with one of the non-selected objects. The distance of each-selected point from the closest candidate medoid is calculated, and this distance is summed over all points. This distance represents the “cost’ of the current configuration. All possible swaps of a non-selected point for a selected one are considered, and the cost of each configuration is calculated.

(3) Select the configuration with the lowest cost. If this is a new configuration, then repeat step 2.

(4) Otherwise, associate each non-selected point with its closest selected point (medoid) and stop.

The PAM approach is not restricted to Euclidean spaces and is likely to be more tolerant of outliers. However, finding a better medoid requires trying all points that are currently not medoids and is computationally expensive.

5.3.2.4 CAST

CAST uses as input the similarity matrix *S*. The affinity of an element *v* to a putative cluster *C* is

$$a(v) = \sum_{i \in C} S(i, v).$$

The algorithm uses a single parameter *t*, clusters are generated one by one, the next cluster is started with a single element, and elements are added or removed from the cluster if their relative affinity is larger or lower than *t*, respectively, until the process stabilizes.

Basic CAST Algorithm for finding K clusters

While there are unclustered elements do:

Repeat **ADD** and **REMOVE** until no changes occur:

ADD: add an unclustered element v with maximum affinity to C if a(v) > t|C|.

REMOVE: remove an element u form C with minimum affinity if a(u) <= t|C|.

Add C to the list of final clusters.

5.3.3 Genetic clustering

The traditional clustering methods, such as agglomerative or divisive hierarchical and partition clustering, use a greedy algorithm, it looks observations into a particular cluster that is deemed best at that point in the algorithm but may not be the best globally when all information is considered. Recently, the use of global optimization techniques such as *Simulated Annealing* and *Genetic Algorithms* (GAs) has emerged in the clustering fields (Cowgill et al., 1999; Maulik and Bandyopadhyay, 2000).

Genetic Algorithms (GAs) introduced by Holland (1962) are randomized search and optimization techniques that guided by the principle of evolution and natural genetics. Because they are aided by large amounts of implicit parallelism (Grefenstettee and Baker, 1989), GAs are capable of searching for optimal or near-optimal solutions on complex, large spaces of possible solutions. Furthermore, GAs allows searching of these spaces of solutions by simultaneously considering multiple interacting attributes. Because of these advantages, GAs may represent another useful tool in the classification of biological phenotypes based on gene expression data.

5.3.3.1 HGACLUS

The detailed description is described by Pan *et al*. (2003).

5.3.3.2 KGACLUS

The detailed description is described by Maulik and Bandyopadhyay (2000).

5.3.3.3 COWCLUS

The detailed description is described by Cowgill *et al*. (1999).

5.3.4 Cluster Validation

Cluster validation refers to procedures that evaluate the results of cluster analysis in a quantitative and objective fashion. In the statistics literature, cluster validation procedures are divided cluster into two main categories: external and internal criterion analysis.

External criterion analysis validates a clustering result by comparing it to a given “gold standard” which is another partition of the objects. The gold standard must be obtained by an independent process based on information other than the given data set. There are many statistical measures that assess the agreement between an external criterion and a clustering result. For example, (Milligan *et al.*, 1983) and (Milligan and Cooper, 1986) evaluated the performance of different clustering algorithms and different statistical measures of agreement on both synthetic and real data.

Internal criterion analysis uses information from within the given data set to represent the goodness of fit between the input data set and the clustering results. For example, compactness and isolation of clusters are possible measures of goodness of fit.

For validation of clustering results, external criterion analysis has the strong benefit of providing an independent, hopefully unbiased assessment of cluster quality. On the other hand, external criterion analysis has the strong disadvantage that an external gold standard is rarely available. Internal criterion analysis avoids the need for such a standard, but has the alternative problem that clusters are validated using the same information from which clusters are derived. Different clustering algorithms optimize different objective functions or criteria. Assessing the goodness of fit between the input data set and the resulting clusters is equivalent to evaluating the clusters under a different objective function.

5.3.4.1 Internal Indices

Variance Ratio Criteria

Variance ratio criteria is defined as the following formula:

$$VRC = \frac{\text{trace }(\mathbf{B})/(k-1)}{\text{trace }(\mathbf{W})/(n-k)}$$

where n and k are the total number of points and the number of clusters in the partition, respectively. **B** and **W** are the covariance matrices of between and the within k-clusters sums of squares. VRC has intuitive appeal to express what constitutes ‘true’ clusters structures. VRC measures the degree of separation between clusters and homogeneity within clusters. Hence, a better clustering algorithm is expected to have a relatively larger VRC value.

Figure of Merit

FOM is defined as the root mean squared deviation in the left-out experimental condition of the individual gene expression levels relation to their cluster means. Assume that a clustering algorithm is applied to the data from experimental conditions 1, 2,..., (j-1), (j+1),..., n and condition j is used to estimate the predictive power of the algorithm. Let there be K clusters, C_1, $C_2, \ldots, C_k$. Let $x(g, j)$ be the expression level of gene g under condition j in the raw data matrix. Let $U_{C_i}(j)$ be the average expression level in condition j of genes in cluster C_i. The FOM under the condition j is defined as:

$$FOM(j,k) = \sqrt{\frac{1}{N}\sum_{i=1}^{k}\sum_{g\in C_i}[x(g,j)-U_{C_i}(j)]^2}$$

where *N* is the total number of genes. FOM (*j*, *k*) measures the means squared error of predicting the expression levels from the average cluster expression level in experiment *j*. Hence, a relatively small figure of merit indicates a *clustering* algorithm having relatively high predictive power

Each of the *n* experiments can be used as the validation experiment. The aggregate figure of merit of all *conditions* is defined as:

$$FOM\,(k) = \sum_{j=1}^{n} FOM\,(j,k)$$

FOM (*k*) is an *estimate* of the total predictive power of the algorithms over all the experiments for *K* clusters in a data set and FOM (*k*) decreases as the number of clusters increases.

Homogeneity and Separation

Homogeneity is calculated as the average distance between each object and the center of the cluster it belongs to. The formula can be described as following:

$$H_{avg} = \frac{1}{n}\sum_{i=1}^{k}\sum_{j=1}^{n_i} D(x_j, z_k)$$

where x_j is the *j*th object and z_k is the center of the cluster that x_j belongs to; *n* is the total number of objects; *D* is the *distance* function.

Separation is calculated as the weighted average distance between cluster centers:

$$S_{avg} = \frac{1}{\sum_{i \neq j} N_{C_i} N_{C_j}} \sum_{i \neq j} N_{C_i} N_{C_j} D(C_i, C_j)$$

where C_i and C_j are the centers of *i*th and *j*th clusters, and Nc_i and Nc_j are the number of objects in the *i*th and *j*th clusters. Thus H_{avg} reflects the compactness of the clusters while S_{avg} reflects the overall distance between clusters. Decreasing H_{avg} or increasing S_{avg} suggests an improvement in the clustering results.

Silhouette Width

Silhouette width is a composite index reflecting the compactness and separation of the clusters, and can be applied to different distance metrics. For each object *i*, its *silhouette width s(i)* is defined as:

$$s(i) = \frac{a(i) - b(i)}{\max\left(a(i), b(i)\right)}$$

where *a(i)* is the average distance of object *i* to other objects in the same cluster, *b(i)* is the average distance of object *i* to objects in its nearest neighbor cluster. The average of *s(i)* across all objects reflects the overall quality of the clustering result. A larger *averaged silhouette width* indicates a better overall quality of the clustering result.

5.3.4.2 External indices

An external index is defined as a measure of agreement between a clustering result and a given "gold" standard. Since the true cluster labels are available for all of the real data sets we used, we are able to evaluate the ability of a clustering procedure to recover true cluster labels using the external criteria.

Consider two partitions of n objects $x_1, \ldots, x_n$: the R-class partition U={ $u_1, \ldots, u_n$}and the C-class partition V={$v_1, \ldots, v_n$}. Externals indices of partition agreement can be expressed in the terms of a contingency table, with entry n_i denoting the number of objects that are both in clusters u_i and v_j , i=1, . . . , R, j=1, . . . , C.

Table 3 Contingency table for two partitions of n objects

	v_1	v_2	$\cdots$	v_C	
u_1	n_{11}	n_{12}	$\cdots$	n_{1C}	$n_{1.}$
u_2	n_{21}	n_{22}	$\cdots$	n_{2C}	$n_{2.}$
$\vdots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$
u_R	n_{R1}	n_{R2}	$\cdots$	n_{RC}	$n_{R.}$
	$n_{.1}$	$n_{.2}$	$\cdots$	$n_{.C}$	$n_{..}=n$

Where $n_{i.} = \sum_{j=1}^{C} n_{ij}$ and $n_{.j} = \sum_{i=1}^{R} n_{ij}$ denote the row and column sums of the contingency table, respectively and let $Z = \sum_{i=1}^{R} \sum_{j=1}^{C} n_{ij}^2$.

The following indices can then be used.

1) **Adjusted Rand index**: Hubert and Arabie (1985)

$$Rand = \frac{\sum_{i=1}^{R}\sum_{j=1}^{C}\binom{n_{ij}}{2} - \left[1/\binom{n}{2}\right]\sum_{i=1}^{R}\binom{n_{i.}}{2}\sum_{j=1}^{C}\binom{n_{.j}}{2}}{(1/2)\left[\sum_{i=1}^{R}\binom{n_{i.}}{2} + \sum_{j=1}^{C}\binom{n_{.j}}{2}\right] - \left[1/\binom{n}{2}\right]\sum_{i=1}^{R}\binom{n_{i.}}{2}\sum_{j=1}^{C}\binom{n_{.j}}{2}}$$

2) **Jaccard**: Jain and Dubes (1988)

$$Jaccard = (Z-n)/\left(\sum_{i=1}^{R} n_{i.}{}^2 + \sum_{j=1}^{C} n_{.j}{}^2 - Z - n\right)$$

3) **FM**: Fowlkes and Mallows (1983)

$$FM = (1/2)(Z-n)/\left[\sum_{i=1}^{R}\binom{n_{i.}}{2}\sum_{j=1}^{C}\binom{n_{.j}}{2}\right]^{1/2}$$

5.4 Principle Component Analysis (PCA)

Principal Component Analysis (PCA), also known as Singular Value Decomposition (SVD) is an exploratory multivariate statistical technique that allows the identification of key variables (or combinations of variables) in a multidimensional data set that best explains the differences between observations. Given *m* observations (experiments) on *n* variables (genes), the goal of PCA is to reduce the dimensionality of the data matrix by finding $r \leq n$ new variables. These *r* principal components account together for as much of the variance in the original *n* variables as possible while remaining mutually uncorrelated and orthogonal. The goal is to reduce dimensionality while filtering noise in the process, making the data more accessible for visualization and analysis.

5.4.1 Mathematical background

Consider *m* observations on *n* random variables represented by the matrix **X**. **D** is a distance matrix of the input matrix **X**. Let **P** denote a *$(m \times m)$* matrix of unknown coefficients such that the quadratic form $\mathbf{P}^T\mathbf{D}\mathbf{P}$ is maximized subject to the constraint $\mathbf{P}^T\mathbf{P} = \mathbf{I}$. This is equivalent to maximizing the Lagrangean expression:

$$\Phi = P^T D P - \lambda I (P^T P - I)$$

where λ**I** is a diagonal matrix of Lagrange multipliers (Eigenvalues). Differentiating with respect to **P** and setting the equation to zero we are receiving

$$\frac{\partial \Phi}{\partial P} = 2D - 2\lambda P = 0 \qquad (1)$$

The normal equations in (1) yield estimates for Eigenvalues and Eigenvectors. To compute the principal components, the *m* Eigenvalues and their corresponding Eigenvectors are calculated from the *$(m \times m)$* distance matrix **D** using for example Singular Value Decomposition (SVD). When **D** is nonsingular, all latent roots are strictly positive and each Eigenvector defines a principal component.

SVD methods are based on the following theorem of linear algebra: any *$(n \times m)$* matrix **X** whose number of rows *n* is greater than or equal to its number of columns *m*, can be written as the product of a *$(n \times m)$* column-orthogonal matrix **U**, a *$(m \times m)$* diagonal matrix **W** with positive or zero elements (the singular values), and the transpose of an *$(m \times m)$* orthogonal matrix **V**.

$$X = U\,W\,V^T$$

SVD now explicitly constructs orthonormal bases for the null space and range of a matrix. Specifically, the columns of **U** whose same-numbered elements w_j are nonzero are an orthonormal set of basis vectors that span the range; the columns of **V** whose same-numbered elements *wj* are zero are an orthonormal basis for the null space.

The matrices **U** and **V** are each orthogonal in the sense that their columns are orthonormal.

$$\mathbf{U}^T\mathbf{U} = \mathbf{V}^T\mathbf{V} = \mathbf{V}\mathbf{V}^T = \mathbf{I}$$

The vectors of **U** contain our Eigenvectors and the diagonal elements of **W** contain the corresponding Eigenvalues. Now the Eigenvectors of **U** are ordered regarding the value of their corresponding Eigenvalues. Each Eigenvector defines a principal component. Principal Component 1 (PC1) is the Eigenvector with the greatest Eigenvalue; PC2 is the Eigenvector with the second largest Eigenvalue and so on.

Since **U** is an orthonormal matrix, it can be seen as a Transformation matrix, which transforms a vector from the input space into the space spanned by the Principal Components.

$$\mathbf{Y} = \mathbf{X}\mathbf{U} \qquad (2)$$

Each component can be viewed as a weighted sum of conditions, where the coefficients of the Eigenvectors are the weights. The projection of object *i* along the axis defined by the *j*th principal component is:

$$y_{ij} = \sum_{t=1}^{m} x_{it} u_{tj}$$

Where u_{tj} is the *t*th coefficient for the *j*th principal component; x_{it} is the value for object *i* under the *t*th feature. **Y** represents the data in terms of principal components and is a rotation of the data from the original space of observations to a new space with principal component axes (PC Space). The variance accounted for by each of the components is its associated Eigenvalue; it is the variance of a component over all genes. Consequently, the Eigenvectors with large Eigenvalues are the ones that contain most of the information; Eigenvectors with small Eigenvalues are uninformative.

5.4.2 Visualization

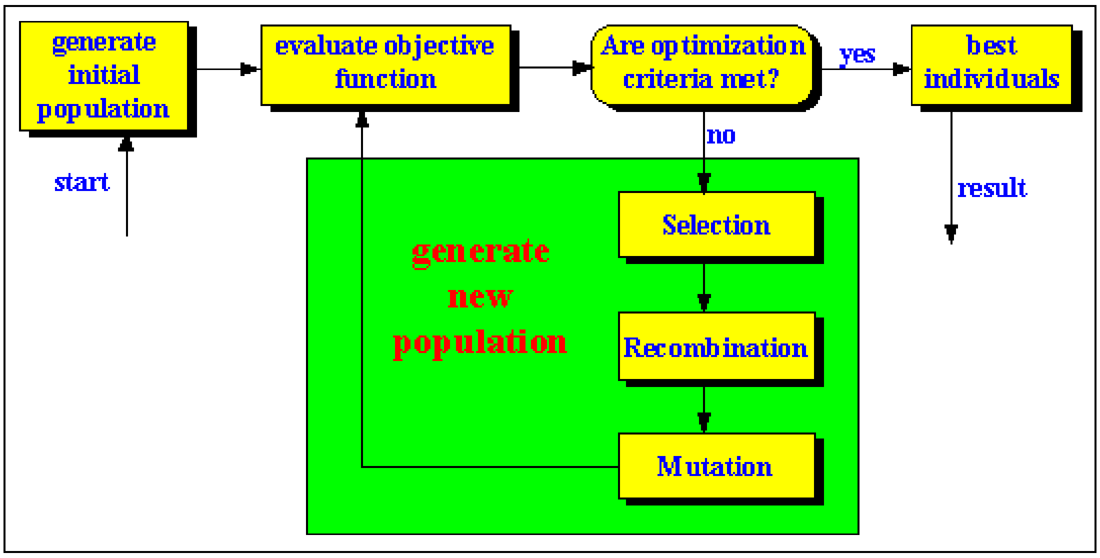
Each variation of a object can be written exactly as a linear combination of these *p* characteristic principal components. If we apply the first three principal components, then we can now use a 3-dimensional coordinate system, where the x-axis represents the PC1, the y-axis PC2 and the z-axis PC3. Plotted in this space is the rotated point y_{ij}. The position of y_{ij} in Principal Component Space gives us information of with patterns the specific objects consists. That in turn means that points near to each other in the Principal Component Space are composed of the same basic patterns and are therefore similar in profile.

5.5 Genetic Algorithm

Genetic algorithms are general-purpose search algorithms based upon the principles of evolution observed in nature. Genetic algorithms combine **selection**, **crossover**, and **mutation** operators with the goal of finding the best solution to a problem. Genetic algorithms search for this optimal solution until a specified termination criterion is met.

Genetic algorithms operate on a population of potential solutions applying the principle of survival of the fittest to produce better and better approximations to a solution. At each generation, a new set of approximations is created by the process of selecting individuals according to their level of fitness in the problem domain and breeding them together using operators borrowed from natural genetics. This process leads to the evolution of populations of individuals that are better suited to their environment than the individuals that they were created from, just as in natural adaptation.

Figure 1 The structure of a simple genetic algorithm



.Evolutionary algorithms work on populations of individuals instead of single solution. In this way the search is performed in a parallel manner.

5.5.1 Selection

Selection is a genetic operator that chooses a chromosome from the current generation's population for inclusion in the next generation's population. Before making it into the next generation's population, selected chromosomes may undergo crossover and / or mutation (depending upon the probability of crossover and mutation) in which case the offspring chromosome(s) are actually the ones that make it into the next generation's population.

5.5.1.1 Rank-based fitness selection

In rank-based fitness assignment, the population is sorted according to the objective values. The fitness assigned to each individual depends only on its position in the individuals rank and not on the actual objective value. Rank-based fitness assignment overcomes the scaling problems of the proportional fitness assignment. The reproductive range is limited, so that no individuals generate an excessive number of offspring. Ranking introduces a uniform scaling across the population and provides a simple and effective way of controlling selective pressure.

5.5.1.2 Roulette wheel selection

The simplest selection scheme is roulette-wheel selection, also called stochastic sampling with replacement. The individuals are mapped to contiguous segments of a line, such that each individual's segment is equal in size to its fitness. A random number is generated and the individual whose segment spans the random number is selected. The process is repeated until the desired number of individuals is obtained (called mating population). This technique is analogous to a roulette wheel with each slice proportional in size to the fitness, see figure 2.

Table 1 shows the selection probability for 11 individuals. Individual 1 is the most fit individual and occupies the largest interval, Individual 11, the least fit interval, has a fitness value of 0 and get no chance for reproduction.

Table 1 Selection probability and fitness value

Number of individual	1	2	3	4	5	6	7	8	9	10	11
fitness value	2.0	1.8	1.6	1.4	1.2	1.0	0.8	0.6	0.4	0.2	0.0
selection probability	0.18	0.16	0.15	0.13	0.11	0.09	0.07	0.06	0.03	0.02	0.0

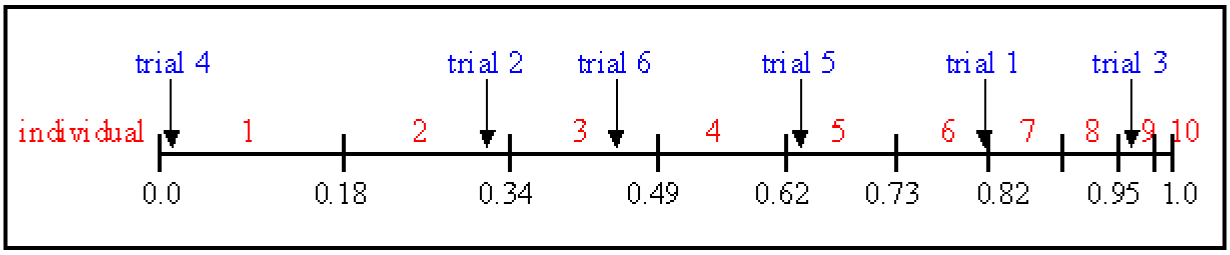
For selecting the mating population the appropriate number of uniformly distributed random numbers (uniform distributed between 0.0 and 1.0) is independently generated.

Sample of 6 random numbers:

0.81, 0.32, 0.96, 0.01, 0.65, 0.42

Figure 2 shows the selection process of the individuals for the example in table 1 together with the above sample trials.

Figure 2 Roulette-wheel selection



After selection the mating population consists of the individuals: 1, 2, 3, 5, 6, 9

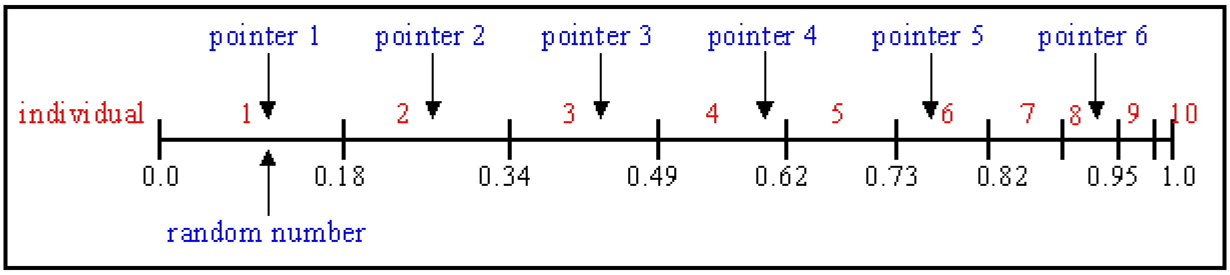
The roulette-wheel selection algorithm provides a zero bias but does not guarantee minimum spread.

5.5.1.3 Stochastic universal sampling

Stochastic universal sampling provides zero bias and minimum spread. The individuals are mapped to contiguous segments of a line, such that each individual's segment is equal in size to its fitness exactly as in roulette-wheel selection. Here equally spaced pointers are placed over the line as many as there are individuals to be selected. Consider *NPointer* the number of individuals to be selected, then the distance between the pointers are 1/*NPointer* and the position of the first pointer is given by a randomly generated number in the range [0, 1/*NPointer*].

For 6 individuals to be selected, the distance between the pointers is 1/6=0.167.

Figure 3 Stochastic universal sampling



After selection the mating population consists of the individuals: 1, 2, 3, 4, 6, 8.

Stochastic universal sampling ensures a selection of offspring which is closer to what is deserved then roulette wheel selection.

5.5.1.4 Truncation selection

Compared to the previous selection methods modeling natural selection truncation selection is an artificial selection method. It is used by breeders for large populations/mass selection.

In truncation selection individuals are sorted according to their fitness. Only the best individuals are selected for parents. These selected parents produce uniform at random offspring. The parameter for truncation selection is the truncation threshold *Trunc*. *Trunc* indicates the proportion of the population to be selected as parents and takes values ranging from 50%-10%. Individuals below the truncation threshold do not produce offspring.

5.5.1.5 Tournament selection

In tournament selection a number *Tour* of individuals is chosen randomly from the population and the best individual from this group is selected as parent. This process is repeated as often as individuals to choose. These selected parents produce uniform at random offspring. The parameter for tournament selection is the tournament size *Tour*. *Tour* takes values ranging from 2 -

Nind (number of individuals in population).

5.5.2 Crossover

Crossover is a genetic operator that combines (mates) two chromosomes (parents) to produce a new chromosome (offspring). The idea behind crossover is that the new chromosome may be better than both of the parents if it takes the best characteristics from each of the parents. Crossover occurs during evolution according to a user-definable crossover probability.

5.5.2.1 Real valued recombination

Discrete recombination

Discrete recombination performs an exchange of variable values between the individuals. Consider the following two individuals with 3 variables each (3 dimensions), which will also be used to illustrate the other types of recombination:

Individual 1	12	25	5
Individual 2	123	4	34

For each variable the parent who contributes its variable to the offspring is chosen randomly with equal probability.

Sample 1	2	2	1
Sample 2	1	2	1

After recombination the new individuals are created:

Offspring 1	123	4	5
Offspring 2	12	4	5

Discrete recombination generates corners of the hypercube defined by the parents. Discrete recombination can be used with any kind of variables (binary, real or symbols).

Intermediate recombination

Intermediate recombination is a method only applicable to real variables (and not binary variables). Here the variable values of the offspring are chosen somewhere around and between the variable values of the parents.

Offspring are produced according to the rule:

Offspring = parent 1 + Alpha (parent 2 - parent 1)

where Alpha is a scaling factor chosen uniformly at random over an interval [-d, 1 + d]. In intermediate recombination d = 0, for extended intermediate recombination d > 0. A good choice is d = 0.25. Each variable in the offspring is the result of combining the variables according to the above expression with a new Alpha chosen for each variable.

Consider the following two individuals with 3 variables each:

Individual 1	12	25	5
Individual 2	123	4	34

The chosen Alphas for this example are:

Sample 1	0.5	1.1	-0.1
Sample 2	0.1	0.8	0.5

The new individuals are calculated as:

Offspring 1	67.5	1.9	2.1
Offspring 2	23.1	8.2	19.5

Intermediate recombination is capable of producing any point within a hypercube slightly larger than that defined by the parents.

Line recombination

Line recombination is similar to intermediate recombination, except that only one value of Alpha for all variables is used:

Individual 1	12	25	5
Individual 2	123	4	34

The chosen Alphas for this example are:

Sample 1	0.5
Sample 2	0.1

The new individuals are calculated as:

Offspring 1	67.5	14.5	19.5
Offspring 2	23.1	22.9	7.9

Line recombination can generate any point on the line defined by the parents.

5.5.2.2 Binary valued recombination (crossover)

Single-point crossover

In single-point crossover one crossover position *k* [*1,2,...,Nvar*-1], *Nvar*: number of variables of an individual, is selected uniformly at random and the variables exchanged between the individuals about this point, then two new offspring are produced. Figure 6 may illustrate this process.

Consider the following two individuals with 11 binary variables each:

Individual 1	0	1	1	1	0	0	1	1	0	1	0
Individual 2	1	0	1	0	1	1	0	0	1	0	1

The chosen crossover position is:

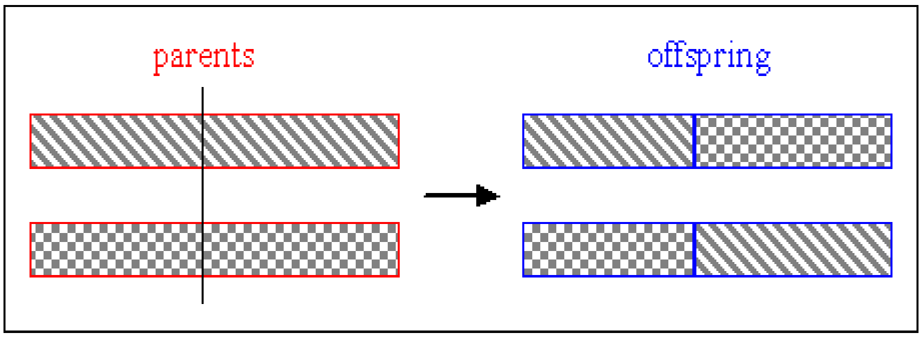
Crossover position: 5

After crossover the new individuals are created:

offspring 1 0 1 1 1 0| 1 0 0 1 0 1

offspring 2 1 0 1 0 1| 0 1 1 0 1 0

Figure 4 Single-point crossover



Multi-point crossover

For multi-point crossover, *m* crossover positions k_i [*1,2,...,*$N_{var\text{-}1}$], *i*=1:*m*, *Nvar*: number of variables of an individual, are chosen at random with no duplicates and sorted in ascending order. Then, the variables between successive crossover points are exchanged between the two parents to produce two new offspring. The section between the first variable and the first crossover point is not exchanged between individuals. Figure 7 may illustrate this process.

Consider the following two individuals with 11 binary variables each:

Individual 1 0 1 1 1 0 0 1 1 0 1 0

Individual 2 1 0 1 0 1 1 0 0 1 0 1

The chosen crossover positions are:

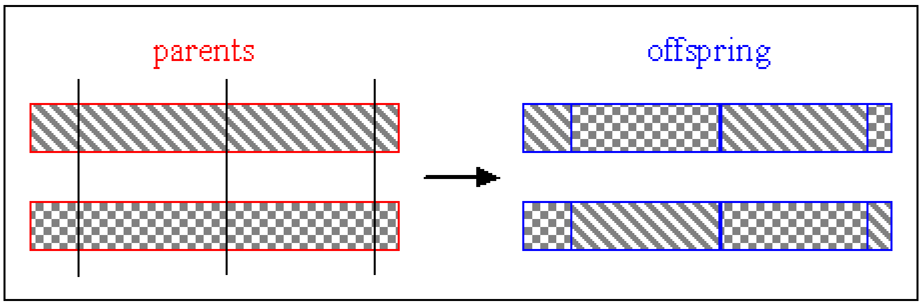
Cross pos. (m=3) 2 6 10

After crossover the new individuals are created:

Offspring 1 0 1| 1 0 1 1| 0 1 1 1| 1

Offspring 2 1 0| 1 1 0 0| 0 0 1 0| 0

Figure 5 Multi-point crossover



The idea behind multi-point, and indeed many of the variations on the crossover operator, is that parts of the chromosome representation that contribute to the most to the performance of a particular individual may not necessarily be contained in adjacent substrings. Further, the disruptive nature of multi-point crossover appears to encourage the exploration of the search space, rather than favoring the convergence to highly fit individuals early in the search, thus making the search more robust.

Uniform crossover

Single and multi-point crossover defines cross points as places between loci where a individual can be split. Uniform crossover generalizes this scheme to make every locus a potential crossover point. A crossover mask, the same length as the individual structure is created at random and the parity of the bits in the mask indicates which parent will supply the offspring with which bits.

Consider the following two individuals with 11 binary variables each:

Individual 1	0	1	1	1	0	0	1	1	0	1	0
Individual 2	1	0	1	0	1	1	0	0	1	0	1

For each variable the parent who contributes its variable to the offspring is chosen randomly with equal probability. Here, the offspring 1 is produced by taking the bit from parent 1 if the corresponding mask bit is 1 or the bit from parent 2 if the corresponding mask bit is 0. Offspring 2 is created using the inverse of the mask, usually.

Sample 1	0	1	1	0	0	0	1	1	0	1	0
Sample 2	1	0	0	1	1	1	0	0	1	0	1

After crossover the new individuals are created:

Offspring 1	1	1	1	0	1	1	1	1	1	1	1
Offspring 2	0	0	1	1	0	0	0	0	0	0	0

Uniform crossover, like multi-point crossover, has been claimed to reduce the bias associated with the length of the binary representation used and the particular coding for a given parameter set. This helps to overcome the bias in single-point crossover towards short substrings without requiring precise understanding of the significance of the individual bits in the individual representation. The algorithm of uniform crossover is identical to discrete recombination.

8.2.2.4 Shuffle crossover

Shuffle crossover is related to uniform crossover. A single crossover position (as in single-point crossover) is selected. But before the variables are exchanged, they are randomly shuffled in both parents. After recombination, the variables in the offspring are unshuffled. This removes positional bias as the variables are randomly reassigned each time crossover is performed.

The binary valued crossover methods are all applicable to the real variables.

5.5.3 Mutation

Mutation is a genetic operator that alters one ore more gene values in a chromosome from its initial state. This can result in entirely new gene values being added to the gene pool. With these new gene values, the genetic algorithm may be able to arrive at better solution than was previously possible. Mutation is an important part of the genetic search as help helps to prevent the population from stagnating at any local optima. Mutation occurs during evolution according to a user-definable mutation probability. This probability should usually be set fairly low (0.01 is a good first choice). If it is set to high, the search will turn into a primitive random search

5.5.3.1 Flip bit mutation

A mutation operator simply inverts the value of the chosen gene (0 goes to 1 and 1 goes to 0). This mutation operator can only be used for binary.

5.5.3.2 Boundary mutation

A mutation operator replaces the value of the chosen gene with either the upper or lower bound for that gene (chosen randomly). This mutation operator can only be used for integer and float genes.

5.5.3.3 Uniform mutation

A mutation operator replaces the value of the chosen gene with a uniform random value selected between the user-specified upper and lower bounds for that gene. This mutation operator can only be used for integer and float genes.

5.5.3.4 Gaussian mutation

A mutation operator replaces the value of the chosen gene with a uniform random value selected between the user-specified upper and lower bounds for that gene. This mutation operator can only be used for integer and float genes.

5.5.4 Reinsertion

Once the offspring have been produced by selection, recombination and mutation of individuals from the old population, the fitness of the offspring may be determined. If less offspring are produced than the size of the original population then to maintain the size of the original population, the offspring have to be reinserted into the old population. Similarly, if not all offspring are to be used at each generation or if more offspring are generated than the size of the old population then a reinsertion scheme must be used to determine which individuals are to exist in the new population.

Different schemes of global reinsertion exist:

- Produce as many offspring as parents and replace all parents by the offspring (**pure reinsertion**).
- Produce less offspring than parents and replace parents uniformly at random (**uniform**

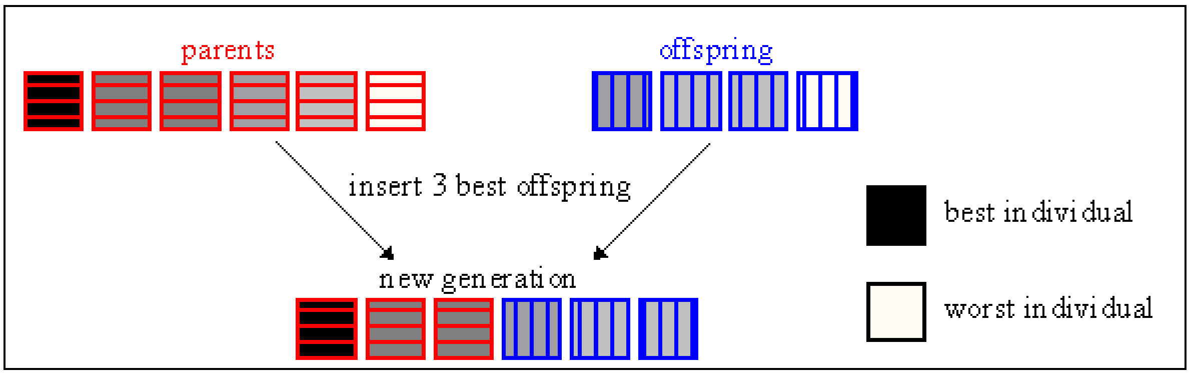
reinsertion).

- Produce less offspring than parents and replace the worst parents (**elitist reinsertion**).
- Produce more offspring than needed for reinsertion and reinsert only the best offspring (**fitness-based reinsertion**).

Pure Reinsertion is the simplest reinsertion scheme. Every individual lives one generation only. This scheme is used in the simple genetic algorithm. However, it is very likely, that very good individuals are replaced without producing better offspring and thus, good information is lost.

The elitist combined with fitness-based reinsertion prevents this losing of information and is the recommended method. At each generation, a given number of the least fit parents is replaced by the same number of the most fit offspring (see figure 6). The fitness-based reinsertion scheme implements a truncation selection between offspring before inserting them into the population (i.e. before they can participate in the reproduction process). On the other hand the best individuals can live many generations. However, every generation some new individuals are inserted. It is not checked whether the parents are replaced by better or worse offspring.

Figure 6 Scheme for elitist insertion



Because parents may be replaced by offspring with a lower fitness, the average fitness of the population can decrease. However, if the inserted offspring are extremely bad, they will be replaced with new offspring in the next generation.

5.5.5 Termination

Termination is the criterion by which the genetic algorithm decides whether to continue searching or stop the search. Each of the enabled termination criterion is checked after each generation to see if it is time to stop.

The following types of termination should be considered:

- **Generation Number:** A termination method that stops the evolution when the user-specified maximum number of evolutions has been run. This termination method is always active.
- **Evolution Time**: A termination method that stops the evolution when the elapsed evolution

time exceeds the user-specified max evolution time. By default, the evolution is not stopped until the evolution of the current generation has completed, but this behavior can be changed so that the evolution can be stopped within a generation.

- ✧ **Fitness Threshold:** A termination method that stops the evolution when the best fitness in the current population becomes less than the user-specified fitness threshold and the objective is set to minimize the fitness. This termination method also stops the evolution when the best fitness in the current population becomes greater than the user-specified fitness threshold when the objective is to maximize the fitness.
- ✧ **Fitness Convergence**: A termination method that stops the evolution when the fitness is deemed as converged. Two filters of different lengths are used to smooth the best fitness across the generations. When the smoothed best fitness from the long filter is less than a user-specified percentage away from the smoothed best fitness from the short filter, the fitness is deemed as converged and the evolution terminates.
- ✧ **Population** Convergence: A termination method that stops the evolution when the population is deemed as converged. The population is deemed as converged when the average fitness across the current population is less than a user-specified percentage away from the best fitness of the current population.
- ✧ **Gene Convergence**: A termination method that stops the evolution when a user-specified percentage of the genes that make up a chromosome are deemed as converged. A gene is deemed as converged when the average value of that gene across all of the chromosomes in the current population is less than a user-specified percentage away from the maximum gene value across the chromosomes.

Reference

1. Calinski R. B.and Harabasz J. (1974) A dendrite method for cluster analysis. *Communications in Statistics*. 3: 1–27.
2. Cowgill, M.C., Harvey, R. J. et al. (1999) A genetic algorithm approach to cluster analysis. *Comput. Math. App.* 37: 99–108.
3. Esien, M.B., Spellman, P.T., Brown, P.O., Botstein, D. (1998) Cluster analysis and display of genome-wide expression patterns. *Proc. Natl. Acad*, *USA*. 95: 14863–14868.
4. Goldberg, D.E. (1989) *Genetic Algorithms in Search. Optimization and Machine Learning*. Addison Wesley, New York.
5. Haupt, R.L., Haupt, S.E. (1998) *Practical Genetic Algorithms*. Wiley, New York.
6. Holland, J.H. (1962) Outline for a logical theory of adaptive systems. Journal of the

Association for Computing Machinery. 9 (3): 297–314.

7. Jain, A. K. and Dubes, R. C. (1988) *Algorithms for Clustering Data*. Prentice Hall, Englewood Cliffs, NJ.
8. Kaufman, L., Rousseeuw, P.J. (1990) *Finding Groups in Data*. John Wiley, New York.
9. Maulik, L., Bandyopadhyay, S. (2000) Genetic Algorithm-Based Clustering Technique. *Pattern Recognition*. 33: 1455–1465.
10. Miller, R.G. (1974) The Jackknife: a review. *Biometrika*, 61: 1–15.
11. Ooi, C.H., Tan, P. (2002) Genetic algorithms applied to multi-class prediction for the analysis of gene expression data. *Bioinformatics*. 19: 37–44.
12. Pan, H.Y., Zhu, J. and Han, D.F. (2003) Genetic algorithms applied to mutli-class clustering for gene expression data. *Geno., Prot. & Bioinfo.*. 1(4): 279–287.
13. Press W, Teukolsky S.A., Vetterling W.T., Flannery B.P. (2000) *The Art of Scientific Computing. Second Edition*. Cambridge University Press.
14. Rao, C. R. (1970) Estimation of heteroscedastic variances in linear models. *J. Am. Stat. Assoc*. 65:161–172.
15. Spath, H. (1989) *Cluster Analysis Algorithm*. Ellis Horwood, Chichester, UK.
16. Searle S.R., Casella G., McCulloch C.E. (1992) *Variance components*. John Wiley and Sons, NY.
17. Zhu, J. and Weir, B. (1994) Clustering populations by mixed linear models. *Journal of Biomathematics*, 9(3): 1–14.
18. Zhu, J. and Weir, B.S. (1996) Diallel analysis for sex-linked and maternal effects. *Theor. Appl. Genet.*, 92: 1–9.