

Tel Aviv University
The Sackler Faculty of Medicine
Graduate School

Field: Bioinformatics

**Subject: Tools for analysis and visualization of gene
expression data obtained using microarrays**

Submitted by Adi Maron-Katz

This work was carried out in partial fulfillment of the requirements for an M. Sc.
degree, in the Sackler Faculty of Medicine, Tel Aviv University.

Advisors: Prof. Ron Shamir
 Prof. Yosef Shiloh

July 2004

Acknowledgements:

I would like to thank all the people who assisted me in this work: I thank Dr. Roded Sharan, for giving me the opportunity of working on EXPANDER, Amos Tanay and Chaim Linhart for their guidance, help and support throughout the development process, and Naama Arbili and Israel Steinfeld for great assistance in user support and version updates. I am grateful to Ran Elkon from Prof. Yossi Shiloh's lab, for devoted counseling, to my dear family for its support and last but not least, to my advisors, Prof. Ron Shamir and Prof. Yossi Shiloh, for their devoted guidance throughout this work.

This study was supported in part by an infrastructure research grant from the Ministry of Science and Technology, Israel.

Contents

1	ABSTRACT	8
	INTRODUCTION AND SUMMARY	9
2.1	Gene Expression Microarray technology	9
2.1.1	cDNA Microarrays	9
2.1.2	High-density oligonucleotide arrays.....	10
2.2	Technology applications	11
2.2.1	Comparing two mRNA populations by identifying differentially expressed genes	11
2.2.2	Identifying co-expressed genes	11
2.2.3	Predicting gene functions	11
2.2.4	Promoter signal analysis	12
2.2.5	Tissue classification.....	12
2.2.6	Drug development.....	12
2.2.7	Toxicogenomics	13
2.3	Existing analysis tools	13
2.4	Existing GEM Databases	16
2.5	Summary of thesis results	17
3	RESEARCH OBJECTIVES	19
4	ANALYSIS METHODS	20
4.1	The analyzed data	20
4.2	Preprocessing	20
4.2.1	Normalization.....	20
4.2.2	Filtration	21

4.2.3	Standardization.....	22
4.3	Clustering	22
4.3.1	K-means.....	23
4.3.2	Self Organizing Maps.....	23
4.3.3	CLICK (Cluster Identification via Connectivity Kernels)	24
4.3.4	Hierarchical clustering.....	25
4.4	Biclustering	26
4.4.1	SAMBA	26
4.5	Analysis of clustering solutions	26
4.5.1	Homogeneity and separation scores	27
4.5.2	Functional analysis	27
	Promoter analysis.....	28
5	VISUALIZATION METHODS.....	30
5.1	Matrix displays	31
5.1.1	Expression matrix	31
5.1.2	Similarity matrix	33
5.2	Pattern displays	34
5.2.1	Mean cluster patterns.....	34
5.2.2	Cluster contour.....	35
5.2.3	Patterns of all genes in a cluster.....	35
5.3	Scatter plots	36
5.3.1	PCA analysis display	36
5.3.2	Data plots of two arrays.....	37
5.4	Histograms	37
5.4.1	Functional analysis display	37

5.4.2	Promoter analysis display	39
5.5	Dendrogram trees	40
5.5.1	Hierarchical clustering results display	40
5.6	Data tables	41
5.6.1	Biclustering results data table	41
6	SOFTWARE DEVELOPMENT AND ARCHITECTURE.....	43
6.1	Selecting the development language	43
6.2	Architectural considerations and overview	43
6.2.1	Architecture considerations	43
6.2.2	Overview	44
6.3	Data management	45
6.3.1	The FloatMatrix class	45
6.3.2	The BasicElement and ElementArray classes.....	45
6.3.3	The MainData class	46
6.3.4	The Biclust and BicSet classes	46
6.3.5	The Preferences class.....	47
6.4	Data analysis	47
6.4.1	The Algorithm class.....	47
6.4.2	The ClusteringAlgorithm class	48
6.4.3	Classes extending ClusteringAlgorithm	48
6.4.4	Classes extending Algorithm	48
6.4.5	Handling external modules via Algorithm classes	48
6.4.6	From data analysis to display: the visualization tools	49
6.5	The graphical Interface	50
6.5.1	The DisplayPanel class	50

6.5.2	Displaying matrices	50
6.5.3	Displaying charts	51
6.5.4	Displaying data tables.....	52
6.5.5	Displaying dendrogram trees	52
6.5.6	Creating the display frames.....	53
6.6	The Main Frame - How it all comes together	53
6.7	The utility Package	54
6.7.1	Handling float vectors – the VecCalc class.....	54
6.7.2	The Strings class	54
6.7.3	The Constants class.....	55
6.7.4	Connecting to the WEB – the URLHandler class.....	55
6.8	A detailed overview	55
7	EXPANDER AS AN INSTRUMENT IN THE HANDS OF THE RESEARCHER.....	57
7.1	Example 1: Analysis of oligonucleotide array data from the mouse lymph nodes	57
7.1.1	The data	57
7.1.2	Loading the data.....	58
7.1.3	Preprocessing the data	59
7.1.4	Viewing raw data	60
7.1.5	Clustering the data	60
7.1.6	Viewing clustered data	61
7.1.7	Performing functional analysis on clusters	64
7.1.8	Performing promoter analysis on clusters	66
7.2	Example 2: Analysis of yeast c-DNA microarray data concerning responses to environmental changes	68

7.2.1	The data	68
7.2.2	Loading the data.....	68
7.2.3	Biclustering the data	69
7.2.4	Performing functional analysis on biclusters	69
7.2.5	Viewing biclusters and significant functional classes	71
7.2.6	Performing Promoter analysis on biclusters.....	76
7.2.7	Discussion	79
7.3	Example 3: Analysis of published cDNA microarray data associated with cell cycle progression in human cells	81
7.3.1	The data	81
7.3.2	Loading the data.....	81
7.3.3	Preprocessing the data	81
7.3.4	Loading a clustering solution	82
7.3.5	Viewing clustered data	82
7.3.6	Functional analysis	84
7.3.7	Promoter analysis.....	87
7.3.8	Discussion	89
8	BIBLIOGRAPHY	90

1 Abstract

In the past few years the gene expression microarray (GEM) technology has become a central tool in the field of functional genomics. This field deals with exploring the functions of different gene products, the control mechanisms regulating their activity, their expression levels and their interactions. In the GEM technology, the expression levels of thousands of genes in a biological sample are determined in a single experiment.

This work describes the development of a bioinformatics software tool called EXPANDER (EXPression ANalyzer and DisplayER), that was designed to help researchers in analyzing GEM data, and allow viewing the raw data and analysis results via convenient graphical displays. The tool incorporates several conventional GEM analysis algorithms and custom ones that have been developed in the computational genomics group in Tel-Aviv University, and provides them with an easy-to-operate user interface. Among the tool's capabilities are clustering, biclustering, functional enrichment and promoter analysis, in addition to a variety of visualizations. EXPANDER was programmed using the Java programming language and it can be run on several platforms, including Windows and Unix. It was written in an object oriented approach, suitable for such a large scale applications that requires many different modules that interact with one another. EXPANDER based analyses are demonstrated using three different biological datasets, and novel biological conclusions are drawn.

The EXPANDER tool is freely available for academic research, and is broadly used both for in-house research projects in biology and medicine at Tel Aviv University, and in other institutions. Over four hundred laboratories have downloaded the software over the last year. It is under ongoing development in order to keep it a state-of-the-art research tool with unique capabilities.

Key terms: Functional genomics, gene expression microarrays, software, cDNA microarrays, high-density oligonucleotide arrays, clustering, biclustering, functional analysis, promoter analysis.

2 Introduction and summary

2.1 Gene Expression Microarray technology

The Gene Expression Microarray (GEM) technology plays a central role in the field of functional genomics. This field is based on the recent progress achieved in genome sequencing (*Hieter et al. 1997*) and other high throughput techniques. It deals with exploring the function of different gene products, the control mechanisms regulating their activity, their expression levels and their interactions.

In the GEM technology, the expression levels of thousands of genes in a biological sample are determined in a single experiment. Genes printed on a slide (usually glass) are hybridized against labeled probes, prepared from the cell lines that are being tested.

There are currently two main methods implementing this technology: cDNA microarrays and high-density oligonucleotide arrays ("DNA chips"). They differ in the way genes are represented on the slide, the way the slide is prepared, and some of the experimental stages, but are used usually for the same needs. The methods used to analyze the data are similar, but while the first method produces relative expression level values, the second produces absolute values.

2.1.1 cDNA Microarrays

This method was developed in the department of Biochemistry of Stanford university in 1996, and has since been adopted by many laboratories. In this method a grid of cDNA dots is printed over a glass slide. Each dot contains cDNA molecules (0.2-2kb long) from a clone of a single gene. A grid containing 10000 such dots can be printed on a slide of size $2.5 \times 2.5 \text{ cm}^2$ (*Shalon et al. 1996*). Currently, newer, more advanced printing methods are being developed, that will allow the representation of a whole genome on a single array (*Hughes et al. 2001*).

The experiment involves the following steps: (1) Extracting mRNA molecules from two cell populations (the *test* population and the *reference* population). (2) Reverse

transcription of the mRNA molecules to create labeled cDNA molecules, by using fluorescent nucleotides (with different colors for the test population and for the reference population). (3) Co-hybridization of the cDNA from the test and the reference populations to the same array. (4) Scanning the array using a laser scanner. The last stage is performed separately for each of the two color frequencies, to create two image files.

The expression levels evaluated in this method are relative (between the two cell populations) since the number of cDNA molecules that are printed in each spot cannot be accurately estimated. Therefore, a common reference population must be used when attempting to test expression level changes over several conditions.

2.1.2 High-density oligonucleotide arrays

This method was developed and is applied primarily by the company Affymetrix. In the oligonucleotide array each gene is represented by 10-20 different oligonucleotides of length 25bp. The oligonucleotides representing a gene are selected in a way that minimizes their homology to other known sequences, in order to increase their specificity. The representation of each gene by 10-20 probes increases probe-gene specificity significantly. The oligonucleotides are synthesized over a glass slide, using a photolithographic method. This method allows the representation of hundreds of thousands of genes on a single array at the size of about 1.5 square centimeters (*McGall et al. 2002*).

The experiment is performed for a single population of cells (unlike in the cDNA microarray method), and involves the following steps: (1) Extracting mRNA molecules from the tested cell population. (2) Reverse transcription of the mRNA molecules, using a primer that contains a promoter for the T7 RNA Polymerase, to create labeled cDNA molecules. (3) Transcription of the synthesized cDNA template, using T7 RNA Polymerase and tagged nucleotides, to create tagged RNA molecules. This process causes a linear induction of the initial RNA concentration (up to 100 fold), which allows determining also very low expression levels. (4) Breaking the tagged RNA molecules into sections of average length 50bp, and hybridizing them against the array. (5) Scanning the array using a laser to create an image file.

It has been shown that the expression levels determined using this method are proportional to the amount of mRNA in the cell (*Wodicka et al. 1997*).

2.2 Technology applications

In this section, we will describe the main current applications of the GEM technology.

2.2.1 Comparing two mRNA populations by identifying differentially expressed genes

The purpose of these experiments is to identify genes that exhibit a distinct difference in expression levels between two tested populations, for example, cells before and after a certain treatment. This way, the involvement of novel genes in different biological processes can be revealed. This approach has been utilized in many studies e.g., to identify novel candidate genes involved in systemic metastases in lung cancer (*Liu et al. 2004*), novel candidate genes involved in neurodegenerative disease (*Glanzer et al. 2004*), genes regulated by p53 (*Zhao et al. 2000*) and E2F (*Ishida et al. 2001*), and genes that comprise the peroxide stimulon in the cyanobacterium *Synechocystis* sp (*Li et al. 2004*).

2.2.2 Identifying co-expressed genes

The goal here is to discover genes sharing a similar expression pattern over a set of tested conditions. Such similarity may indicate their involvement in a common function (e.g., the same metabolic pathway), or in common regulatory mechanisms. Identifying such gene groups can be achieved using clustering algorithms, applied to GEM data containing several different conditions (*Spellman et al. 1998, Sharan et al. 2000*). This approach has been utilized in many studies in order to extract new biological information from the GEM data (*Shannon et al. 2003*).

2.2.3 Predicting gene functions

The goal here is to find gene subgroups that share a common function, by detection of functions that are significantly overrepresented in one cluster of co-expressed genes. This criterion is called "Functional Enrichment". If a cluster is significantly enriched

for genes having a certain function, other non-annotated genes in that cluster are more likely to have the same function. This approach has been successfully used on yeast to predict the function of over 800 uncharacterized genes (*Tanay et al. 2004*).

2.2.4 Promoter signal analysis

The goal here is to reveal cis-regulatory mechanisms that are activated as a result of exposing the cell to certain experimental conditions. To achieve this goal, promoters of genes in the same co-expression cluster are scanned to find gene subgroups that share common transcription factor binding sites in a statistically significant manner. It is plausible that the transcription factors that bind such sites directly regulate the genes that are responsible for the observed changes in expression levels. This approach has been used successfully for several organisms and tissues, including the yeast (*Jelinsky et al. 2000, Pilpel et al. 2001*) and human HeLa cancer cells (*Elkon et al. 2002*).

2.2.5 Tissue classification

This can be done by identifying gene expression profiles that are typical for certain tissues (e.g., tissues from a certain type of cancer). Such profiles constitute a molecular 'fingerprint' that can be used to identify the tissue. It has been shown that these profiles can help distinguish between a cancerous tissue and a normal tissue (*Alon et al. 1999*) and even between tissues of different types of cancer (*van de Vijver et al. 2002, Dyrskjot et al. 2003, Golub et al. 1999, Eisen et al. 1998*). This application can have a major contribution in the field of medical diagnosis and treatment.

2.2.6 Drug development

The GEM technology assists in several stages in the process of drug development (*Lord et al. 2004, Clarke et al. 2001, Marton et al. 1998, Braxton et al. 1998*): (a) Choosing the target protein by tracing the genes that exhibit significant changes in expression levels between the normal and the pathological states. (b) Testing candidate drugs by comparing the cellular expression profile that is achieved after treatment to the normal (desired) profile (*Waddell et al. 2004*). (c) Identifying potential side effects by examination of the differences between the desired cellular

expression profile and the profile achieved after treatment. (d) Predicting the toxicity of a drug (as explained in the next section).

2.2.7 Toxicogenomics

In this field, different toxic substances are characterized by the cellular expression profile that they induce. In recent years, several databases that characterize toxic substances according to their induced expression profile have been established (*Lord 2004, Irwin et al. 2004, Nuwaysir et al. 1999*). It has been found that substances with similar toxic activity induce a similar expression profile. In the future, the toxic potential of a substance will be evaluated by comparing its induced expression profile to existing database records (*Nuwaysir et al. 1999, Fredrickson et al. 2001*).

2.3 Existing analysis tools

Computational analysis tools are crucial for the efficient exploitation of the large amounts of data produced by GEM experiments. Dealing with such large datasets requires the development and use of data analysis algorithms that will extract biologically meaningful information out of the raw data (*Dresen et al. 2003, Eisen et al. 1998, Quackenbush et al. 2001*).

Many of the computational tools that are currently used for GEM analysis focus on one or several stages of the analysis. Hence, analysis requires the porting of the data between different software tools. This often requires reformatting the data according to the different software tools, and makes the employment of more than a small number of tools simultaneously very cumbersome.

Among the commonly used analysis tools are:

dChip (<http://biosun1.harvard.edu/complab/dchip>) – A windows application, that operates on high-density oligonucleotide arrays. The tool performs several normalizations, including tracing and omitting data in contaminated areas, or data that were cross hybridized (i.e., hybridization of mRNA of one gene to probes of another gene that has a highly similar sequence) (*Li et al. 2001*), filtering out non informative genes, and identifying genes that are differentially expressed between two conditions. The program also performs hierarchical clustering and principal component analysis

(PCA) on the processed data, and produces graphical displays. The software utilizes the R application (*Ripley 2001*) and operates on Windows 2000 operating system.

GeneX-CyberT (<http://visitor.ics.uci.edu/genex/cybert/>) - a statistical program with a web interface that can be used on both cDNA microarray data and oligonucleotide arrays data for the identification of statistically significant differentially expressed genes. The analysis is based on Bayesian approach and generates text output files and a file of statistical charts.

Cluster & TreeView - (<http://rana.lbl.gov/EisenSoftware.htm>) – These are an integrated pair of programs for analyzing and visualizing the results of both cDNA microarray and high density oligonucleotides experiments (*Eisen et al. 1998*). The **Cluster** program implements the following clustering and analysis methods: hierarchical clustering, self-organizing maps (SOM), k-means, PCA and hierarchical clustering. The program operates on Windows only. The **TreeView** graphical program enables viewing the results of clustering and other analyses from Cluster. It supports tree-based and image based browsing of hierarchical trees. It produces multiple output formats for the generation of images for publications. It operates only on Windows. Another visualization program, Maple Tree, which is cross-platform (i.e., runs on all operating systems) is now available from the same group. It allows to graphically browse the results of clustering analyses from the Cluster software, and many other clustering and analysis programs.

JExpress (<http://www.ii.uib.no/bjarted/jexpress/>, *Dysvik et al. 2001*) – This program operates on cDNA microarray data. Performs high-level normalization, filtering and high-level analysis. The analysis methods implemented in JExpress are: Hierarchical clustering, SOM, PCA, K-means and profile search. The program contains several visualization tools.

Genesis (<http://genome.tugraz.at>, *Sturn et al. 2002*) – A cross platform program that performs data normalization based on a variety of techniques, for sets of genes or experiments (mean centering, median centering, division by SD/RMS and log transformation) and data filtering (according to missing values and standard deviation). It implements several clustering algorithms (hierarchical clustering, k-means, SOM) and provides also other analysis methods such as Principal Component Analysis (PCA), and support vector machines (SVM, a classification tool). The

program utilizes several different similarity measurements (ranging from Pearson correlation to more sophisticated approaches, like mutual information). It supplies several visualizations to view the above analysis results and allows the mapping of gene expression data onto chromosomal sequences.

Spotfire DecisionSite (<http://www.spotfire.se/>) – An application for microarray data analysis and visualization. It implements the following analysis methods: hierarchical and K-means clustering, expression profile searches and PCA. Other analysis methods require the R application (*Ripley 2001*) and include normalization schemes, variance analysis using ANOVA and rule induction analysis with decision trees. The application incorporates various visualizations such as box plots, pattern displays, matrix displays, pie charts and dendrogram trees. Annotation information from various sources can be loaded and integrated into the visualizations. DecisionSite operates on Windows only.

GeneXPress (<http://genexpress.stanford.edu/>, *Segal et al. 2004*) – Given a clustering solution (or a file generated by the TreeView software), this application performs functional analysis and promoter analysis and provides various suitable displays.

GeneCluster (<http://www.broad.mit.edu/cancer/software/software.html>, *Reich et al. 2004*) – A cross platform program that facilitates filtering and preprocessing data in a variety of ways, clustering expression profiles using the SOM algorithm, and viewing the results. It also allows supervised classification, gene selection and permutation test methods (Permutation test methods are used to assess the significance of the score for each gene, i.e. the estimated signal to noise ratio). It includes algorithms for constructing and testing supervised models that will be able to predict different variables (e.g. tumor type, treatment outcome etc.) based on the expression values using weighted voting (WV) and k-nearest neighbors (KNN) algorithms.

TM4 (<http://www.tigr.org/software/tm4/>, *Dudoit et al. 2003*) – A package that consists of four major applications, two of which (Microarray Data Analysis System, and Multi-experiment Viewer) perform high-level preprocessing, analysis and visualization of microarray data. These software tools were developed for spotted two-color arrays, but can be easily adapted to work with single-color formats such as high density oligonucleotide arrays. Both programs are cross platform. Microarray Data Analysis System (MIDAS) performs normalization using locally

weighted linear regression (lowess) and total intensity normalization. It also performs filtering using several methods. MIDAS provides scatter plots that illustrate the effects of each algorithm on the data. It reads “.tav” files generated by TIGR Spotfinder program or retrieved from the database via MADAM (another TM4 application). Multi-experiment Viewer (MeV) operates on normalized and filtered expression files. It incorporates several clustering algorithms such as: hierarchical clustering, K-means, SOM, SOT (Self Organizing Trees), Gene Shaving and QT_clust, along with other analysis algorithms such as PCA, Significance Analysis of Microarrays (SAM) etc. Results can be graphically displayed. MeV can handle several input file formats.

DMT (<http://www.affymetrix.com/products/software/specific/dmt.affx>) - The Data Mining Tool (DMT) software, developed by Affymetrix, provides several tools for filtering and sorting microarray data generated using the Affymetrix GeneChips. Key features include: pairwise statistical analysis for replicate samples, clustering (SOM and a modified Pearson’s Correlation Coefficient method) and an option to integrate annotation information into the data. DMT operates on Windows only.

2.4 Existing GEM Databases

In this section I present some of the more commonly used GEM databases.

Gene Expression Omnibus (GEO) (<http://www.ncbi.nlm.nih.gov/geo/>) – one of the most commonly used public repositories for a wide range of high-throughput experimental data. These data include single and dual channel microarray experiments measuring mRNA, genomic DNA and protein abundance, as well as non-array techniques such as serial analysis of gene expression (SAGE), and mass spectrometry proteomic data. It allows data browsing, query and retrieval. It currently contains over 20,000 sample records (arrays).

Array Express (<http://www.ebi.ac.uk/arrayexpress/>) – The main European public repository for microarray data, which is aimed at storing well annotated data in accordance with the Microarray Gene Expression Data (MGED) Society recommendations. The MGED society has defined the MIAME (minimum information about a microarray experiment) requirements in order to enable the

interpretation of the results of the experiment unambiguously and potentially to reproduce the experiment (*Brazma et al. 2001*). The data deposited in Array Express is expected to fill these requirements. Array Express currently contains record from over 6000 profiles.

GeneX (<http://www.research.ibm.com/journal/sj/402/mangalam.html>) – an open source gene expression database and integrated toolset that allows researchers to store and evaluate their gene expression data independently of the technology used to obtain the data.

Gene Expression Database (GXD)

(<http://www.informatics.jax.org/mgihome/GXD/aboutGXD.shtml>) - a community resource for gene expression information from the laboratory mouse. GXD stores and integrates different types of expression data and makes these data freely available in formats appropriate for comprehensive analysis. There is particular emphasis on endogenous gene expression during mouse development.

Stanford Microarray Database (SMD) (<http://genome-www5.stanford.edu/>) - one of the first academic databases to be used on an institutional scale. It contains the largest amount of data of any academic database, due to its close association with one of the first groups to develop large-scale arrays. It stores raw and normalized data from microarray experiments, as well as their corresponding image files. It also provides interfaces for data retrieval, analysis and visualization. Data is released to the public at the researcher's discretion or upon publication.

2.5 Summary of thesis results

This work describes the development of a bioinformatics software tool called EXPANDER (EXpression ANalyzer and DisplayER), that was designed to help researchers in analyzing GEM data, and allow viewing the raw data and analysis results via convenient graphical displays. The tool incorporates several conventional GEM analysis algorithms and custom ones that have been developed in the computational genomics group in Tel-Aviv University, and provides them with an easy-to-operate user interface. Among the tools capabilities are clustering, biclustering, functional enrichment and promoter analysis, in addition to a variety of visualizations. EXPANDER was programmed using the Java programming language

and it can be run on several platforms, including Windows and Unix. It was written in an object oriented approach, suitable for such a large scale applications that requires many different modules that interact with one another.

EXPANDER based analyses are demonstrated using three different biological datasets, and novel biological conclusions are drawn.

The EXPANDER tool is freely available for academic research (it can be downloaded from www.cs.tau.ac.il/~rshamir/EXPANDER). Over four hundred laboratories have downloaded the software over the last year. It is broadly used both for in-house research projects in biology and medicine at Tel Aviv University and in other institutions. Among the in-house research projects that utilize EXPANDER are a microarray project that analyzes DNA damage responses in human cells and mouse tissues, conducted at Yossi Shiloh's laboratory in the Sackler medical school, a microarray project that studies inflammation processes in brain of mouse models for Alzheimer disease, conducted at Danny Michelson's laboratory in the George S. Wise faculty of Life Science and a microarray project that studies mis-regulated signaling pathways neuroblastomas, conducted at Yoel Klug's laboratory in the George S. Wise faculty of Life Science. EXPANDER is under ongoing development in order to keep it a state-of-the-art research tool with unique capabilities.

A preliminary report on the EXPANDER project has been published in (*Sharan et al. 2003*).

3 Research Objectives

My objective in this research was to develop a computerized tool that will achieve the following goals:

1) Incorporation of several analysis stages in one program:

Bringing together different tools from all stages of GEM data analysis under a single platform. These include preprocessing tools, advanced downstream analysis tools and various visualization tools. The purpose of this integration is to help the user, partly by eliminating the work that is involved in formatting data to be transferred from one application to another.

2) Incorporation of novel analysis algorithms which are developed in the computational genomics laboratory in Tel-Aviv University:

Several GEM analysis tools have been developed in the computational genomics laboratory in Tel-Aviv University and are available for academic use, e.g., the SAMBA (Statistical Algorithmic Method for Bicluster Analysis) algorithm for biclustering GEM data (*Tanay et al. 2002*) and the PRIMA (Promoter Integration for Microarray Analysis) algorithm for promoter analysis (*Elkon et al. 2003*). The incorporation of these tools into the program will provide them with an easy-to-operate user interface, and will enable viewing and manipulating their results via convenient graphical displays.

3) Generation of original graphical visualizations:

Such visualizations will hopefully provide an additional point of view on the biological data, in order to promote the discovery of new insights by analyzing GEM experiments.

4) Cross platform application:

Our program was designed to run on the two most commonly used operating systems (Windows and Unix).

Such a program may encourage the usage of the tools and improve the ability of the user to analyze and extract new biological knowledge from GEM data.

4 Analysis Methods

In this chapter we describe the algorithms and procedures that are included in EXPANDER, and provide examples of their output. The technical implementation details will be described in chapter 6.

4.1 The analyzed data

The analysis methods described below are performed on data matrices, in which each row corresponds to a gene and each column corresponds to an experimental condition. Thus, a row vector is the expression pattern of a gene, and a column vector is the expression profile under a particular condition.

The values in the matrix represent the relative (in cDNA microarray data) or absolute (in high-density oligonucleotide data) measured expression levels. For example, the value in the i^{th} row and the j^{th} column represents the expression level of the i^{th} gene in the dataset, as measured in the j^{th} experimental condition.

4.2 Preprocessing

The purpose of preprocessing is to remove insignificant and useless expression patterns, and bring all remaining data into a unified form on which downstream analysis (such as gene clustering, biclustering etc.) can be performed. To achieve this goal the data from different experiments should be adjusted to the same scale, and data size should usually be reduced by filtering out non-informative patterns, so that downstream analysis will run in a reasonable time and will provide meaningful results.

4.2.1 Normalization

Normalization is the process of reducing sources of variation of non-biological origin between arrays (*Bolstad et al. 2003*). In EXPANDER, normalization schemes are implemented only for oligonucleotide arrays (it is assumed that for cDNA microarrays, entries are given in log red/green values that are already normalized).

EXPANDER implements two non-linear normalization schemes presented and tested by *Bolstad et al. (2003)*:

- a) The first method, "Quantile normalization" (*Bolstad et al. 2003*), is a complete data method (i.e., it is applied to all arrays together). This method is aimed at creating an identical distribution to each array in the data set, by ranking the entries in each condition (breaking ties arbitrarily), and replacing each entry by the average of the entries of its rank in all conditions.
- b) The second method, "Non linear baseline normalization", is a baseline method (i.e., all arrays are normalized according to one selected baseline array). In this method a non-linear regression is used in order to map each array expression values to the baseline array (*Schadt et al. 2002*). An XY-scatter plot is created using the values in the array that is being normalized as the X values and the corresponding values from the baseline array as the Y values. A nonlinear regression is performed on this scatter plot using a Lowess (locally weighted smoothing scatter plots)-like function, in which each x value is mapped to the average y value of its n nearest neighbors (i.e., the average y of the center of the window to which it belongs). This normalization should be performed using a subset of the genes that is considered relatively non-variant (under the experimental conditions). For this purpose, the user can choose between using all genes (in case data set is expected to contain mostly non-differential genes) and using a rank invariant subset of genes. Calculating the rank-invariant set is based on the method presented by *C. Li and W.H.Wong (2001)*.

4.2.2 Filtration

Genes that do not exhibit significant changes in their expression levels under the tested conditions do not add relevant information to the analysis. Thus, it is preferable to filter out such genes before performing any downstream analysis such as clustering or bi-clustering.

EXPANDER implements two filtration schemes:

- a) Fold change filter – only genes whose expression level varies by at least k fold across the tested conditions are selected. k, as well as the reference array are determined by the user (the reference array can be set to the array with

minimal expression level). The user can also set an additional requirement of a minimal (>1) number of conditions in which the required fold change in must occur.

- b) Variation filter - the k genes that exhibit the highest variation in expression levels throughout all conditions are selected (k is a parameter that is determined by the user). Variance is used to measure variation for cDNA microarray data, and coefficient of variation is used to measure variation for oligonucleotide data.

4.2.3 Standardization

When the range of expression values of different genes is very different, but their general expression patterns are similar (i.e., they have high correlation coefficient), we would like to see this similarity when looking on a pattern display. Since the absolute values of expression are different, a manipulation is required in order to view the patterns on the same scale. This manipulation is called standardization.

EXPANDER implements two standardization schemes:

- a) Mean 0 and variance 1 – the expression pattern of each probe is set to have a mean equal to 0 and a variance equal to 1. This method is suitable in most cases when working on genes.
- b) Fixed norm – for each probe, expression levels are divided by the norm of that expression vector (the root of sum of squares of that vector's entries). This method is suitable in cases where we expect to find different means between patterns, or different variance values. For example, when working on time series conditions, we may expect larger variance in later phases of a response.

4.3 Clustering

Clustering is the process of partitioning elements (in our case, usually expression patterns of genes) into subsets, which are called clusters, so that two criteria are satisfied: *homogeneity* – high similarity between elements from the same cluster, and *separation* - low similarity between elements from different clusters (*Sharan et al.*

2000). There is very rich literature on cluster analysis (*Hartigan 1975, Everitt 1993, Mirkin 1996, Hansen & Jaumard 1997*).

4.3.1 K-means

K-means is a classical clustering algorithm (*Tavazoie et al. 1999*), which assumes that the number of clusters (k) is known. It aims to minimize the distances between elements and the centroids of their assigned clusters. The algorithm maintains a partition of the elements into k clusters. Each iteration of k-means modifies the current partition by checking all possible modifications of the solution, in which one element is moved to another cluster, and making the change that minimizes the following error function:

$$\sum_{m=1}^k \sum_{j \in C_m} \|x_{ji} - \bar{x}_m\|^2$$

Where: k is the number of clusters, C_m is the set of indices of elements in cluster m, n is the pattern length (the number of conditions), and \bar{x}_m is the mean pattern of cluster m, i.e.

$$\bar{x}_m = \frac{1}{|C_m|} \sum_{j \in C_m} x_{ji}.$$

Hence, each iteration reduces the sum of distances between elements and the centers of their clusters. This procedure is repeated until no further improvement is achieved. The above error function uses Euclidian distance as the distance measure. Other distance measurements can be used in the same way.

4.3.2 Self Organizing Maps

The Self Organizing Maps (SOM) (*Tamayo et al. 1999*) algorithm assumes that the number of clusters is known. Those clusters are organized as a set of nodes in a two dimensional $k \times l$ grid, where $k \times l$ is the number of clusters. Each of the nodes is associated with a reference vector of the same dimension as the expression patterns. The algorithm iteratively selects a random data point (p), identifies its nearest reference vector n_p , and updates all reference nodes according to a learning function. In that function the extent of change in vector j is proportional to the proximity of its node n_j to node n_p in the grid, and also decreases with iteration number.

The learning function used in EXPANDER is the 'neighborhood function' (Tamayo *et al.* 1999):

$$f(n_j, n_p, i) = \begin{cases} a(i) & \text{if } d(n_p, n_j) \leq r \\ 0 & \text{Otherwise} \end{cases}$$

$$a(i) = \frac{0.02 * I}{I - 100 * i}$$

$$r = \text{const} * (1 - \frac{i}{I})$$

Here I is the total number of iterations, i is the current iteration number and the constant value is set to 3. $a(i)$ is called the 'learning rate' and decreases with the iteration number. $d(x,y)$ is distance between the grid points corresponding to x and y .

After calculating the learning function, the algorithm updates reference vectors using the following function:

$$n_{j(i,k)} = n_{j(i-1,k)} + f(n_j, n_p, i) * (x_{pk} - n_{j(i-1,k)})$$

Where $n_{j(i,k)}$ is the value of reference vector j in position k after i iterations, and x_{pk} is the value of the randomly chosen data-point (vector) p in position k .

4.3.3 CLICK

The CLICK algorithm (CLuster Identification via Connectivity Kernels) was developed in the Computational Genomics group of Tel-Aviv University (Sharan *et al.* 2000). It uses a graph theoretic approach to clustering. The input data are represented as a weighted graph, in which each gene is represented by a vertex, and the similarity between the expression patterns of each two genes is used to calculate the weight of the edge connecting their vertices.

The algorithm recursively partitions the current set of elements into two subsets by computing a minimum weight cut. If the sub graph induced by the current set of elements has a positive minimum cut value, then it is declared a kernel. Otherwise,

the set is split into two subsets separated by the minimum cut. The set of kernels and the set of singletons (elements not assigned to kernels) serve as a basis for the eventual clusters that are obtained by merging singletons and kernels heuristically.

CLICK uses a probabilistic model in order to determine the weights on graph edges and the stopping criterion. The key probabilistic assumption of the model is that pairwise similarity values between elements, $S(x,y)$, are normally distributed, i.e. $S(x,y) \sim N(\mu_T, \sigma_T^2)$ if (x,y) are 'mates', and $S(x,y) \sim N(\mu_F, \sigma_F^2)$ if (x,y) are 'non mates', where $\mu_T > \mu_F$. This assumption often holds on real data. These parameters as well as the probability that two elements are 'mates' (p_{mates}), are estimated using the EM algorithm (see e.g., *Mirkin et al. 1996*).

EXPANDER operates CLICK via an external module written in C++ by Naama Arbily and Dr. Roded Sharan from the Computational Genomics group of Tel-Aviv University.

4.3.4 Hierarchical clustering

Hierarchical clustering does not partition the genes into subsets. Instead it creates a hierarchy of the elements that can be represented by a dendrogram. This can be done using the 'agglomerative' method (*Eisen et al. 1998*), which starts with an initial partition into single element clusters and successively merges clusters until all elements belong to the same 'cluster'.

The algorithm iteratively merges clusters whose similarity value is the highest. After merging two clusters the dissimilarity (distance) matrix changes, and the new distances (between the merged clusters and all the other clusters) are calculated in one of three schemes:

a) Single-linkage: $d_{k,i \cup j} = \min\{d_{ki}, d_{kj}\}$

b) Complete-linkage: $d_{k,i \cup j} = \max\{d_{ki}, d_{kj}\}$

c) Average-linkage: $\frac{n_i * d_{ki} + n_j * d_{kj}}{n_i + n_j}$ where n_i is the number of elements in cluster

i.

In EXPANDER, hierarchical clustering is performed using the above method, and distance calculation scheme can be selected by the user.

4.4 Biclustering

In gene expression data, a bicluster is a subset of the genes exhibiting consistent patterns over a subset of the conditions. Biclustering overcomes some of the limitations of clustering: first, in clustering one assumes that related genes behave similarly across all measured conditions. This assumption does not hold for large datasets containing hundreds of heterogeneous conditions from many experiments. Second, a clustering solution is a partition of the genes into disjoint sets, implying an association of each gene with a single biological function or process, which may be an oversimplification of the biological system (*Tanay et al. 2002*).

4.4.1 SAMBA

The SAMBA algorithm for biclustering (Statistical Algorithmic Method for Bicluster Analysis) was developed in the computational genomics group of Tel-Aviv University (*Tanay et al. 2002*). It detects significant biclusters in a large expression dataset, using a graph theoretic approach coupled with statistical modeling of the data. The data is represented as a bipartite graph $G=(U,V,E)$, where: U is the set of conditions, V is the set of genes, and there exists an edge $e=(u,v)$ if and only if u responds to v (expression level of gene u changes significantly in condition v).

The SAMBA algorithm detects significant biclusters by using graph algorithms to find sub-graphs of the described bipartite graph that are relatively dense. For more details see (*Tanay et. al. 2002*).

EXPANDER operates SAMBA via an external module written in C++ by Amos Tanay from the Computational Genomics group of Tel-Aviv University.

4.5 Analysis of clustering solutions

After clustering/bi-clustering gene expression data, we wish to explore the biological quality and meaning of the results. Several methods are implemented in EXPANDER to assess the quality of a clustering solution and to explore its biological meaning.

4.5.1 Homogeneity and separation scores

Homogeneity and separation measurements can be used in order to assess the quality of a clustering solution. In EXPANDER the extent of homogeneity within each cluster is calculated by averaging the similarity of all pairs of genes that belong to that cluster. The extent of separation is calculated by averaging the similarity of all pairs of elements from different clusters.

4.5.2 Functional analysis

The functional analysis calculation is performed in order to detect clusters that are significantly enriched for genes from a certain functional class. Enrichment is evaluated by comparing the frequency of genes of a certain function in the cluster to the frequency of that function in the set of all genes, which is called the background set. To achieve this EXPANDER utilizes functional annotations (supplied in external files for mouse, human and yeast), which use the standard vocabulary introduced by the Gene Ontology (GO) consortium (*Ashburner et al. 2000*). To identify enriched functional categories a hyper geometric calculation is performed and a p-value is calculated for each pair of a cluster C and functional class f:

$$P\text{-value}(C, f) = \frac{\binom{|C|}{t} \binom{n - |C|}{K_b - t}}{\binom{n}{K_b}}$$

Where n is the size of the background set, K_c is the number of genes in cluster C that belong to functional class f, and K_b is the number of genes in background set that belong to functional class f. If a p-value is below a certain threshold, then the cluster appears to be enriched with that functional category.

EXPANDER operates functional analysis via an external module written in Perl by Amos Tanay from the Computational Genomics group of Tel-Aviv University. In this implementation the user can control the level of the functional class in the GO tree by setting a parameter of 'maximal class size' so that classes larger than that size are not taken into account (they are considered too general). If two groups of genes

corresponding to two functional classes in the same cluster are very similar, both functions are treated as one. This is performed in order to reduce the level of degeneracy in the results.

The background set used for calculation is determined by the user. It can be the whole dataset, the filtered data set or a set provided by the user.

Currently, no correction for multiple tests is applied when performing functional analysis via EXPANDER. The used functional attributes are highly redundant and strongly inter-dependent, and the subject of multiple tests correction when tests are not independent is still being studied. Thus, correction will have to be added in the future (A conservative Bonferroni correction would multiply the p-values by the total number of tests, i.e., the number of clusters times the number of functional classes).

4.5.3 Promoter analysis

The goal of promoter analysis is to identify the transcription factors that bring about the observed differences in gene expression in the data. To achieve this, EXPANDER employs a promoter analysis software called PRIMA (PRomoter Integration in Microarray Analysis) that was developed at the computational genomics group in Tel-Aviv University (*Elkon et al. 2003*).

Based on the assumption that genes exhibiting similar expression patterns across multiple conditions will share cis-regulatory elements in their promoters, PRIMA seeks out these common sequence elements. Given a target set of promoters (e.g., the promoters of genes in an identified cluster), a background set of promoters and a collection of known binding site profiles (see below), PRIMA performs statistical tests (using a hyper-geometric calculation) in order to identify transcription factors (TFs) whose binding site profiles are significantly more prevalent in the target set than in the background set. For each cluster and each TF binding site profile a p-value is calculated.

The background set used for calculation is determined by the user. It can be all genes, the genes left after filtering or a set of genes provided by the user. If the p-value is sufficiently low, then the cluster appears to be enriched with that binding site. At the user's request, the Bonferroni multiple tests correction can be applied on the results.

In order to perform this analysis efficiently, TF motif fingerprint files for each species (currently human and mouse) are supplied with EXPANDER. A set of 19,244 human promoter sequences, spanning from 1000 bp upstream the transcription start site (TSS) to 200 bp downstream the TSS, was scanned using PRIMA in order to locate putative binding sites (hits). The binding sites are modeled as a position-specific weight matrix, or PWM. The scan was performed for each TF motif in the Transfac database (*Matys et al. 2003*) version 7.4 (April 2004). The number of hits of a PWM in each promoter is called the PWM's fingerprint. The human promoter sequences were downloaded from Ensembl (<http://www.ensembl.org>), release 19.34b. Another set of fingerprints was prepared on mouse promoters (19,923 promoters, Ensembl release 19.30).

EXPANDER operates PRIMA via an external module written in C++ by Chaim Linhart from the Computational Genomics group of Tel-Aviv University.

5 Visualization methods

In this chapter we describe the main visualization methods used in EXPANDER.

Two datasets are used in the following examples: the first, referred to as dataset A, is yeast cell cycle data that is constructed of 698 genes over 72 conditions (*Spellman et al. 1998*). The second, referred to as dataset B, is expression profiles of mouse lymph nodes of wild-type and ATM^{-/-} mice at different time points after irradiation. Genes in dataset B were filtered using the fold change method (see section 7.1) and the filtered dataset used here is constructed of 1205 genes over 6 conditions (see section 7.1). Additional examples are provided in chapter 7.

5.1 Matrix displays

5.1.1 Expression matrix

This tool is very similar to "heat map" matrix representation introduced by *Eisen et al. (1998)*. Gene-expression data are rendered on the screen either in color or gray levels (colors can be configured via the 'Settings' dialog, accessible from the 'Options' menu of the main menu bar). In the color display blue (green) indicates under-expression, and yellow (red) indicates over-expression. In the grayscale display a darker rectangle indicates a higher expression value.

A color scale appears next to the matrix (upper right side) and is also available as a mobile frame through the 'Options' menu (or through the right click pop-up menu).

When data are clustered, this visualization is available also with rows ordered according to clusters. Columns appear in their original order in the matrix.

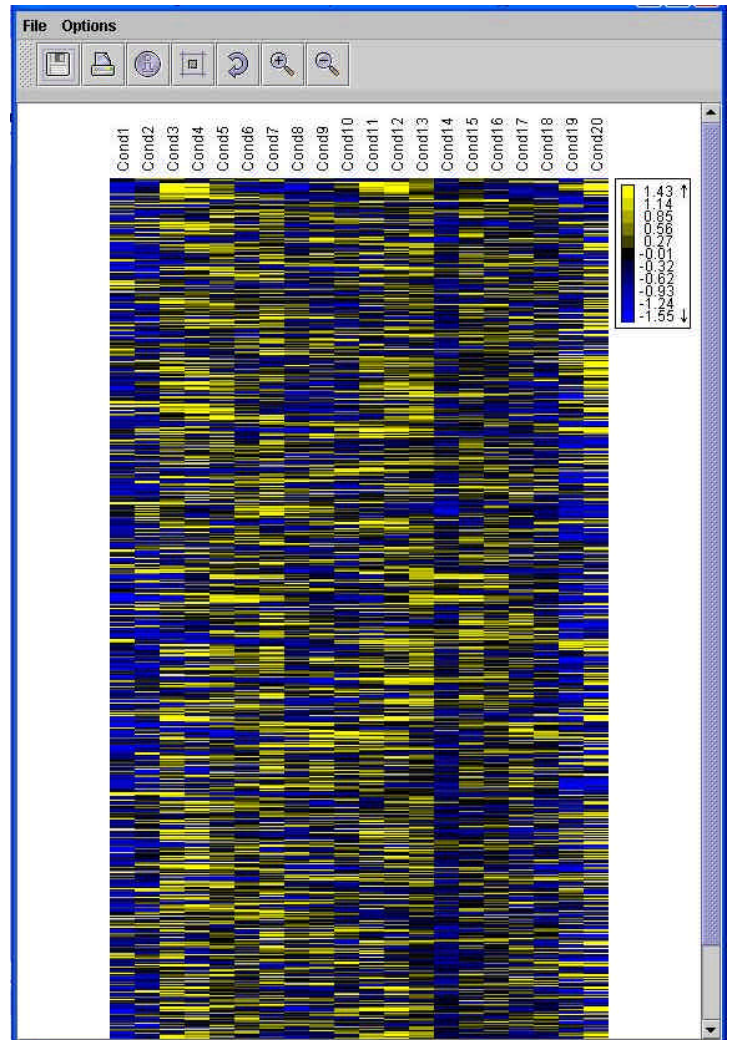


Figure 5.1.1.a: An example of the matrix visualization of dataset A. Only 20 out of 72 conditions are shown here (the user can select which conditions will appear in the visualization). Row and column orders here are as in the input, but reordering can be done in several ways (see below).

In grayscale display grey levels can take values between 0 (black) and 255 (white).

The value is calculated using the following equation:

$$GL = 255 * \left(\frac{X_i - X_{\min}}{X_{\max} - X_{\min}} \right) * 255$$

Where X_i is the expression level that is being rendered, X_{\max} and X_{\min} are the maximal and minimal expression values in the matrix, respectively. In color display the color is determined according to the sign of the expression value, i.e., yellow (red) for positive values and blue (green) for negative values. The user can also set the center of scale (i.e., the value in which colors are switched) to be the average expression value. In this case the color will be determined according to the sign of the expression value minus the average expression value. The intensity (In) of the color is determined according to the equation:

$$In = \left\lceil \log_2 \left(\frac{X_i}{range} + 1 \right) * 255 \right\rceil$$

$$range = \begin{cases} X_{\max} - SV & \text{if } X_i \geq SV \\ SV - X_{\min} & \text{if } X_i < SV \end{cases}$$

Where SV is the value of color switching and X_i , X_{\max} and X_{\min} are as described above. The logarithmic scale is used since it is more sensitive to small values and less influenced by higher extreme values.

In order to reduce the influence of extreme values on the color scale, the maximal, minimal and average expression values are calculated using a sample vector of numbers that are randomly selected from the matrix, and disregarding the top and the bottom 5%.

5.1.2 Similarity matrix

This tool shows the similarity matrix representation used by *Ben-Dor et al. (1999)*. It presents similarity values between expression patterns of all pairs of genes. The similarity measurement used in EXPANDER is the Pearson correlation coefficient.

The symmetric similarity matrix has rows and columns corresponding to elements (usually genes). Similarity values are rendered on the screen either in color or gray levels (colors can be configured through the 'Matrix Display' dialog box, accessible from the 'Options' menu of the main menu bar).

In the color display, yellow (red) indicate a similarity value above average, and blue (green) indicate a similarity value below average. In the grayscale display, a darker cell indicates a higher similarity. A color scale appears next to the matrix (upper right side) and is also available as a mobile frame through the 'Options' menu (or the right click pop-up menu).

When data are clustered, this visualization is available also with rows ordered according to clusters.

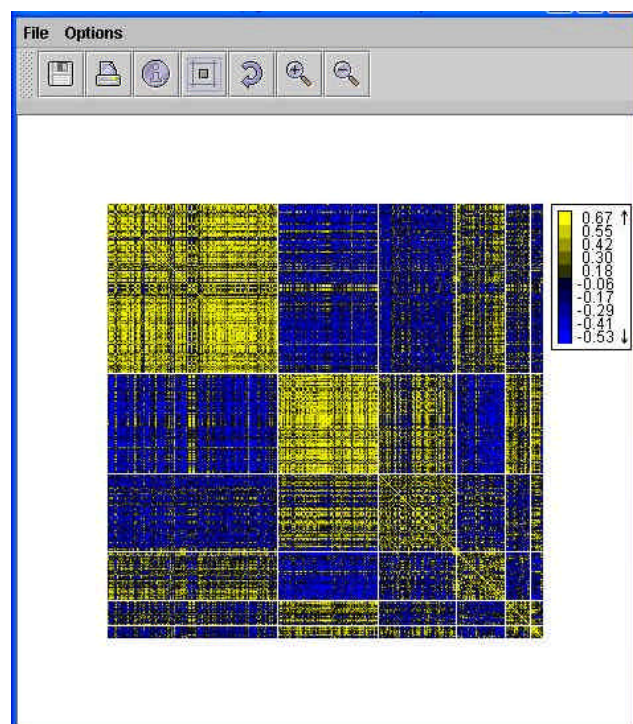


Figure 5.1.2.a. a similarity matrix display for clustered gene expression data of dataset A. Rows and columns here are ordered according to the clustering solution, so that clustered elements appear contiguously. The clusters are noticeable as rectangles along the main diagonal.

5.2 Pattern displays

5.2.1 Mean cluster patterns

This tool displays the mean pattern of each cluster in a separate panel. In each panel the x-axis contains the conditions and the y-axis is the expression value. The mean pattern is displayed along with error bars representing standard deviations.

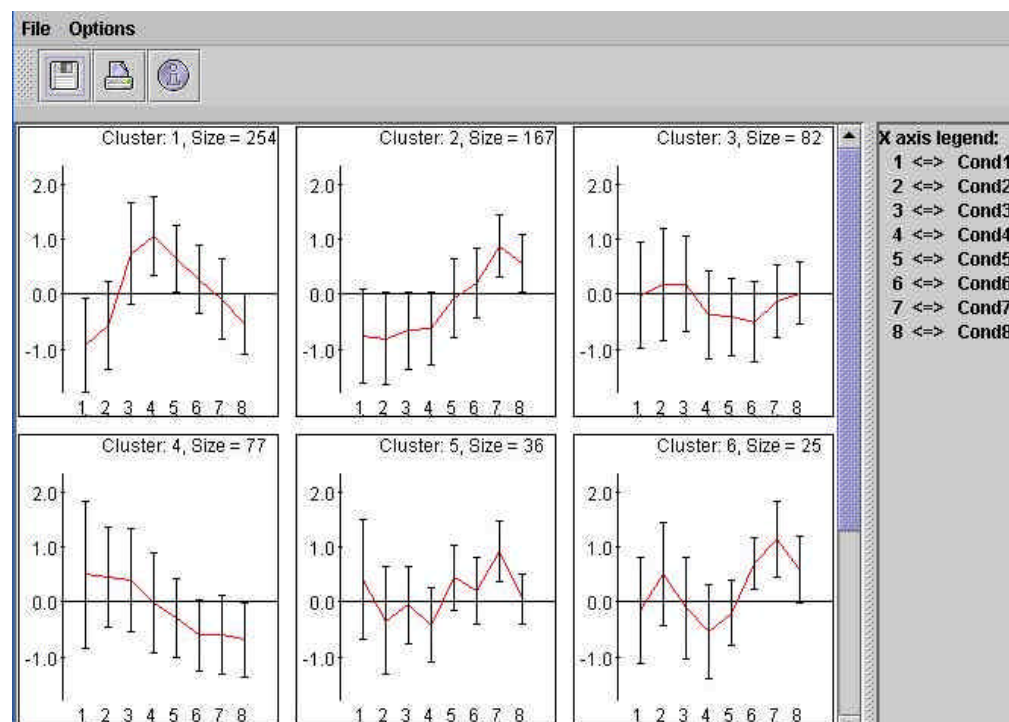


Figure 5.2.1.a. Mean patterns display of the clustering solution of dataset A. Only 8 conditions are displayed (can be determined by the user) so that pattern differences will be noticeable. (The clustering is the same as in fig 5.1.2.a).

Upon clicking on one of the panels, a frame is opened containing a list of all genes in the cluster (probe IDs and gene symbols). The list can be sorted according to one of the columns by clicking on the column header.

5.2.2 Cluster contour

This tool displays a contour of a particular cluster selected by the user. A contour of a cluster consists of two sets of line segments, one representing the mean pattern plus one standard deviation, and the other representing mean pattern minus one standard deviation.

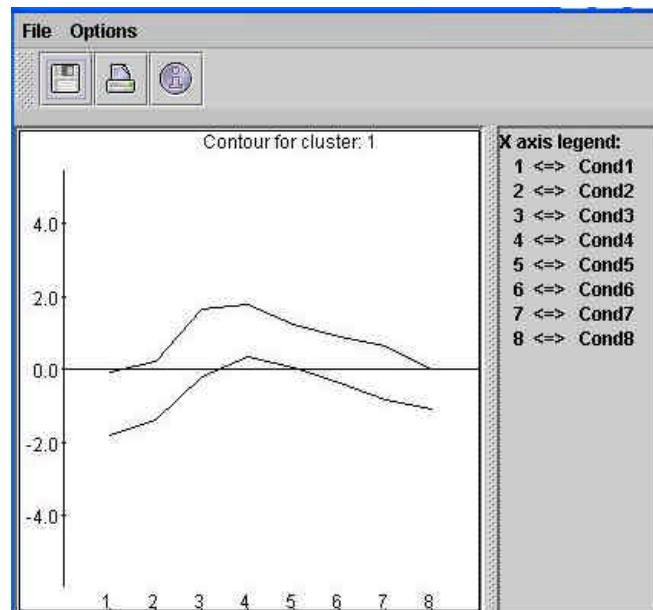


Figure 5.2.2.a. A contour of cluster number 1 from a clustering solution for dataset A. Only 8 conditions are displayed (can be determined by the user).

5.2.3 Patterns of all genes in a cluster

This tool displays a graph of all gene patterns in a particular cluster selected by the user. Each pattern appears in a different color.

Upon clicking on the panel, a frame is opened containing a list of all genes in the cluster. The list can be sorted according to one of the columns by clicking on the column header.

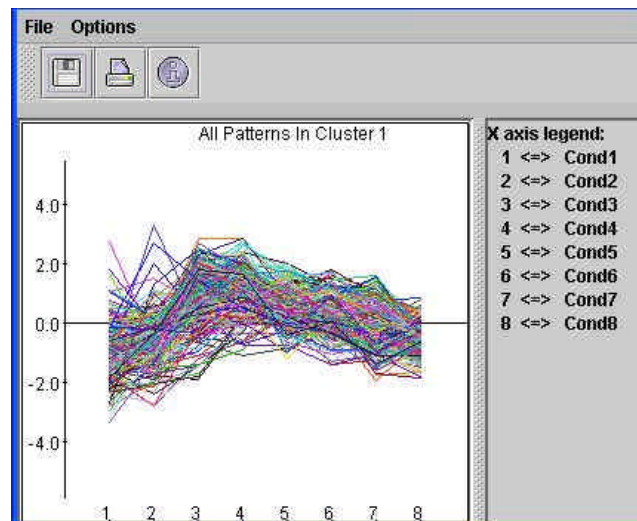


Figure 5.2.3.a. Patterns of all genes in cluster number 1 from a clustering solution for dataset A. Only 8 conditions are displayed (can be determined by the user).

5.3 Scatter plots

5.3.1 PCA analysis display

This tool transforms the original data from an n-dimensional space (where n is the original pattern length) to a 2 dimensional space, so that each gene is represented by a dot on an XY scatter plot. The transformation is based on the PCA (Principal Component Analysis) algorithm (Raychaudhuri *et al.* 2000). If clustering is performed before operating the tool, the dots representing the genes on the chart appear in different colors, according to their cluster numbers. The display tool tip shows the name of the gene represented by the dot located under the cursor.

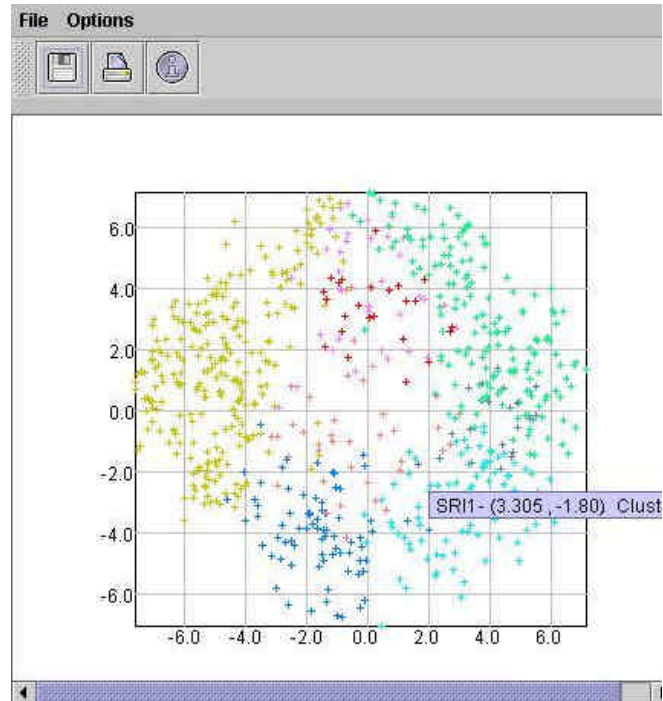


Figure 5.3.1.a. A PCA analysis display of dataset A after it has been clustered. Each color represents a different cluster. Cursor position on the scatter plot corresponds to the gene SR11.

5.3.2 Data plots of two arrays

This tool displays a scatter plot of two arrays, selected by the user. The i^{th} point is (x,y) if the expression value of the i^{th} gene is x in array 1 and y in array 2. For normalized data, points should be located around the $y=x$ line (marked on the scatter plot). Two additional lines corresponding to $y=x+1$ and $y=x-1$ are also marked on the plot. Genes that deviate markedly from these bounds indicate significant overexpression in one array (condition) versus the other, and may be potentially useful for explaining the biological differences between the conditions.

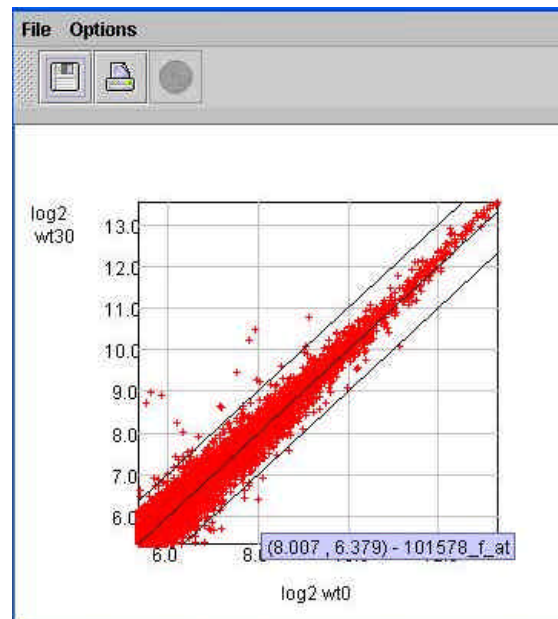


Figure 5.3.2.a. A scatter plot of two arrays from dataset B. The displayed arrays are wt0 - wild type before treatment with ionizing radiation vs. wt30 - wild type 30 min. after treatment with ionizing radiation. Data are well normalized, thus, most points are located around the $y=x$ line. Cursor position corresponds to affymetrix probe with id: 101578_f_at.

5.4 Histograms

5.4.1 Functional analysis display

After performing functional analysis (see section [4.4.2](#) for details) this tool displays a histogram for each cluster, containing a column for each significant functional class (i.e., one that is much more frequent than would be expected at random). The definition of significance depends on the user's selection of threshold p-value. The height of a column is proportional to the percentage of the corresponding functional class in the cluster.

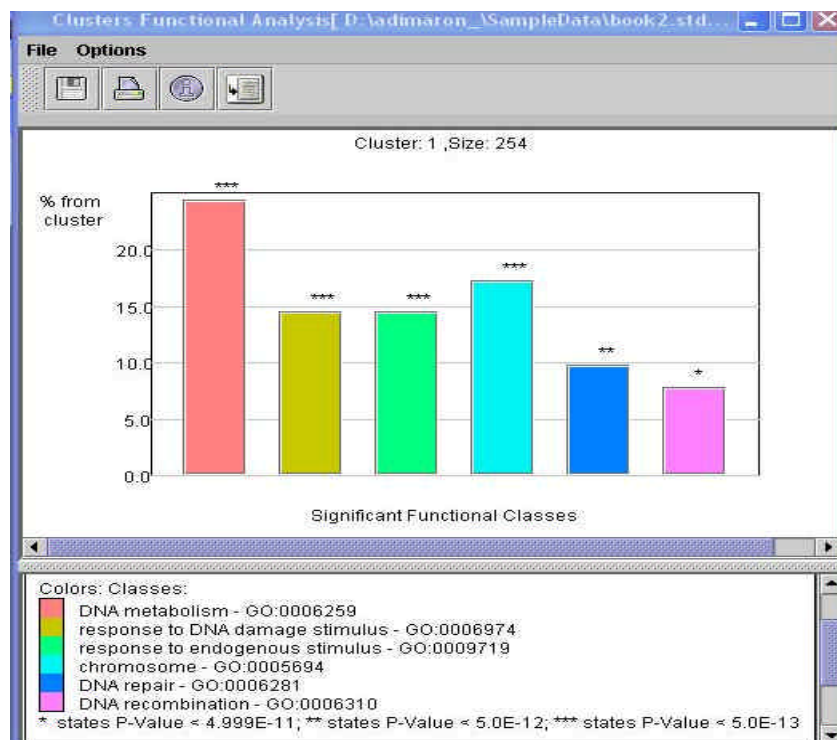


Figure 5.4.1.a. The display of functional analysis performed on a clustering solution dataset A. Threshold p-value had been set to 5×10^{-10} .

Upon clicking on a column, a dialog box is displayed containing the class name, p-value, and a list of the genes in the cluster that belong to the class. For mouse and human these lists are connected to the web, so that when a user clicks on an item in the list, the relevant LocusLink page is displayed with information regarding the gene.

Class Info

DNA metabolism - GO:0006259

p-Value: 1.119E-17

24.409% From cluster

62 elements

Gene ID	Gene Sym...	Probe ID
YPR175W	DPB2	YPR175W
YPR135W	CTF4	YPR135W
YPR120C	CLB5	YPR120C
YPR018W	RLF2	YPR018W
YPL283C	YRF1-7	YPL283C
YPL153C	RAD53	YPL153C
YPL127C	HHO1	YPL127C
YOR144C	ELG1	YOR144C
YOR033C	EXO1	YOR033C
YOL094C	RFC4	YOL094C
YOL090W	MSH2	YOL090W
YOL034W	SMC5	YOL034W
YOL017W	ESC8	YOL017W
YNL220C	YRF1-6	YNL220C

Ok

Figure 5.4.1.b. A dialog box which is displayed upon clicking the DNA metabolism column in the histogram shown in Figure 5.4.1.a.

5.4.2 Promoter analysis display

After performing promoter analysis (see section 4.4.3 for details), this tool displays a histogram for each cluster, containing a column for each significantly enriched transcription factor motif. The required significance level is determined by the user's selection of threshold p-value.

The height of a column is proportional to the ratio of the frequency of the TF motif in the cluster vs. its frequency in the background set. Upon clicking on a column, a dialog box is displayed containing the TF name, p-value, the percentage of promoters in the cluster that contain the motif, relative abundance (frequency in cluster divided by frequency in background set) and a list of the genes in the cluster that contain the motif in their promoters. For mouse and human these lists are linked to the web, so that when a user clicks on an item in the list, the relevant LocusLink (<http://www.ncbi.nlm.nih.gov/LocusLink/>) page is displayed with information regarding the gene.

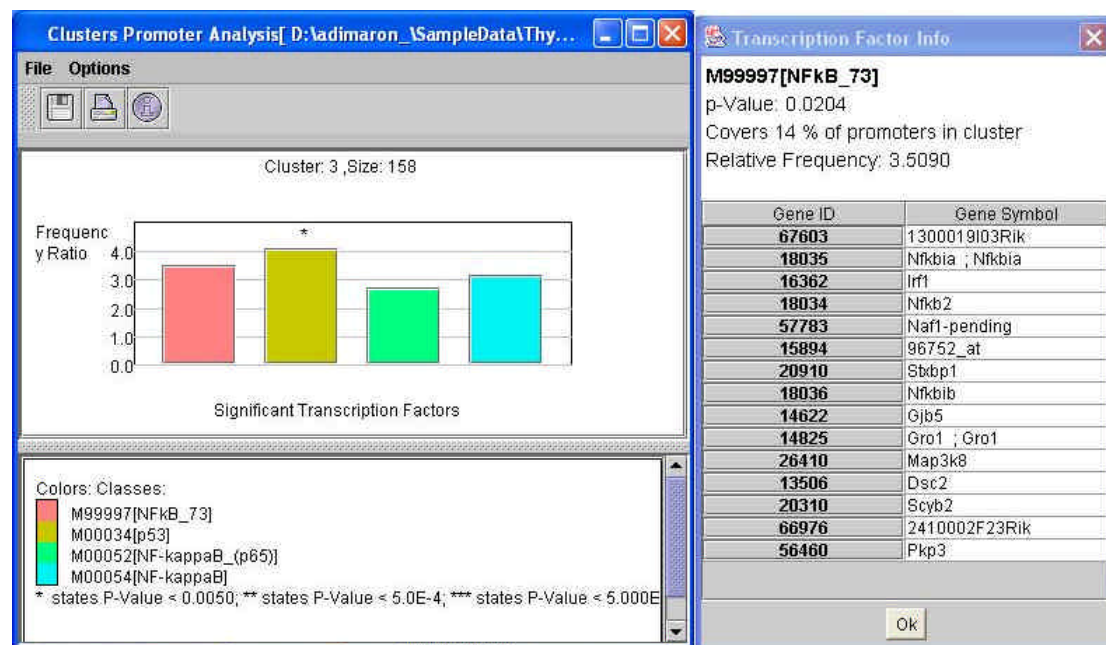


Figure 5.4.2.a. The display of promoter analysis performed on a clustering solution of dataset B. Threshold p-value has been set to 0.05, and the Bonferroni correction for multiple tests has been used.

5.5 Dendrogram trees

5.5.1 Hierarchical clustering results display

After performing a hierarchical clustering (see section [4.2.4](#) for details) this tool displays the resulting dendrogram tree in one of the following manners according to the user's selection:

- A stand alone vertical tree with gene names next to the leaves.
- A vertical tree at the left side of an expression matrix, so that the matrix rows are ordered according to the order of the tree leaves.
- A tree that appears both vertically (at the left side) and horizontally (above) the similarity matrix, with rows and columns ordered according to the order of the tree leaves.

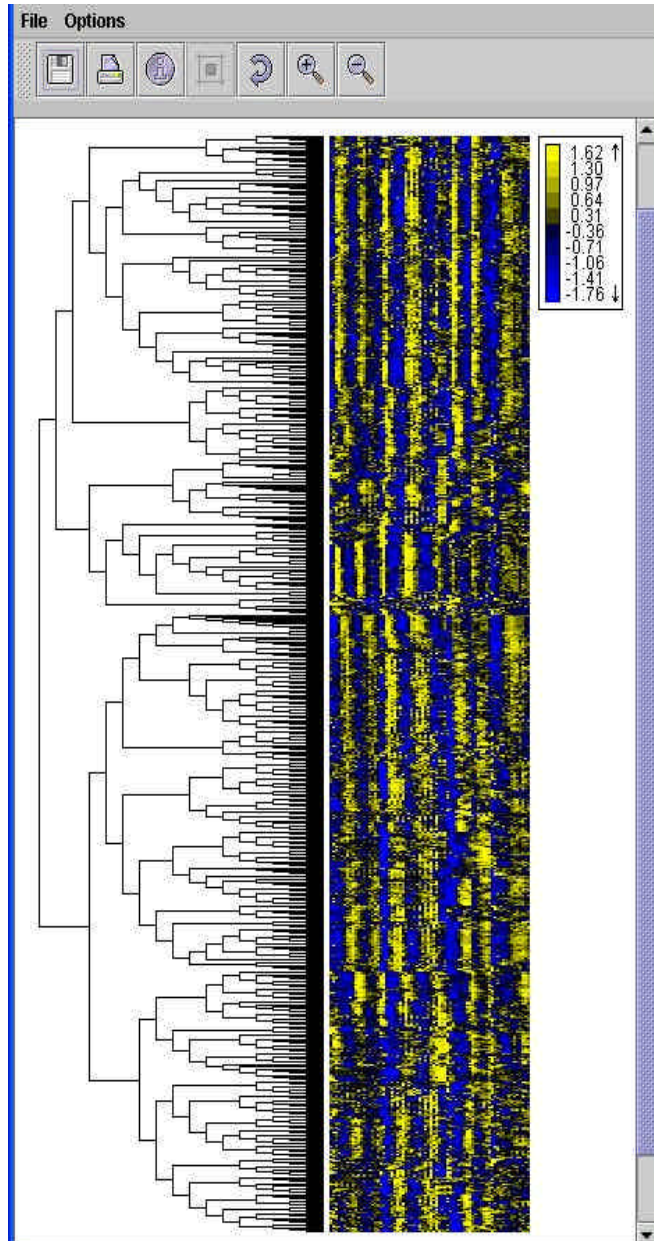


Figure 5.5.1.a. A display of hierarchical clustering results on dataset A. The complete linkage scheme had been used to calculate distances. In this example the dendrogram tree is displayed next to the expression matrix.

5.6 Data tables

5.6.1 Biclustering results data table

After performing biclustering (see section 4.3 for details), this tool displays a table of all biclusters. Filtering can be performed according to: bicluster score, number of genes, number of conditions or maximum p-value for enriched functional class.






File Options						
    						
Bic num	Bic Score	# Conditions	# Genes	Significant Gene Annotation:		
				Annotation1	pValue1	% Annotated1 # of annota...
31	114.571	6	28	cell cycle - GO:0007049	1.611E-4	46 13
41	239.044	7	47	chromosome - GO:0005694	1.128E-4	25 12
29	134.596	4	24	conjugation - GO:0000746	1.009E-4	29 7
19	142.221	6	25	glucan 1,3-beta-glucosidase activity - GO:0004338	8.228E-5	16 4
6	150.677	5	35	maintenance of fidelity during DNA dependent DNA...	4.453E-4	14 5
13	135.364	6	24	maintenance of fidelity during DNA dependent DNA...	1.081E-4	20 5
7	164.751	5	42	nuclear nucleosome - GO:0000788	6.783E-5	14 6
14	193.198	8	29	nuclear nucleosome - GO:0000788	3.13E-12	31 9
15	163.606	6	33	nuclear nucleosome - GO:0000788	3.259E-9	24 8
26	194.86	16	13	nuclear nucleosome - GO:0000788	5.867E-17	69 9
32	121.087	7	26	nuclear nucleosome - GO:0000788	7.924E-8	26 7
11	225.736	6	42	organic acid transporter activity - GO:0005342	3.212E-4	11 5
18	199.098	6	40	pre-replicative complex - GO:0005656	5.083E-6	15 6
12	118.217	4	27	reproduction - GO:0000003	1.977E-4	29 8
16	209.095	8	36	site of polarized growth - GO:0030427	1.134E-4	27 10
1	310.985	8	44	telomerase-independent telomere maintenance - ...	4.779E-8	15 7
2	227.574	7	46			
3	131.085	6	25			
4	155.962	6	33			
5	191.235	7	40			
8	155.383	5	29			
9	191.365	8	29			

Figure 5.6.1.a. Part of the data table displaying the results of biclustering of dataset A. Functional analysis had been performed on the biclusters. Here the biclusters are sorted alphabetically according to the name of the most enriched functional category.

Upon selecting (double clicking) a line in the table, an expression matrix is displayed (see section 5.1.1 for details). It shows the sub-matrix of the expression data for the genes and conditions that belong to the bicluster. An additional column is displayed for each significantly enriched functional class that appears in the table, indicating for each gene, whether it belongs to that class. Gene and condition names appear next to the matrix.

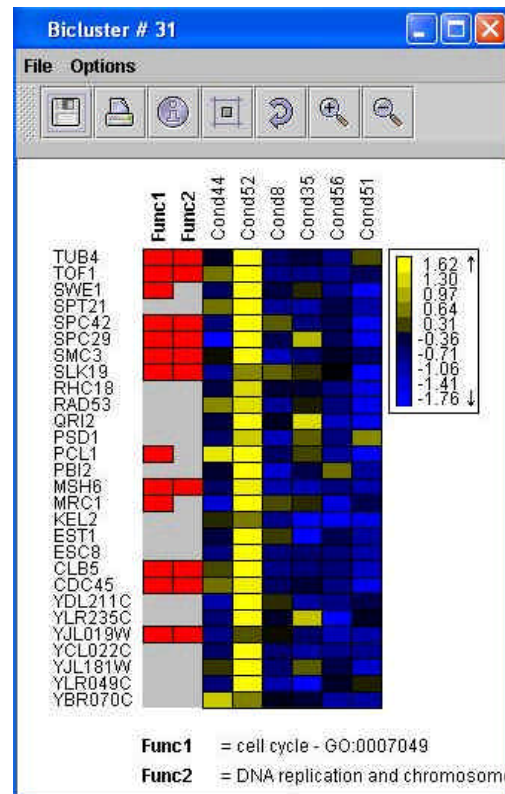


Figure 5.6.1.b. An expression matrix of bicluster #31 from the biclustering results shown in Figure 5.6.1.a. The first two columns correspond to functional classes that were detected as significantly enriched in this bicluster. A red mark in such a column indicates that the gene belongs to this functional class. We can see that there are 13 'cell cycle' genes of which 10 are 'DNA replication and chromosome cycle' genes in this bicluster.

6 Software development and architecture

In this chapter we describe the software development process and outline the architecture and its main building blocks. We also explain our considerations in making the key decisions regarding the development.

6.1 Selecting the development language

EXPANDER was developed in Java for the following reasons:

- ✍ The java programming language is object oriented, and thus suitable for such a large scale application that requires many different modules that interact with one another.
- ✍ A code written in Java is cross platform, i.e., it runs with little or no changes on different operating systems. This enables biologists to use the application on any operating system.
- ✍ The Java language incorporates graphical implementations for window application programming, that give solutions to differences between windowing systems of different operating systems (e.g. Win32 vs. Linux), and can be easily expanded. This is very important to the development of graphical visualization tools such as this one.
- ✍ It is relatively simple to run external modules written in other (more efficient) programming languages via an application written in Java. This is achieved by using the *Runtime* class supplied by SUN.

6.2 Architectural considerations and overview

6.2.1 Architecture considerations

The main guidelines leading me in my design were:

- ? Create a clear separation between data management, data analysis and graphic display.
- ? Exploit the advantages of object oriented programming (inheritance, encapsulation etc.) in order to make the code as simple and elegant as possible, and to simplify the addition of new functionalities.

6.2.2 Overview

The considerations described above led to the planning of the scheme described in Figure 6.2.2.a. The program structure is presented as three main modules, the **Data Management** module that consists of the *Data package*, the **Data Analysis** module that consists of the *Algorithm* and the *Visualization Tools* packages and the **Graphical Interface** module that consists of the *Display*, *Display Frame* and *Dialog* packages. During the program operation the modules interact with each other. The *Utility Package* contains several classes that are used by all modules, and thus, is an open access package. A more detailed overview scheme will be given in section 6.8, after the components have been described.

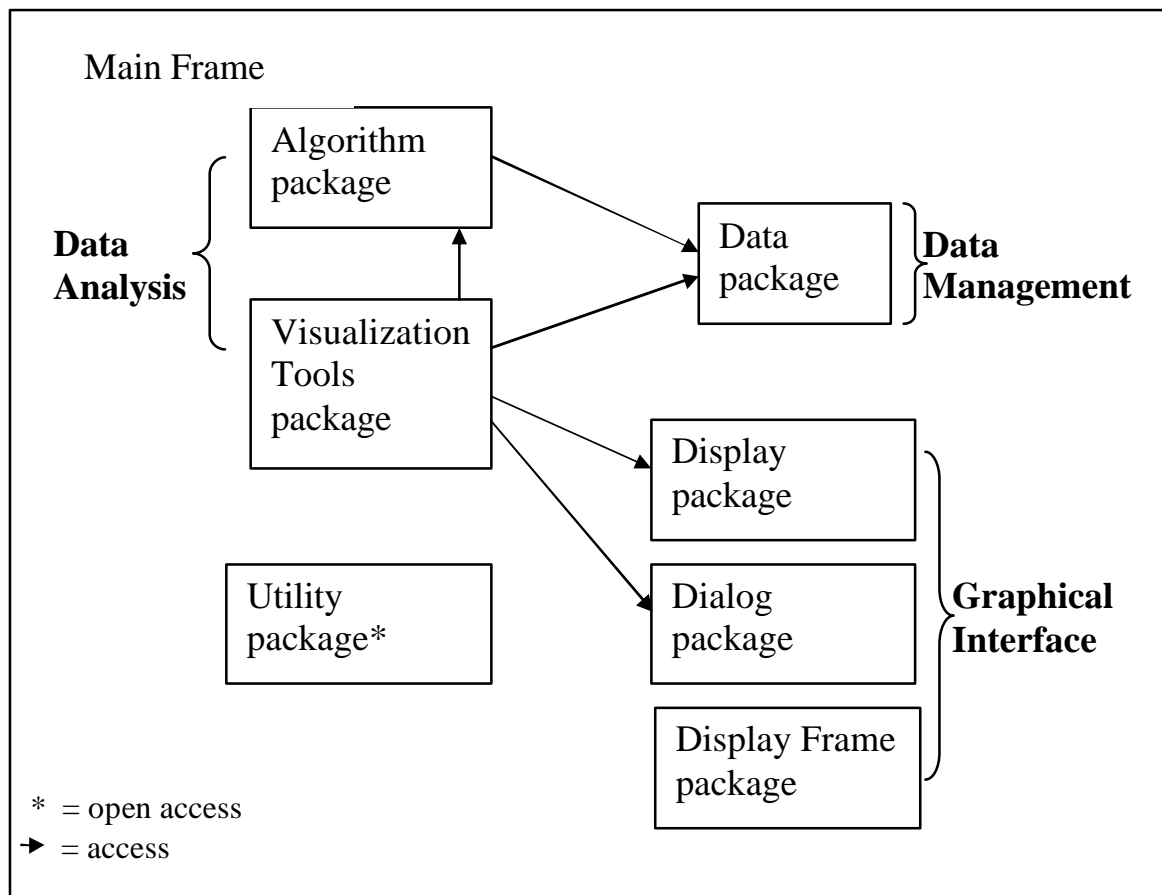


Figure 6.2.2.a. A scheme describing the general design of the functional modules in the program. Each general module (package) is represented by a rectangle, and interactions between modules are represented by arrows.

6.3 Data management

The following classes are used in EXPANDER to store and access the data, while the application is running:

6.3.1 The FloatMatrix class

This class was created in order to simplify operations on matrices containing numbers in floating point representation. Such matrices are often used in the program. Each object contains a two dimensional array of floating point numbers, and some additional parameters such as the average value, whether or not the matrix is transposed, etc. In addition to ordinary 'get' and 'set' methods, this class also contains some matrix normalization and standardization schemes that can be applied on any matrix, and are used for microarray data normalization and standardization.

6.3.2 The BasicElement and ElementArray classes

The *BasicElement* class was created in order to represent one gene or one condition in the data. For each gene/condition, an object of *BasicElement* is created, containing its name, id, whether or not it is being used for current calculations and a few other details.

The *ElementArray* class holds and manages the entire set of genes/conditions in the data. It contains an array of objects of type *BasicElement*. In addition to ordinary 'get' and 'set' methods, this class can perform variable queries regarding the data such as: getting an element index by its name, getting an array of names of the used elements only (elements that are currently flagged as used for analysis), etc.

6.3.3 The MainData class

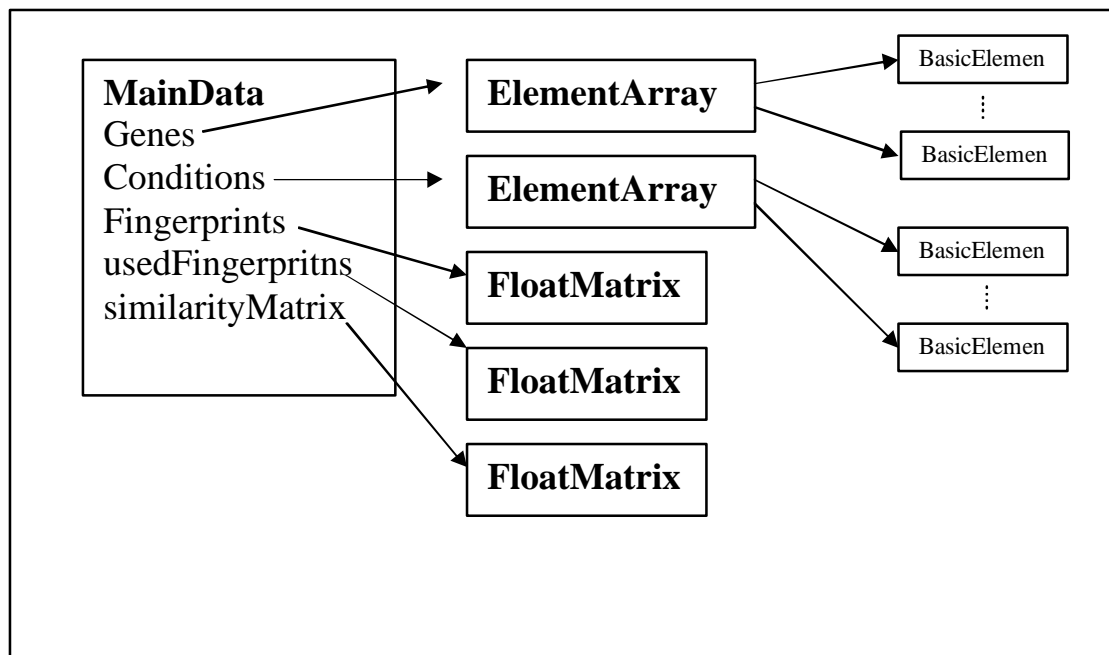


Figure 6.3.3.a. A scheme describing the structure of the *MainData* module of the program. Each rectangle represents an object of a class. Class names are written in the rectangle. The names of all main data members in the *MainData* class are written in the *MainData* rectangle, and the arrows are pointing at the corresponding class objects.

This class holds and manages most of the data used by the program. A single object of this type exists at any given time, and is initialized when the user requests to load data from an input file. The structure of the class is described in Figure 6.3.3.a.

The class contains two objects of type *ElementArray* (one for genes and one for conditions) and three objects of type *FloatMatrix* (for input expression data, preprocessed expression data and similarity data). It also contains some additional parameters such as the recently used standardization method, the used similarity type, the input data type, etc. In addition to ordinary 'get' and 'set' methods, this class also contains methods for loading input data from file, for preprocessing data (filtering, normalizing and standardizing expression data), calculating similarity matrix according to expression data, and writing data into files (this is required for operating external modules via EXPANDER).

6.3.4 The Bicluster and BicSet classes

The *Bicluster* class was created in order to contain all information regarding one bicluster calculated by operating a biclustering algorithm. It contains a vector of

indices of all genes in the bicluster and a vector of indices of all conditions in the bicluster, as well as a score. It also contains required 'get' and 'set' methods.

The *BicSet* class holds and manages the entire set of *Bicluster* objects generated by a biclustering algorithm. In addition to the set of objects of type *Bicluster*, this class also contains a vector of significant functional annotations for each bicluster (if calculated). It contains some simple 'get' and 'set' methods as well as several query methods such as getting an array of all gene/condition names in a particular bicluster, or getting a *FloatMatrix* object containing expression data of a particular bicluster.

6.3.5 The Preferences class

This class contains all of the user's preferences for the application settings. These include the threshold p-value for functional enrichment analysis, the organism being studied, matrix display preferences (colors etc.), biclustering algorithm parameters, and more. Only one object of this type exists when the application is running, and it reads and writes itself from and into a preferences file, saved after the application is closed, so that user preferences will be saved from one session to another.

6.4 Data analysis

Each of the top level analysis methods used by EXPANDER is incorporated in a class that is derived from the general abstract class *Algorithm*.

6.4.1 The Algorithm class

The *Algorithm* class is an abstract class that represents a general algorithm. It is used as a super-class from which all algorithm classes in the program are derived. It contains a reference to the *MainData* object, as well as a list of temporary files that are to be deleted after completing the operation of *Algorithm* by incorporating a clean-up method. It also contains the virtual *getType* and *operate* methods (these are methods that must be implemented in derived classes).

6.4.2 The ClusteringAlgorithm class

The *ClusteringAlgorithm* class is derived from the *Algorithm* class and is a super-class of all clustering algorithm classes. It contains an array of integers that holds the clustering results, and some additional data such as clusters homogeneity and separation measurements.

This class implements all operations that are to be performed by all clustering algorithms, such as calculating homogeneity and separation of clustering solutions, calculating the mean patterns of clusters, writing a clustering solution into a file, and more.

6.4.3 Classes extending ClusteringAlgorithm

A separate class has been created for each clustering algorithm implemented by EXPANDER. Each such class is derived from the *ClusteringAlgorithm* class, and contains a different implementation for the 'operate' method. The *SOMAlgorithm* and *KMeansAlgorithm* classes contain the algorithm steps in the code, whereas the *ClickAlgorithm* class operates an external module, and reads the resulting output into its clustering results vector.

6.4.4 Classes extending Algorithm

Some of the algorithms implemented by EXPANDER are not clustering algorithms. These are also derived from the generic *Algorithm* class. An example for such a class is the *PCAAlgorithm* class which implements the PCA algorithm in order to project each expression pattern from n dimensions to two dimensions.

6.4.5 Handling external modules via Algorithm classes

Some of the algorithms act as interfaces which operate external modules in order to perform the required analysis. An example for such an algorithm is the CLICK clustering algorithm or the functional analysis algorithm. These classes are required to operate an external script/application located under the EXPANDER directory. To handle the operation of such external modules we have used the java class *Process*. In order to overcome a documented Java bug which causes the Process object to 'hang' during operation, we have created the class *ExternalProcessHandler*. An object of

this class is created for each process, and operates by reading the input and error streams generated by the process, and generating relevant messages to be displayed by EXPANDER.

6.4.6 From data analysis to display: the visualization tools

In the process of generating and manipulating a visualization in EXPANDER, some calculations are carried out (with or without using an *Algorithm* class), after which, a suitable display object is created. In some cases, additional input is requested from the user. These stages are all performed by the visualization tool.

The visualization tool is the 'glue' that connects the data, the analysis and the display. It is created by the main frame when the user requests the use of a tool (for example: expression matrix). The tool has constant access to the main data objects, and it creates the display panel and frame to display analysis results.

Each visualization tool is defined in a class that is derived (directly or indirectly) from the abstract super-class *VisualizationTool*.

The VisualizationTool class

The *VisualizationTool* class is a super class from which all visualization tool classes are derived. It creates and holds a *DisplayPanel* object and a *DisplayFrame* object. It also holds a reference to the *MainData* object. It defines all operations that are required in all or most of the visualization tools (such as the method for creating a display frame).

The VisualizationToolWithClust class

The *VisualizationToolWithClust* class is derived from the *VisualizationTool* class. It holds a reference to a *ClusteringAlgorithm* object, as well as methods designed to handle clustering results (e.g., a special method for creating a table that contains clustering information). All visualization tools that operate on clustered data only are derived from this class. Two examples for such classes are the *FunctionalAnalysisTool* and the *PromoterAnalysisTool* that are both derived from the *VisualizationToolWithClust* class. The *FunctionalAnalysisTool* class operates an external module that calculates the significance of different functional classes in the

each cluster, and detects significantly enriched classes (for details see section 4.5.2). The *PromoterAnalysisTool* class operates an external module, the PRIMA software, that performs statistical tests in order to identify transcription factors (TFs) whose binding site profiles are significantly enriched in the different clusters (for details see section 4.5.3). Both classes utilize the *ExternalProcessHandler* class, described in section 6.4.5, to handle these external processes.

6.5 The graphical Interface

The graphical interface is composed of the input dialogs (used for data input) and the different display panels that are used for visualizations and display of analysis results.

The *Dialog* package contains the classes that are used for requesting input data from the user. All are derived from the java class *Dialog*.

To generate the different displays, a class was defined for each display type. Each display class is derived (directly or indirectly) from the abstract super-class *DisplayPanel* (described below). The Display package contains all classes that are used to display graphical visualizations on the screen. Drawing on the screen is performed using the Swing graphical user interface library (<http://java.sun.com/docs/books/tutorial/uiswing/>).

6.5.1 The DisplayPanel class

The *DisplayPanel* class is derived from the Swing class *JPanel*. It implements all operations that are required for the graphical displays such as: a print method that enables sending its contents to be printed, a paint method that is called whenever the panel is repainted on the screen, and a mouse motion listener event handler that detects mouse motions and operates a method that updates the tool-tip text according to the position of the cursor on the display.

6.5.2 Displaying matrices

The matrix display is performed using the *MatrixDisplayer* class, which is derived from the *DisplayPanel* class. Matrix colors are rendered according to the float values in each position in the matrix.

Color rendering is performed once upon display initiation, and saved for as long as the *displayer* object exists. This is because recalculating the colors each time a 'paint' event takes place is very time consuming. The colors are kept in a data structure called *colorMatrix*, which is an array of objects of type *ColorArr*. Each *ColorArr* represents one color, and contains a vector of positions on screen (x,y) which are to be colored in that color.

Whenever a 'paint' event takes place, the matrix is repainted on the screen via the '*onDisplay*' method which uses the Java *Graphics* class to change the color and then fill all rectangles that are to be colored in that color (according to the *colorMatrix* data structure). Rectangle size is determined according to the scale used, and is changed when the user operates the zoom in/out option.

6.5.3 Displaying charts

The following chart displays are implemented in EXPANDER: XY scatter plot, pattern displayer and histogram. These are implemented in three separate classes, *XYScatterDisplayer*, *PatternDisplayer*, and *Histogram*, which are all derived from the *Chart* super-class.

The *Chart* super-class holds all parameters and methods required for drawing the axis system and labels. The display magnitude (scale) is determined according to the size of the frame, and is updated automatically whenever the frame is resized (i.e., whenever a 'paint' event takes place).

The *XYScatterDisplayer* class holds a vector of points (x, y, point name, point color) to be displayed on the chart. Points are painted on the screen in the form of a '+' symbol, which is centered round the relevant (x, y) values. This class also holds a vector of point positions in pixels on the screen. This vector is updated whenever the display is repainted on the screen. It is used in order to efficiently display the appropriate tool-tip text whenever the cursor is placed on or very close to one of the points.

The *PatternDisplayer* class holds a vector of patterns. Each pattern is described by a series of float numbers that represent the y values of the pattern, a name, a series of error bar sizes (one for each point) and a color (the x values of the pattern are consecutive, e.g., time points). This class also holds a vector of pattern positions in

pixels on the screen. This vector is updated whenever the display is repainted on the screen. It is used in order to efficiently display the appropriate tool-tip text whenever the cursor is placed on or very close to one of the patterns.

The *Histogram* class holds a vector of all histogram columns. Each column is described by its position, width, height, name, color and reference to a dialog box that contains some information regarding the column. On the display each column is defined by the application as a button, which when clicked, causes the display of the associated dialog box.

6.5.4 Displaying data tables

Data tables are used by EXPANDER to display clustering results, bi-clustering results, the contents of a selected cluster, and other types of required information. The Swing class *JTable* is used in order to display a data table. Data are input into a table using a table model (implemented in class *ClusteredDataTableModel*). In order to allow sorting a table according to the contents of a column, a *TableSorter* class was written.

The biclustering results table can be filtered according to the user selection. This is performed in the 'updateTable' method of the class *BicResDisplayer* by using a *BicsFilter* object. The *BicsFilter* class holds various parameters that define a filter operation, such as: minimal and maximal score, minimal and maximal number of conditions per bicluster, minimal and maximal number of genes per bicluster, threshold functional annotation p-value, etc. This class implements a method that receives a *Bicluster* object, and returns whether or not it should be displayed under the current filter definitions.

6.5.5 Displaying dendrogram trees

Dendrogram trees are used in EXPANDER to display the results of a hierarchical clustering algorithm. The *TreeNode* class is used to hold all dendrogram tree data. In order to display a dendrogram tree on the screen the *DisplayTree* class was defined. This class holds the tree in the form of a *TreeNode* object as well as an array of leaf labels and leaf positions on screen. It also implements several drawing methods that are used to draw the tree on the screen.

6.5.6 Creating the display frames

In order to show a display panel on the screen, a frame that contains this panel must be created. This frame is created using the *DisplayFrame* class (or one of the classes derived from it). An object of that type is created by the visualization tool in the 'createDisplayFrame' method. The display frame contains a separate menu bar, which enables the user to perform operations on the display, such as zoom in, zoom out, save etc.

Each *DisplayFrame* object contains one *DisplayPanel* object, a tool bar containing buttons that enable operations on the display, a pop-up menu and a dialog box with information regarding the display. Most of the methods implemented in the *DisplayFrame* class are event handlers for handling different menu options.

6.6 The Main Frame - How it all comes together

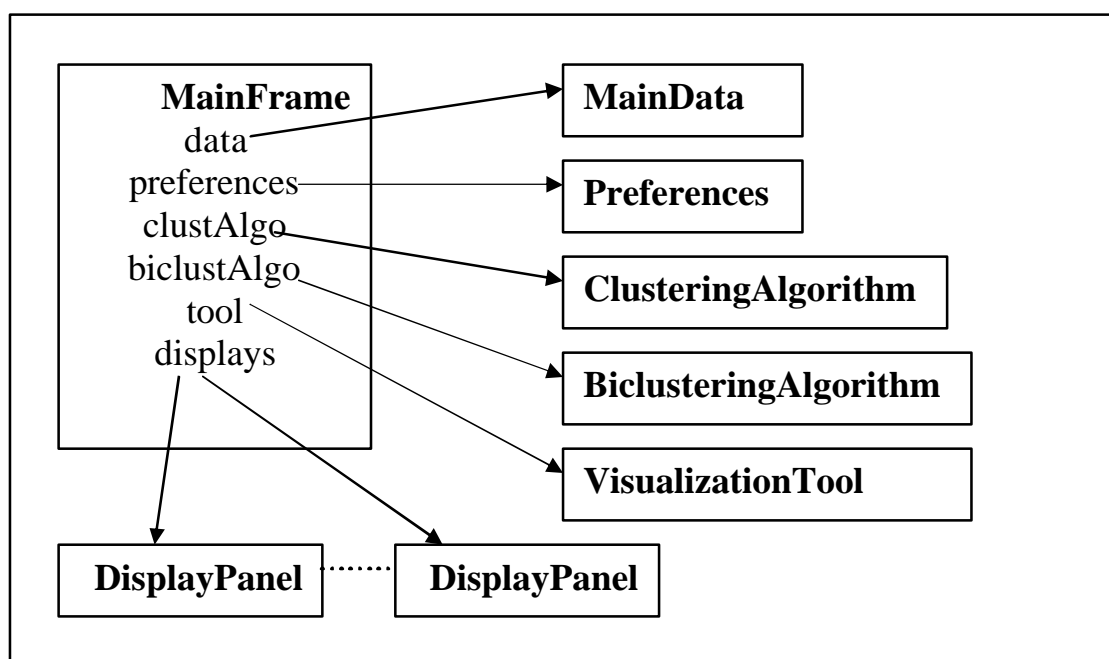


Figure 6.6.a. A scheme describing the *MainFrame* class and its key components. Data members of the class are listed in the *MainFrame* rectangle, and arrows are pointing at the rectangles representing the different objects that comprise them.

The *MainFrame* class is derived from the Swing class *JFrame*. A single object of this type is created upon running the application. This object exists for as long as the application is running, and it can be referred to as the application 'manager'.

The main frame contains the main menu bar, through which the user operates the different analysis and display tools, as well as a status bar, through which the application can send messages to the user regarding the application's status. When an input data file is loaded by the user, the main frame creates and holds the *MainData* object. The clustering and biclustering algorithms are also created and operated by the main frame upon user's request, and are kept as data members. The main frame also holds a vector containing all open displays, and an object of type *Preferences*, containing all currently selected application settings.

Most of the methods implemented in the main frame class are event handlers for handling different menu options.

6.7 The utility Package

This package contains classes for which no objects are formed. These classes are designed to contain information/methods that are relevant for the whole application at all times.

6.7.1 Handling float vectors – the VecCalc class

The *VecCalc* class contains various methods that are designed to handle floating number vectors. It implements operations such as sort, find maximal value, find minimal value, calculate average, calculate standard deviation, etc. The methods can be used by all objects in the application, and do not require the existence of an object of this type.

6.7.2 The Strings class

All constant strings used by the application (e.g., messages to the user) are defined in the *Strings* class. This simplifies the process of changing text since it allows all changes to be performed only once, and in a well known, fixed place.

The *Strings* class also defines several string manipulation methods that are not available in the *Java.String* class, and are required by several classes in the application. An example for such a method is the *floatToStr* method, which deals with displaying a float number as a string.

6.7.3 The Constants class

All constant integers (enumerations) that are used in more than one place in the application are defined in the Constants class. Again, this simplifies the process of changing such enumerations, since it allows all changes to be performed only once, and in a well known, fixed place.

6.7.4 Connecting to the WEB – the URLHandler class

Some of the data tables displayed by EXPANDER contain gene names, which can be used as links to web pages containing information regarding those genes (upon clicking such a name, a web browser is opened, and displays the relevant information). To implement this feature I have created the *URLHandler* class. The *URLHandler* class operates the web browser via the *Java.Runtime* object, by sending the appropriate command line, selected according to the operating system on which the application is running. In case of failure, this class generates the proper error message. Once again, the methods can be used by all objects in the application, and do not require the existence of an object of this type.

6.8 A detailed overview

We are now ready to view in more detail the overall architecture of the system. Figure 6.8.a shows a detailed version of the scheme displayed in Figure 6.2.2.a. The main classes in each package are displayed within the package rectangle. The relations between the different classes are described using arrows.

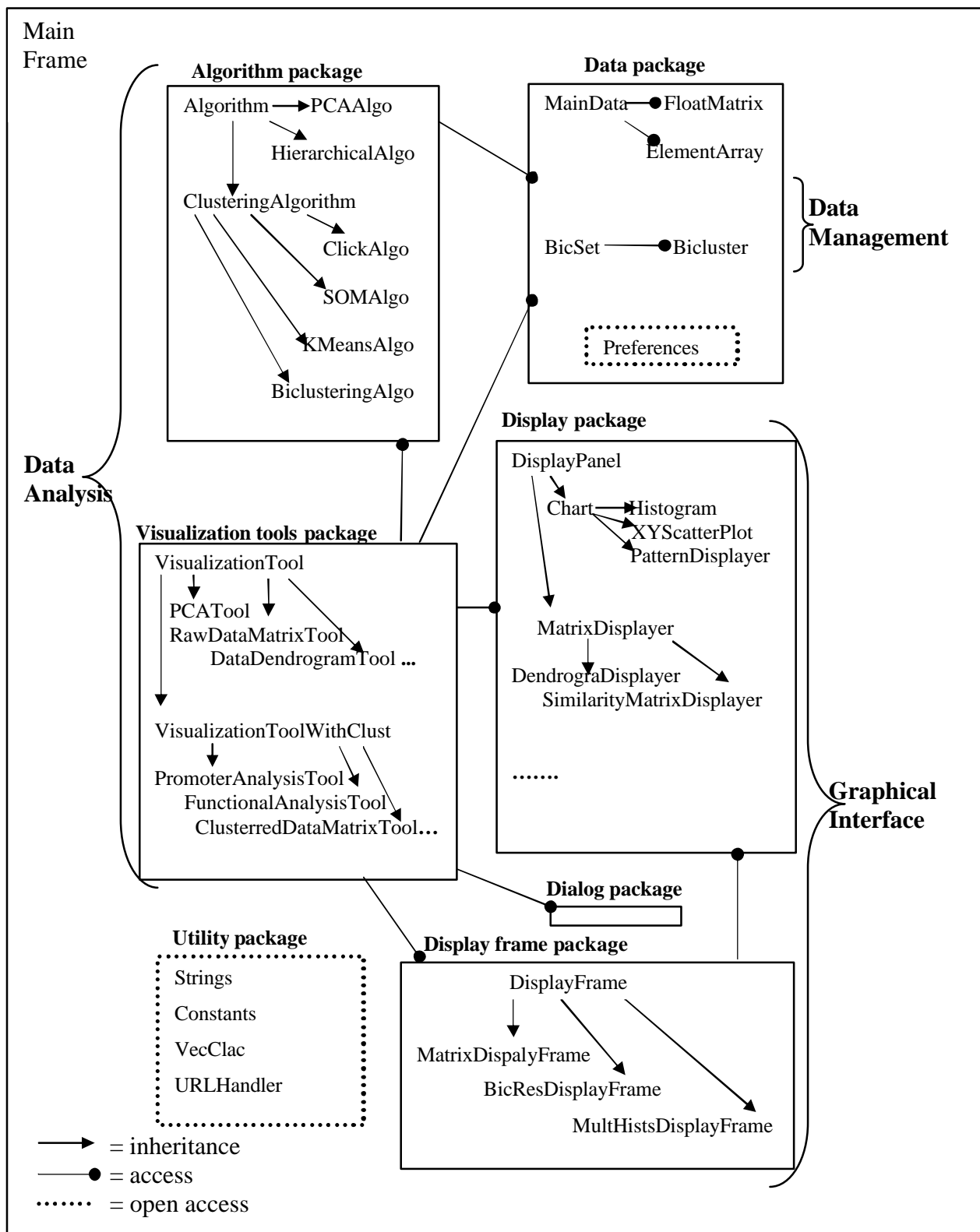


Figure 6.8.a. A General scheme, describing the different packages, the central classes that they contain and the different relationships among them. The MainFrame class creates and manages all other components. Classes of the Algorithm package can access instances of the Data package classes. Classes of the Visualization tool package can access, create and manage instances of classes from the Data package, Algorithm packages, Dialog package and Display Frame package, that are required for the operation of these tools. The Utility package has open access since it contains classes that are required by all components.

7 EXPANDER as an instrument in the hands of the researcher

In this chapter we present three examples of EXPANDER-based analysis of biological datasets. This allows us to demonstrate various capabilities of the software and to draw novel biological conclusions.

7.1 Example 1: Analysis of oligonucleotide array data from the mouse lymph nodes

7.1.1 The data

The data we describe here were generated as part of an attempt to dissect the DNA damage response using gene expression profiles. The experiments were conducted by Sharon Rashi-Elkeles, Ran Elkon from the Shiloh lab in cooperation with Nir Weizmann and Ari Barzilai from the George S. Wise Faculty of Life Science in Tel-Aviv University, Ninette Amariglio and Gideon Rechavi from the Department of Pediatric Hemato-Oncology unit of Functional Genomics at the Sheba Medical Center, and Chaim Linhart, Roded Sharan and Ron Shamir from the computational genomics laboratory in Tel Aviv University.

Atm is a protein kinase encoded by the gene that is mutated in the human disorder ataxia-telangiectasia (A-T). The disease which is characterized by progressive neurodegeneration that leads to severe ataxia and many other defects including immune deficiencies, cancer proneness, chromosomal instability, and ionizing radiation sensitivity (*Chong et al. 2000*). Atm activity is required in cell cycle checkpoints and DNA repair after exposure to ionizing radiation. ATM-deficient cells exhibit an extremely high sensitivity to ionizing radiation and to multiple double stranded breaks.

In this study, global transcriptional responses were recorded in wild-type and in Atm-deficient lymph node tissues of mice exposed to whole body irradiation with 15 Gy of IR. mRNA was collected 0, 30 and 120 min after irradiation. Affymetrix GeneChips MGU74Av2 were used in this study. The chips containing above 12,000 probe sets, of which 6000 correspond to functionally characterized mouse genes and the rest

correspond to ESTs. Samples from untreated mice were probed in independent hybridization triplicates (three repetitions) and samples from irradiated mice were probed in independent hybridization duplicates (two repetitions). A representative expression level for each probe set in each of the six tested conditions was computed by averaging the probe-set signal intensities in the replicate arrays. This study has not been published yet.

7.1.2 Loading the data

The input file used in this analysis contains data for 6982 gene probes (Affymetrix IDs) and 6 conditions. These genes remained after filtering out all genes that were marked "Unpresent" by the Affymetrix software (i.e., genes that are not expressed in lymph node cells under any of the 6 conditions). Expression levels under 40 were arbitrarily set to 40. The conversion file contains the LocusLink IDs corresponding to probe Affymetrix IDs. Figure 1.1.2.a contains an image of the input dialog box that was used to load the data.

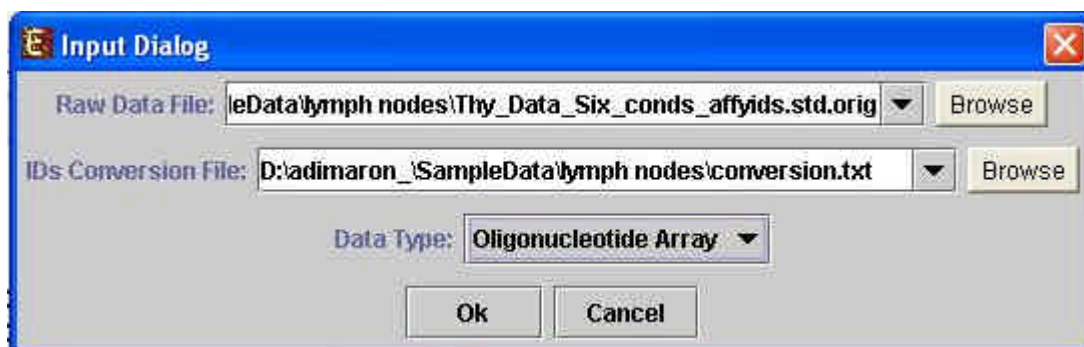
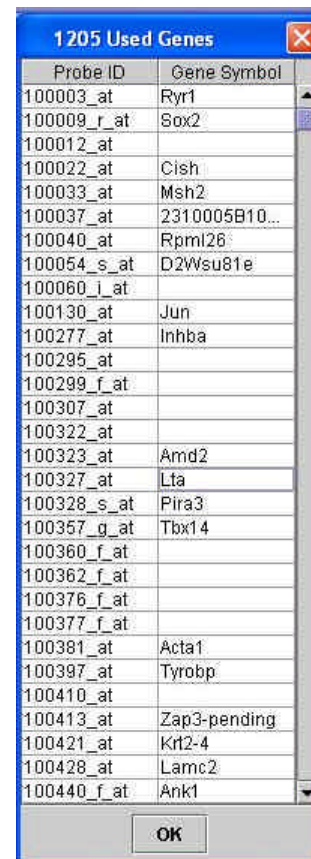


Figure 1.1.2.a. The file input dialog box that was used to load the input data. The Raw Data File Name field contains the name of the expression data file. The IDs Conversion File field contains the name of the file that contains the LocusLink IDs corresponding to each gene ID from the expression file. The Data Type field was set to Oligonucleotide Array.

7.1.3 Preprocessing the data

A fold change filter was applied, so that only genes changed by a factor of at least 1.75 across the six tested conditions were selected. 1205 of the probe sets met this criterion. The remaining genes were displayed in a list shown in Figure 7.1.3.a.



Probe ID	Gene Symbol
100003_at	Ryr1
100009_r_at	Sox2
100012_at	
100022_at	Cish
100033_at	Msh2
100037_at	2310005B10...
100040_at	Rpmi26
100054_s_at	D2Wsu81e
100060_i_at	
100130_at	Jun
100277_at	Inhba
100295_at	
100299_f_at	
100307_at	
100322_at	
100323_at	Amd2
100327_at	Lta
100328_s_at	Pira3
100357_g_at	Tbx14
100360_f_at	
100362_f_at	
100376_f_at	
100377_f_at	
100381_at	Acta1
100397_at	Tyrobp
100410_at	
100413_at	Zap3-pending
100421_at	Krt2-4
100428_at	Lamc2
100440_f_at	Ank1

Figure 7.1.3.a. Part of the list of genes that remained after applying the fold change filter.

Next, data were standardized so that the expression levels of each gene would have mean 0 and variance 1. Figure 7.1.3.b shows the menu option and dialog box that were used to operate the standardization.

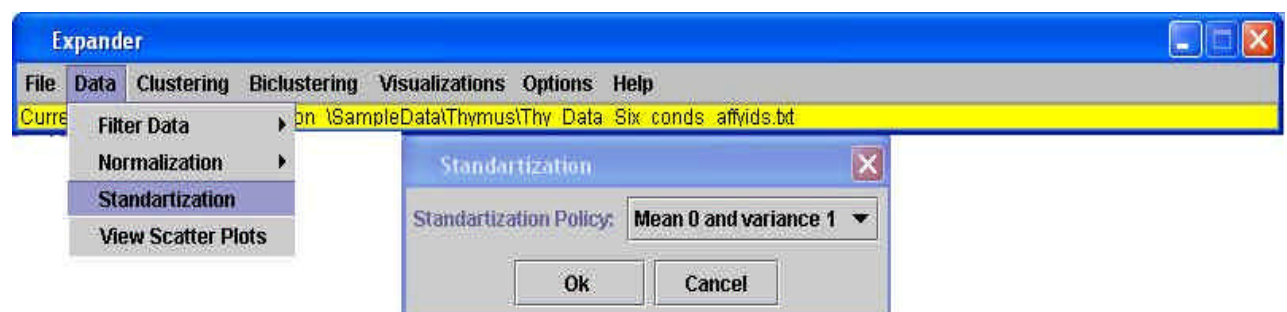


Figure 7.1.3.b. The menu option and dialog box that were used to operate the standardization.

7.1.4 Viewing raw data

The raw data were viewed using the Raw Data Matrix visualization. Figure 7.1.4.a. shows the displayed matrix. The resolution was changed using the *Zoom in* and *Zoom out* options. When using high resolution, gene names were displayed next to their corresponding rows in the matrix (not shown in figure).

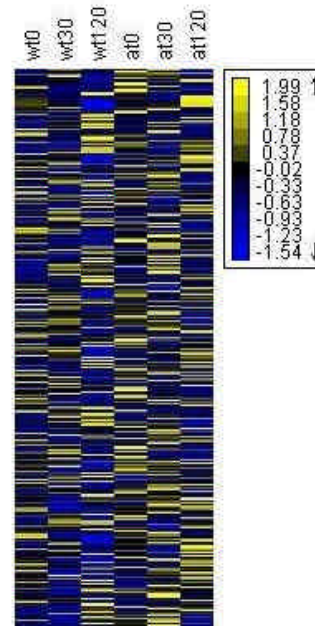
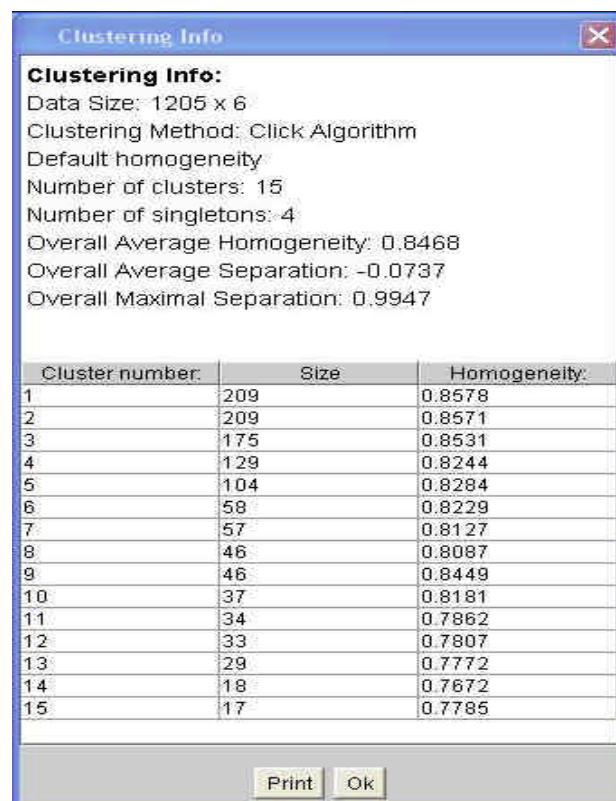


Figure 7.1.4.a. A fraction of the raw data matrix display of the data after filtration and standardization have been performed. A color scale appears at the top right corner of the display.

7.1.5 Clustering the data

The CLICK algorithm was used to cluster the genes into distinct subsets. CLICK identified 15 clusters, out of which 9 contain more than 40 genes, and left 4 outlier genes unclustered (singletons). For each cluster, the size and homogeneity are specified. The overall average homogeneity is 0.8468 and the overall average separation value is -0.0737.

Figure 7.1.5.a. The clustering info dialog that was displayed after running the CLICK algorithm.



7.1.6 Viewing clustered data

A general impression of the clustering results was achieved by using the clustered expression matrix and the clustered similarity matrix visualizations.

Figure 7.1.6.a. shows the clustered matrix display. The order of the six conditions here and in all other displays is: wild type time points 0, 30, 120, ATM^{-/-} time points 0, 30, 120. For example, clusters 1 and 2 contain genes that respond primarily in the mutant, at time points 30 and 120 respectively.

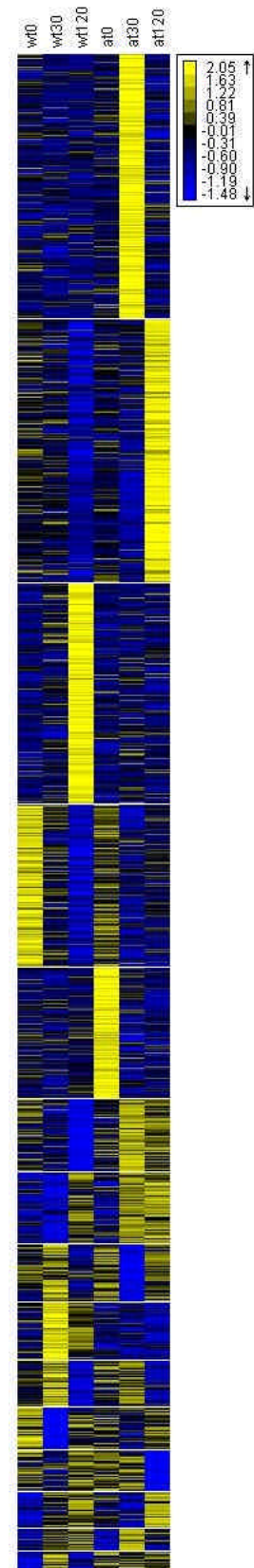


Figure 7.1.6.a. A clustered data matrix visualization of the CLICK clustering solution that was described in section 7.1.5. Patterns of genes that were clustered together appear consecutively. Clusters are separated by white lines.

The clustered similarity matrix display can give a general impression of the similarity/dissimilarity between different clusters and the similarity of genes within the same cluster. In Figure 7.1.6.b. for example, we can see that cluster 1 is very different from clusters 2-5, 8, 9 and 13, and more similar to clusters 6-7, 10-12 and 14.

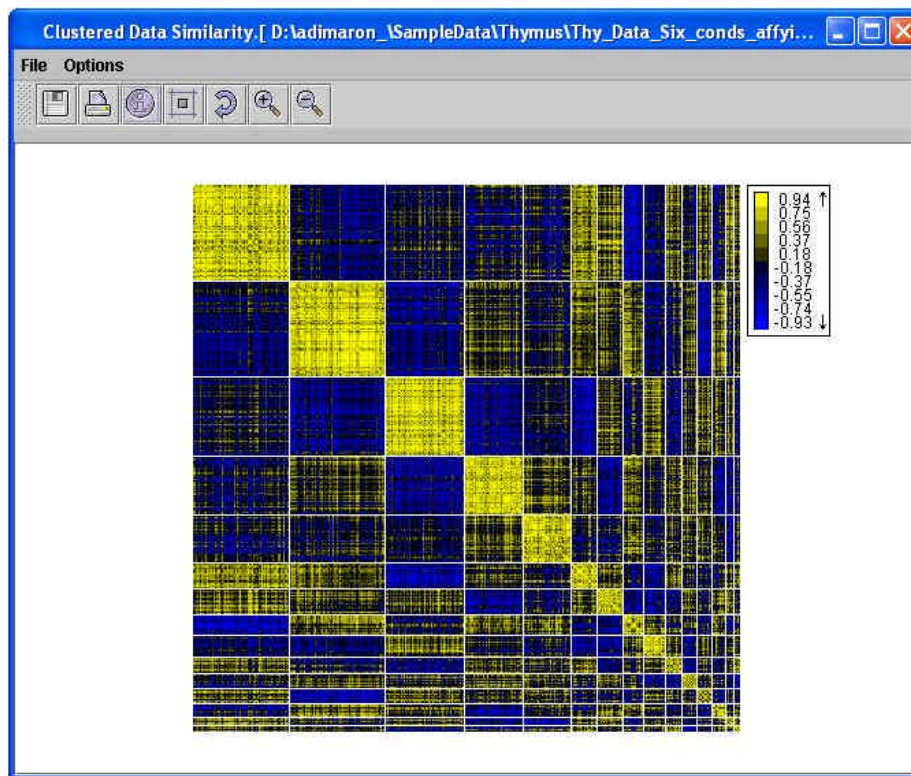


Figure 7.1.6.b. A clustered data similarity matrix display of the CLICK clustering solution. It can be seen that similarity within clusters is much higher than the similarity between genes from different clusters. An impression of the similarity between clusters is also given here.

Cluster patterns were examined by using the 'All clusters mean patterns' visualization, shown in Figure 7.1.6.c. Cluster 1, for example contains 209 genes, which have an expression peak in experimental condition 5 (at30). Biologically, this cluster seems to contain genes that are over-expressed in Atm deficient mice shortly after exposure to IR (30 min.), and then return to basal level.

Cluster 3 contains 175 genes, which have an expression peak in experimental condition 3 (120 min after exposure of wt cells to IR). Biologically, this cluster seems to contain genes that respond more slowly in wild-type mice after exposure to IR ("second wave" response), but do not respond to IR in Atm-deficient mice. These may

be genes that are directly or indirectly under regulatory control of ATM, and therefore the knock-out prevents their upregulation in response to irradiation.

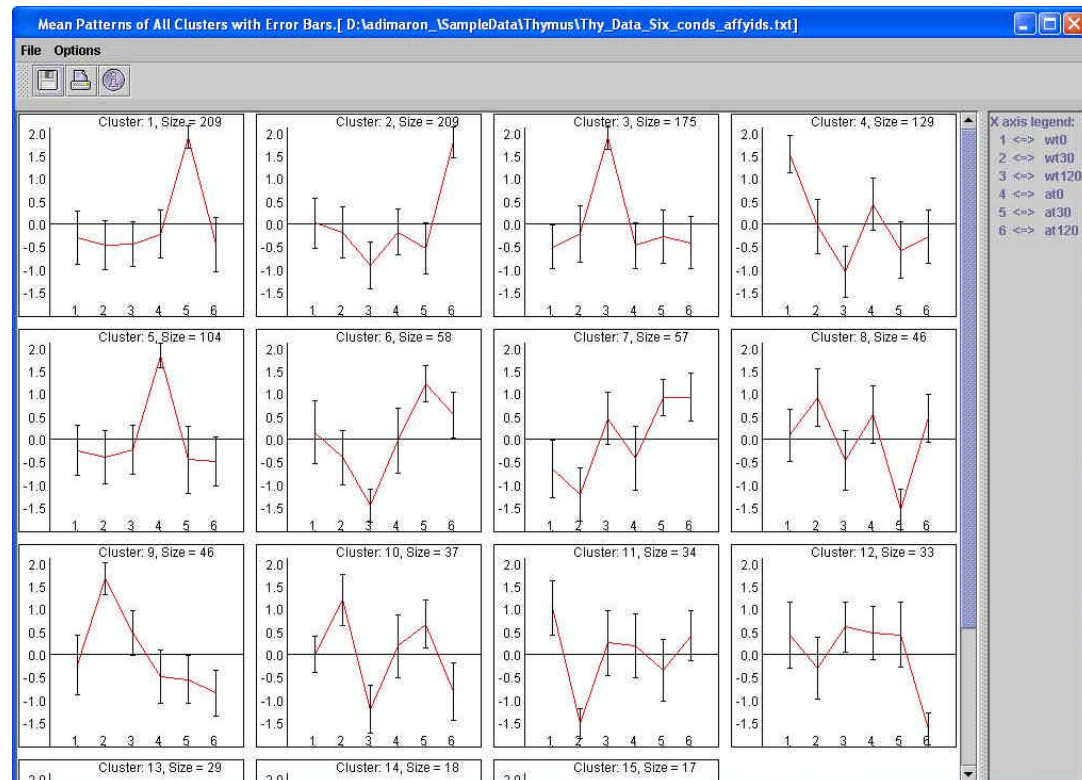


Figure 7.1.6.c. A mean patterns display of the CLICK clusters. Cluster number and size appear at the top of each panel. The X axis contains the condition numbers, and is interpreted in the legend at the right side of the frame. The Y axis contains the expression values. Error bars represent ± 1 standard deviation. In this display patterns seem to be very different from each other, indicating high separation between clusters, while error bars are not very big, indicating high homogeneity within clusters.

Cluster content can be viewed by clicking the relevant panel in the display. This brings up a dialog box, containing all genes that belong to that cluster (fig. 7.1.6.d).

Cluster: 1, Size = 209

Probe ID	Gene Symbol
93018_at	N/A
AFFX-DapX-M_at	N/A
93108_at	P4hb
93118_at	Hnrpa2b1
94065_at	N/A
95482_at	N/A
96085_at	Gsta4
96095_l_at	2610207116Rik
96663_at	Surf6
96685_at	N/A
97199_at	N/A
97800_at	0610011K02Rik
97836_at	Rnf7
97860_at	Nap114
97886_at	Spr
98896_at	N/A
98925_at	Vamp2
94468_at	N/A

OK

Figure 7.1.6.d. A part of the cluster contents list that was displayed upon clicking the mean patterns display of cluster 1.

7.1.7 Performing functional analysis on clusters

In order to characterize the biological processes activated following IR, a functional analysis was performed on the clustered data. First, parameters were set through the settings dialog box (this box is reached from the *Options* menu). Mouse was selected as the examined organism, annotation type was set to 'GO' (using all three type of GO categories: process, function and location), analysis background set was set to 'Original Data' (unfiltered) and threshold p-value set to: 5×10^{-6} (fig. 7.1.7.a).

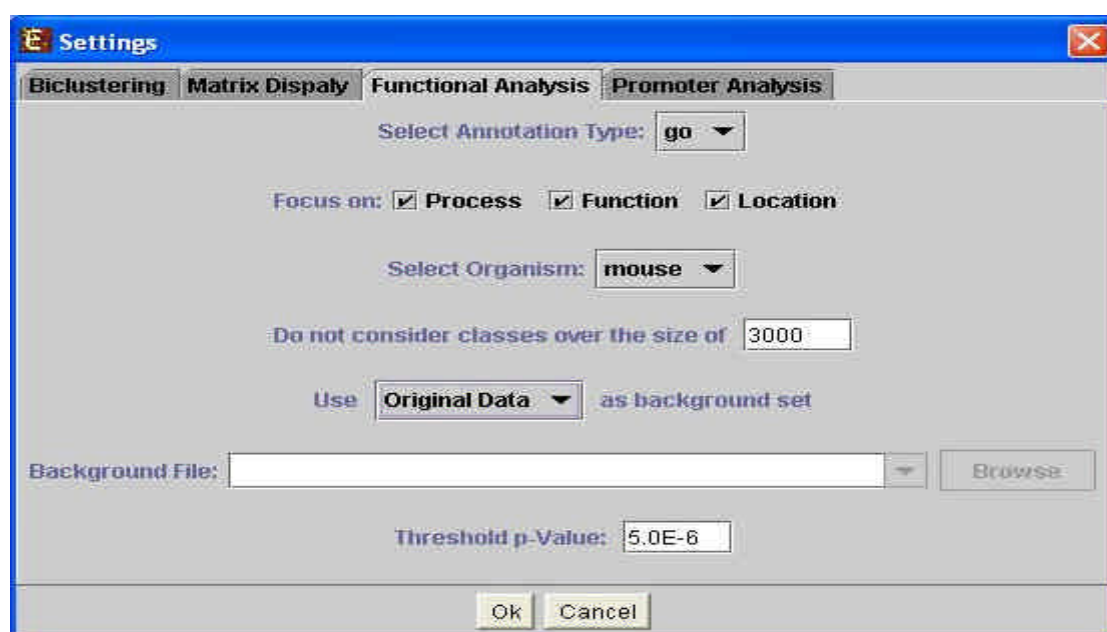


Figure 7.1.7.a. A snapshot of the Functional analysis settings dialog box, as it was configured prior to the functional analysis that is described in this section.

The results of the analysis are shown in Figure 7.1.7.b. The p-values, frequency of classes within clusters and the lists of genes for each column were displayed upon clicking the relevant columns in the histogram (not shown). Cluster 2 was found to be highly enriched for genes of the *immune response* functional class ($p = 7.3 \times 10^{-14}$) and with genes of the *response to pest/pathogen/parasite* functional class ($p = 4.54 \times 10^{-9}$). Since cluster 2 contains genes that exhibit an expression pattern of "second wave" response in $ATM^{-/-}$ mice only, a possible biological explanation for these results is that the absence of a normal response in the cell to irradiation damages causes an inflammatory reaction in which the genes in cluster 2 are highly expressed.

Cluster 5 was found to be highly enriched for genes of the *muscle development* and *muscle contraction* functional classes ($p = 7.78 \times 10^{-14}$ and 1.4×10^{-10} respectively). It was also found to be enriched for genes of the *actine cytoskeleton* and the *cytoskeleton* functional classes ($p = 6.78 \times 10^{-7}$ and 7.03×10^{-7} respectively). We found no good biological explanation for these results.

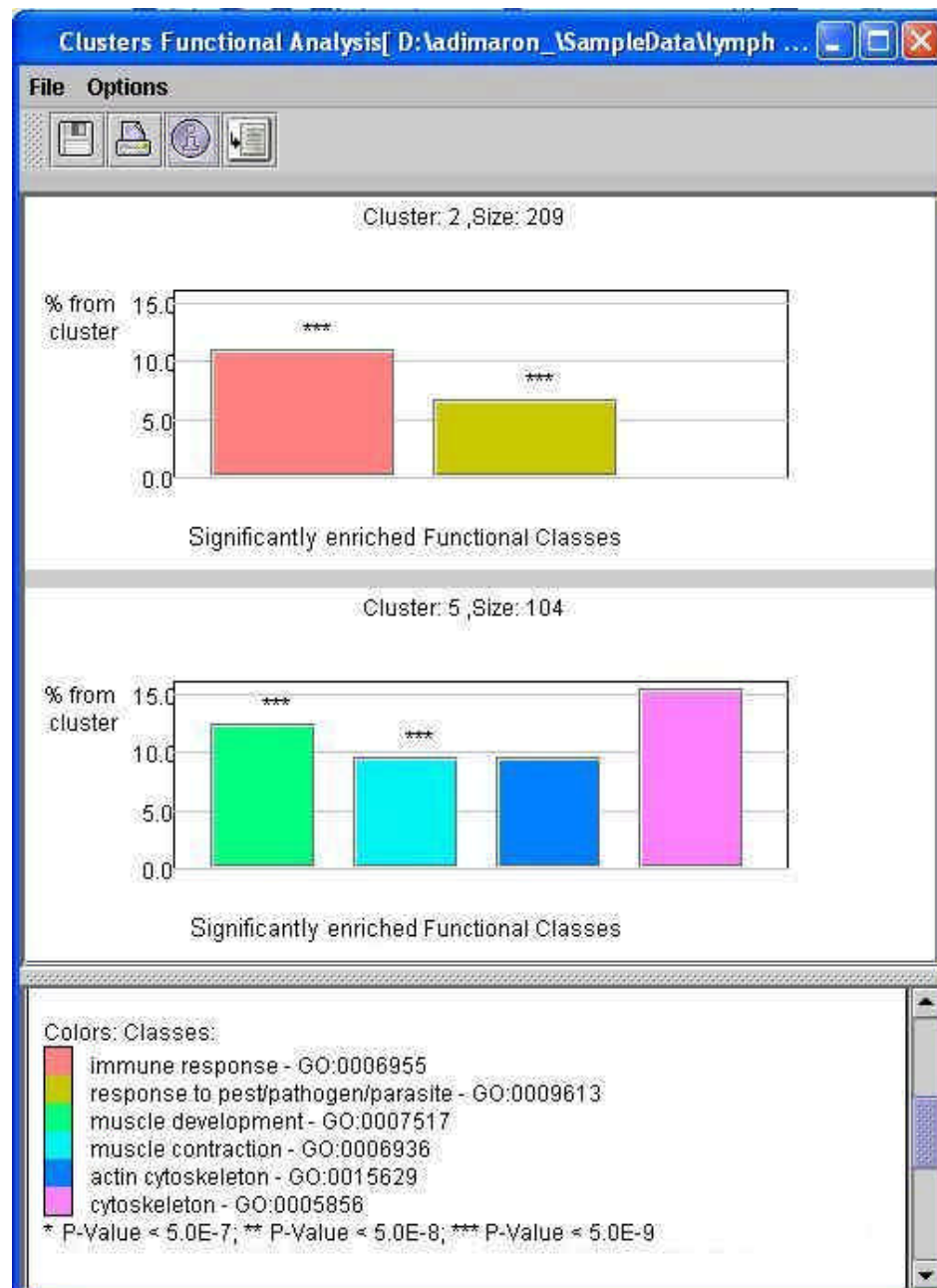


Figure 7.1.7.b. A snapshot of the Functional analysis results display.

7.1.8 Performing promoter analysis on clusters

In order to reveal regulators whose activation is compromised in Atm-deficient tissues, promoter analysis was performed, assuming that genes that exhibit similar transcriptional expression patterns across multiple conditions will share cis-regulatory elements in their promoters.

Parameters were set through the *settings* dialog. 'mouse' was selected as the examined organism, fingerprint file was selected, the background set was set to "Original Data" (unfiltered), and the threshold p-value was set to 0.05 with FDR Bonferroni correction (fig. 7.1.8.a).

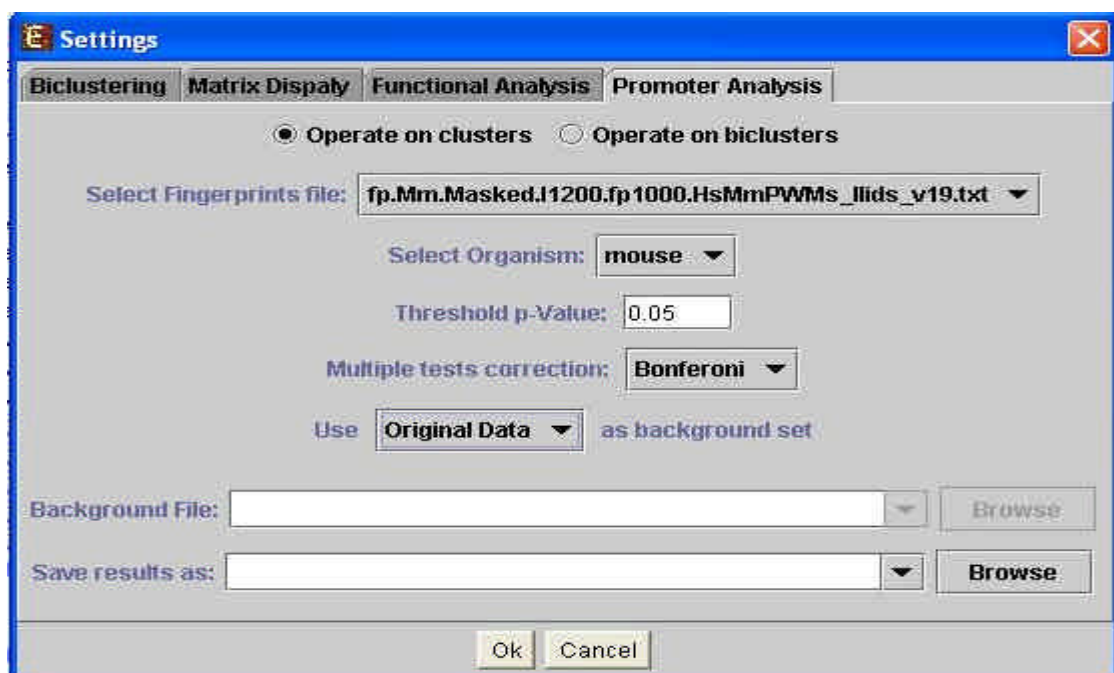


Figure 7.1.8.a. A snapshot of the promoter analysis settings dialog box, as it was configured prior to the analysis that is described in this section.

The results of the analysis are shown in Figure 7.1.8.b.

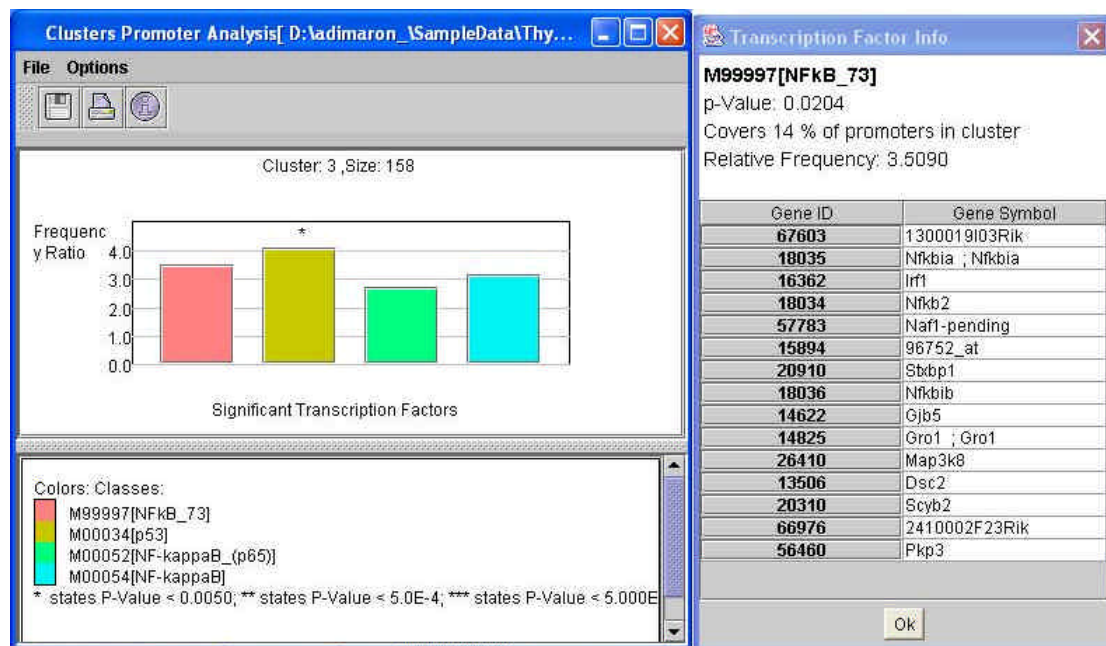


Figure 7.1.8.b. A snapshot of the resulting display of promoter analysis performed on the clusters described in section 7.1.5. p-values, relative frequencies and gene lists were displayed by clicking on the relative columns.

Cluster 3 was found to be highly enriched with promoters that contain binding sites for NF-kappaB (NF?B_73 is another PWM for NF-kappaB), NF-kappaB_(p65) (which is a subunit of NF-kappaB) and p53. The display shows that the incidence of the p53 binding profile is 4-fold higher among the promoters of cluster 3 than in the background set ($p = 0.0041$), and the incidence of the NF-kappaB binding profile is 3-fold higher among the promoters of cluster 3 than in the background set ($p = 0.0204$).

The results suggest that genes in cluster 3 might be regulated by one or more of these transcription factors, which are well established stress-induced transcriptional regulators (Amudson *et al.* 2003). These results support previous studies that reported compromised IR-induced activation of both NF-kappaB and p53 in Atm-deficient tissues and in cell lines derived from A-T patients (Banin *et al.*, 1998; Li *et al.*, 2001c; Piret *et al.*, 1999 ; Saito *et al.*, 2002).

7.2 Example 2: Analysis of yeast cDNA microarray data concerning responses to environmental changes

7.2.1 The data

In this example we analyze a published dataset dealing with yeast stress responses. DNA microarrays were used to measure changes in transcript levels over time for the yeast genes, in response to a variety of stress conditions. These include temperature shocks, hydrogen peroxide, menadione (a superoxide-generating drug), diamide (a sulfhydryl-oxidizing agent), dithiothreitol (a disulfide-reducing agent), hyper-osmotic shock, amino acid starvation, nitrogen source depletion and progression into stationery phase. The expression levels were also measured under several environmental change conditions that are not considered stressful, such as temperature change from 37° to 25° and hypo-osmotic shock. The dataset contains gene expression measurements for all 6153 putative yeast genes in 15 different time series under various environmental conditions, generating a set of 173 expression profiles (*Gasch et al. 2000*).

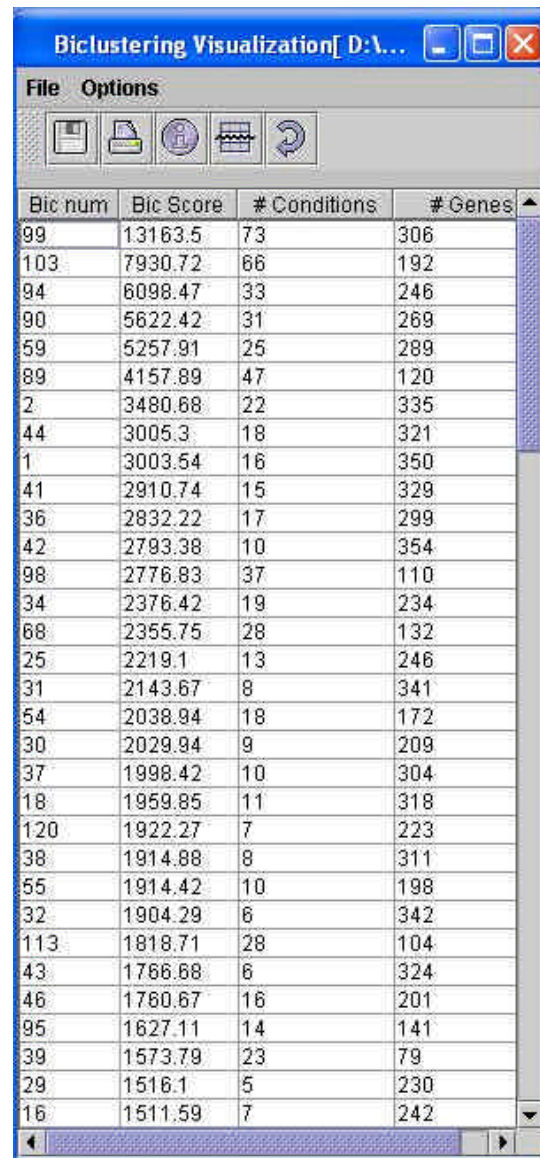
7.2.2 Loading the data

Data was loaded without a conversion file since gene IDs in the input file match the IDs in fingerprint and annotation files supplied by EXPANDER.

7.2.3 Biclustering the data

Biclustering was performed on the entire dataset using default parameters. 124 biclusters were detected. A part of the results is shown in Figure 7.2.3.a.

Note the variability in the dimensions of the biclusters, with the number of conditions varying between 5 and 73 and the number of genes varying between 79 and 350.



The screenshot shows a window titled "Biclustering Visualization[D:\...". It has a menu bar with "File" and "Options", and a toolbar with icons for file operations. Below the toolbar is a table with four columns: "Bic num", "Bic Score", "# Conditions", and "# Genes". The table contains 30 rows of data, representing the top-scoring biclusters. The scores are sorted in descending order.

Bic num	Bic Score	# Conditions	# Genes
99	13163.5	73	306
103	7930.72	66	192
94	6098.47	33	246
90	5622.42	31	269
59	5257.91	25	289
89	4157.89	47	120
2	3480.68	22	335
44	3005.3	18	321
1	3003.54	16	350
41	2910.74	15	329
36	2832.22	17	299
42	2793.38	10	354
98	2776.83	37	110
34	2376.42	19	234
68	2355.75	28	132
25	2219.1	13	246
31	2143.67	8	341
54	2038.94	18	172
30	2029.94	9	209
37	1998.42	10	304
18	1959.85	11	318
120	1922.27	7	223
38	1914.88	8	311
55	1914.42	10	198
32	1904.29	6	342
113	1818.71	28	104
43	1766.68	6	324
46	1760.67	16	201
95	1627.11	14	141
39	1573.79	23	79
29	1516.1	5	230
16	1511.59	7	242

Figure 7.2.3.a: A part of the biclusters table. Biclusters are sorted according to their score, and only the top scoring biclusters are displayed.

7.2.4 Performing functional analysis on biclusters

Functional analysis was performed on the biclusters. The whole dataset was used as background set, and only "Process" annotations were tested. Threshold p-value was set to 5×10^{-4} (Figure 7.2.6.a).

Figure 7.2.4.a: Functional analysis settings used to analyze biclustering results.

The resulting biclusters table contained also information on the significant functional classes in each bicluster. Part of the table is shown in Figure 7.2.6.b. 53 out of all 124 biclusters scored higher than 1000. Over half of the biclusters consist of more than 100 genes, and almost all of them consist of less than 40 conditions.

Bic num	Bic Score	# Conditions	# Genes	Significant Gene Annotation:				Significant Gene Annotation:			
				Annotation1	pValue1	%	# of annotate	Annotation2	pValue2	% A	# of annotate
99	13163.5	73	306	ribosome - GO:0005840	0.0	45	138	cytosolic rib...	0.0	41	128
103	7930.72	66	192	nucleolus - GO:0005730	0.0	53	103	ribosome bi...	3.178E-42	50	97
94	6098.47	33	246	cytosolic large ribosomal subunit (se...	0.0	31	77	ribosome - ...	0.0	56	139
90	5622.42	31	269	ribosome - GO:0005840	0.0	49	132	cytosolic rib...	0.0	45	123
59	5257.91	25	289	cytosolic ribosome (sensu Eukarya) - ...	2.246E-38	31	92	ribonucleop...	2.699E-35	44	128
89	4157.89	47	120	nucleolus - GO:0005730	1.018E-31	55	67	ribosome bi...	5.517E-30	55	67
2	3480.68	22	335	carbohydrate metabolism - GO:0005...	8.045E-7	12	42	protein fold...	1.852E-4	4	16
44	3005.3	18	321	carbohydrate metabolism - GO:0005...	4.039E-7	13	42	protein fold...	4.811E-5	5	17
1	3003.54	16	350	carbohydrate metabolism - GO:0005...	3.191E-6	11	41	protein fold...	2.946E-5	5	18
41	2910.74	15	329	oxidative phosphorylation - GO:00061...	6.001E-13	7	24	energy path...	2.388E-11	16	53
36	2832.22	17	299	energy derivation by oxidation of orga...	2.146E-11	17	51	energy path...	2.146E-11	17	51
42	2793.38	10	354	energy derivation by oxidation of orga...	3.662E-5	10	37	energy path...	3.662E-5	10	37
98	2776.83	37	110	nucleolus - GO:0005730	2.223E-19	43	48	ribosome bi...	7.306E-19	44	49
34	2376.42	19	234	oxidative phosphorylation - GO:00061...	1.212E-10	8	20	mitochondri...	4.423E-10	7	18
68	2355.75	28	132	cytosolic ribosome (sensu Eukarya) - ...	2.647E-32	49	65	ribosome - ...	1.516E-28	53	71
25	2219.1	13	246	nucleolus - GO:0005730	3.657E-39	39	97	ribosome bi...	2.79E-33	34	86
31	2143.67	8	341								
54	2038.94	18	172	protein folding - GO:0006457	2.829E-5	8	14	response to...	2.805E-4	19	33
30	2029.94	9	209	amine metabolism - GO:0009308	1.583E-22	29	62	amino acid ...	2.847E-21	27	57
37	1998.42	10	304	energy derivation by oxidation of orga...	7.398E-9	14	45	energy path...	7.398E-9	14	45
18	1959.85	11	318	protein folding - GO:0006457	1.505E-5	5	18	carbohydrat...	5.872E-5	11	35
120	1922.27	7	223								
38	1914.88	8	311	energy derivation by oxidation of orga...	1.304E-5	11	36	energy path...	1.304E-5	11	36
55	1914.42	10	198	nucleolus - GO:0005730	4.253E-36	43	86	ribosome bi...	4.561E-33	42	85
32	1904.29	6	342								
113	1818.71	28	104	carbohydrate metabolism - GO:0005...	3.954E-4	17	18				
43	1766.68	6	324	energy derivation by oxidation of orga...	2.247E-5	11	36	energy path...	2.247E-5	11	36
46	1760.67	16	201	oxygen and reactive oxygen species ...	1.282E-5	7	15	regulation o...	3.092E-4	3	7
95	1627.11	14	141								
39	1573.79	23	79	cytosolic ribosome (sensu Eukarya) - ...	2.876E-15	43	34	ribosome - ...	2.877E-15	50	40
29	1516.1	5	230	amine metabolism - GO:0009308	7.065E-23	27	64	amino acid ...	1.009E-21	25	59

Figure 7.2.4.b. Part the biclusters table that contains also significantly enriched functional classes. Four columns are used to describe each enriched functional class. These include annotation name, p-value, percentage in bicluster and the number of genes in the cluster that have this annotation. Scroll bars facilitate browsing through the data.

7.2.5 Viewing biclusters and significant functional classes

Biclusters with high scores are viewed by clicking on the corresponding rows in the table. For each bicluster, an expression matrix is displayed. For each significant functional class, a column is added to the matrix display, indicating for each gene, whether or not it belongs to that class.

Bicluster #99 has the highest score (13,163.5), and it consists of 306 genes and 73 conditions. It was found to be enriched with the following functional classes:

Class Name:	Go ID:	p-value:
ribosome	GO:0005840	Under 10^{-45}
cytosolic ribosome	GO:0005830	Under 10^{-45}
Ribonucleoprotein complex	GO:0030529	Under 10^{-45}
cytosolic large ribosomal subunit (sensu Eukarya)	GO:0005842	5.12×10^{-42}
cytosol	GO:0005829	3.43×10^{-39}

Figure 7.2.7.a shows a part of the expression matrix of bicluster #99. According to the expression matrix, the genes in this bicluster are suppressed under most of the tested stress conditions, but are not suppressed under environmental conditions that are not considered stressful (i.e., hypo-osmotic shock in columns 8,9 and temperature change from 37° to 25° in columns 3, 4). A cluster with a similar expression profile and enriched functional classes was detected in the research that was previously performed on this data by Gasch et al., where different methods were used for analysis (*Gasch et al. 2000*). These results support previous observations of repression of ribosomal protein genes during multiple stress responses (*Warner 1999*; *Sakaki et al. 2003*). These results support the conclusion, presented by Gasch et al., that suppression of genes involved in protein synthesis is a general feature of the "environmental stress response" (ESR) (*Gasch et al. 2000*).

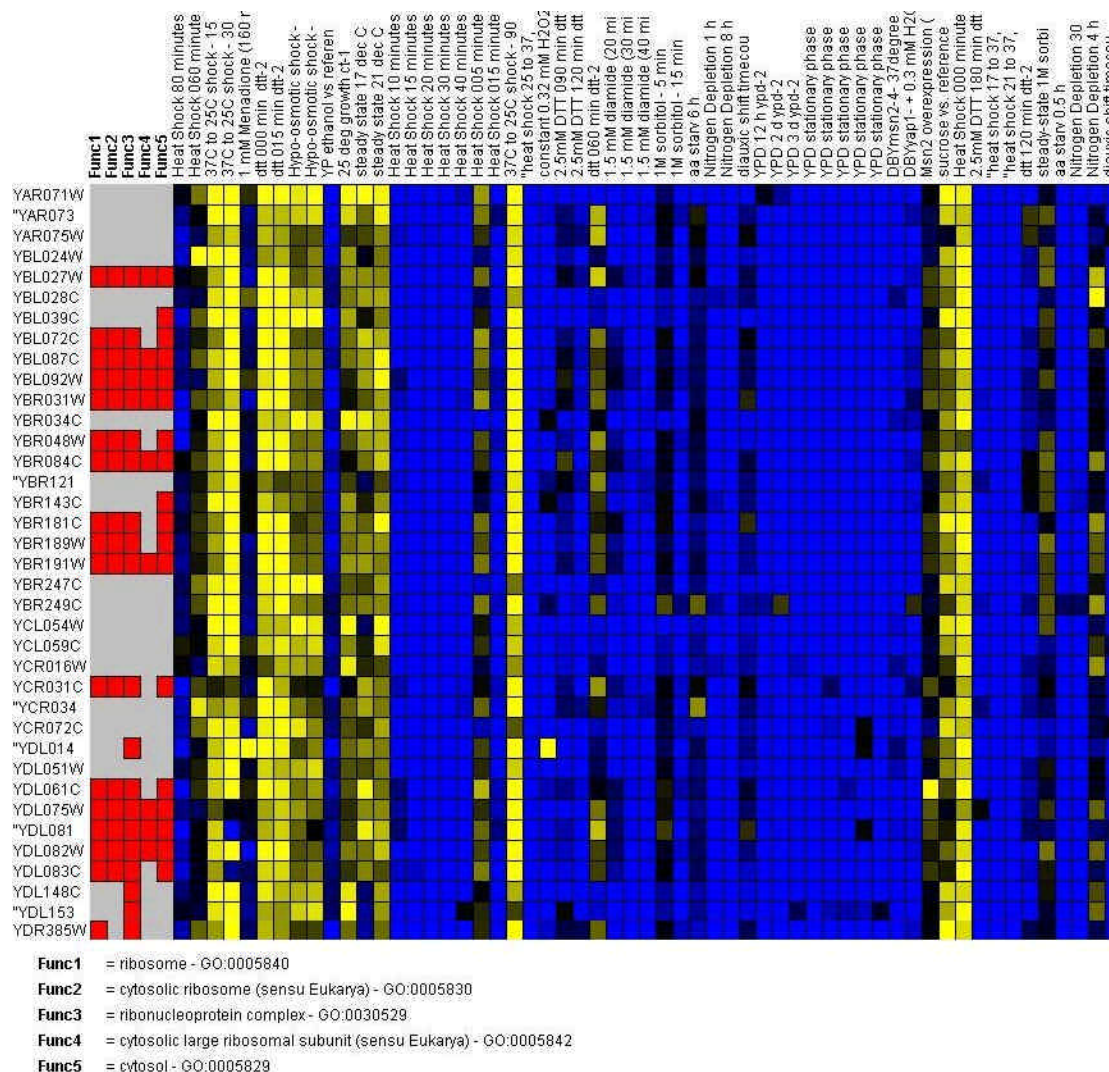


Figure 7.2.7.a: Part of the expression matrix of bicluster #99. Yellow indicates over-expression and blue indicates under expression.

Bicluster #103 consists of 192 genes and 66 conditions. It was found to be enriched with the following functional classes:

Class Name:	GO ID:	p-value:
nucleolus	GO:0005730	Under 10^{-45}
Ribosome biogenesis and assembly	GO:0042254	$3.18 \cdot 10^{-42}$
Ribosome biogenesis	GO:0007046	$4.41 \cdot 10^{-42}$
Transcription from Pol1 promoter	GO:0006360	$2.75 \cdot 10^{-33}$
RNA processing	GO:0006396	$6.27 \cdot 10^{-26}$

Figure 7.2.7.b shows a part of the expression matrix of bicluster #99. According to the expression matrix, the genes in this bicluster are suppressed under stress conditions, but are not suppressed under environmental conditions that are not considered

stressful (e.g., hypo-osmotic shock and temperature change from 37° to 25°). A cluster with a similar expression profile and enriched functional classes was detected by Gasch et al., using the TreeView software. This cluster was produced by hierarchically clustering the whole dataset (i.e., using all conditions), and it consists almost entirely of genes encoding ribosomal proteins (*Gasch et al. 2000*). The IDs of the genes in the cluster and its exact size were not published, so a direct comparison of the gene sets is, unfortunately, impossible.

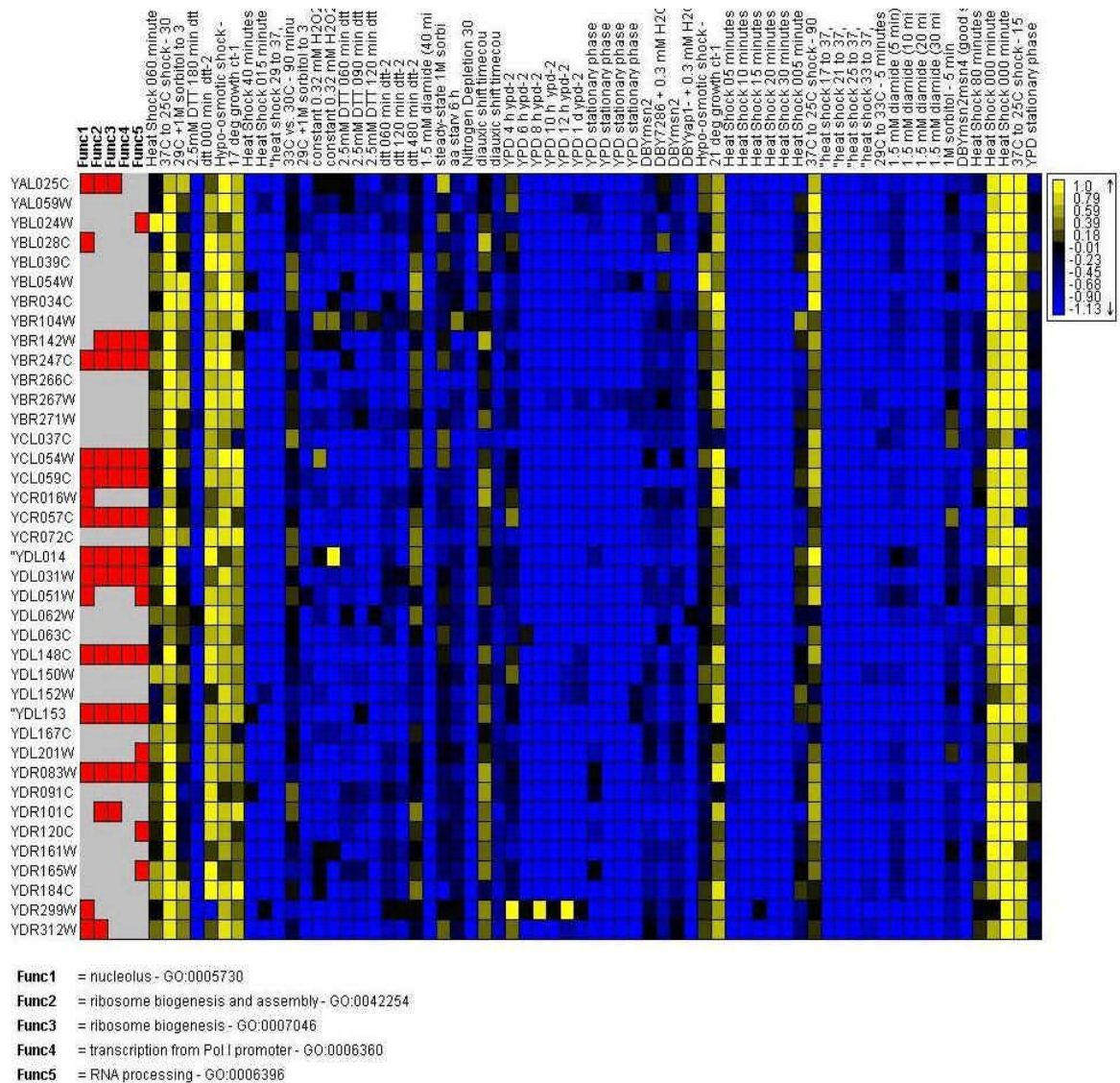


Figure 7.2.7.b: Part of the expression matrix of bicluster #103. Yellow indicates over-expression and blue indicates under expression.

Bicluster #2 consists of 335 genes and 21 conditions. It was found to be enriched with the following functional classes:

Class Name:	Go ID:	p-value:
carbohydrate metabolism	GO:0005975	8.045×10^{-7}
Protein folding	GO:0006457	1.85×10^{-4}
Energy pathways	GO:0006091	2.39×10^{-4}
Response to stress	GO:0006950	3.387×10^{-4}

Figure 7.2.7.c shows the expression matrix of bicluster #2. According to the expression matrix, the genes in this bicluster are induced under various kinds of stress conditions (Diamide exposure, DTT exposure, heat shock and hyper-osmotic shock), and are not induced under environmental conditions that are not considered stressful (e.g., temperature change from 37° to 25° in columns 14 and 21). The first condition in the matrix ("heat shock 005 minutes hs-2") seems to show repression under stress, in conflict with the rest of the results. We suspect that this experiment is faulty, since (1) it contradicts results of a repeated experiment under the same condition ("heat shock 05 hs-1", column 15) and (2) it also appears to differ from the other heat shock conditions also in biclusters 103 and 99.

These results are consistent with the results reported by Gasch et al., describing a set of approximately 300 genes that were induced in ESR. This set was reported to consist of genes that are involved in a wide variety of processes, including carbohydrate metabolism and protein folding (*Gasch et al. 2000*). The IDs of the genes in this set were not published.

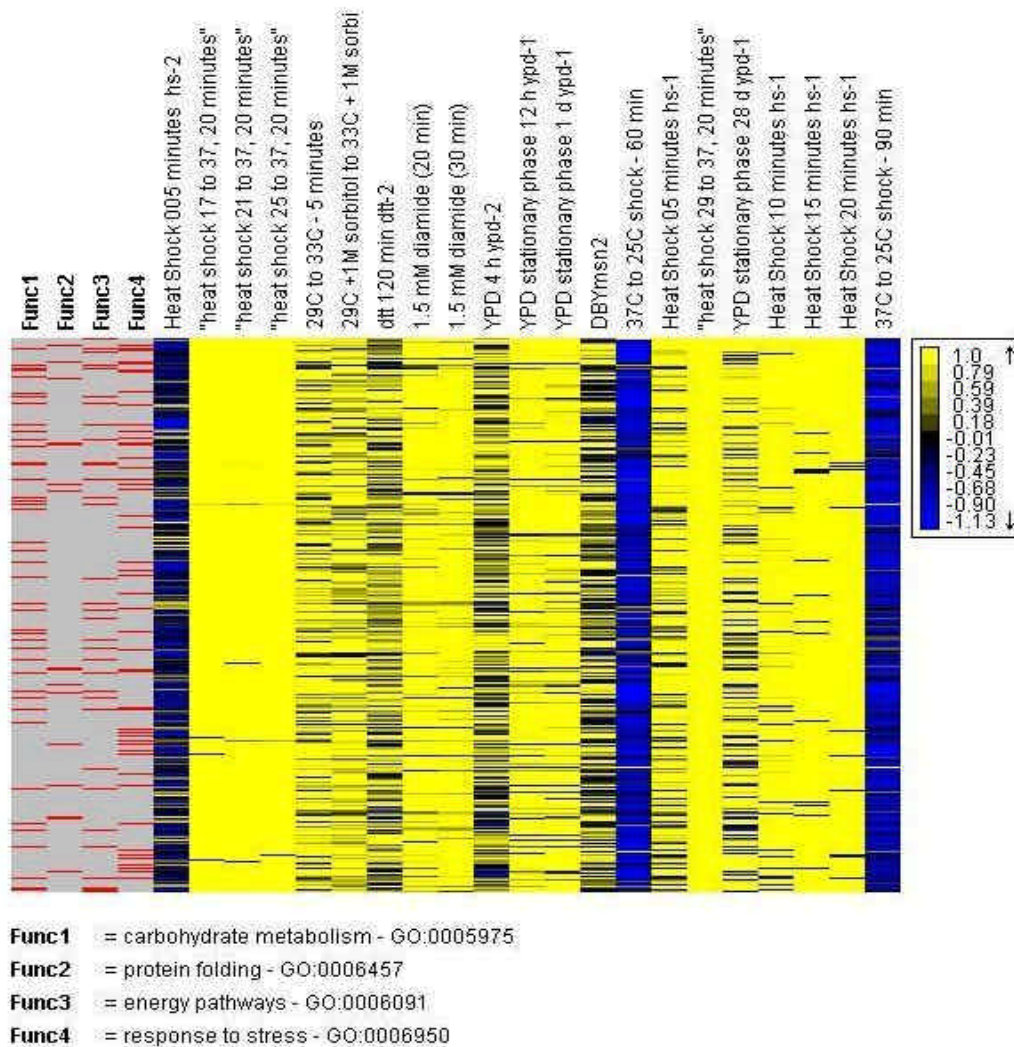


Figure 7.2.7.c: The expression matrix of bicluster #2. Yellow indicates over-expression and blue indicates under expression.

Bicluster #30 consists of 209 genes and 9 conditions (all amino-acid starvation conditions along with the first three nitrogen depletion conditions). It was found to be enriched with the following functional classes:

Class Name:	Go ID:	p-value:
Amine metabolism	GO:0009308	1.58×10^{-22}
Amino acid metabolism	GO:0006520	2.85×10^{-21}
Glutamine family amino acid biosynthesis	GO:0009084	3.51×10^{-10}
Nitrogen metabolism	GO:0006807	1.38×10^{-8}
Sulfur metabolism	GO:0006790	3.85×10^{-8}

Figure 7.2.7.d shows the expression matrix of bicluster #30. According to the expression matrix, the genes in this bicluster are induced under amino acid starvation or nitrogen depletion conditions, and suppressed in steady state with sorbitol.

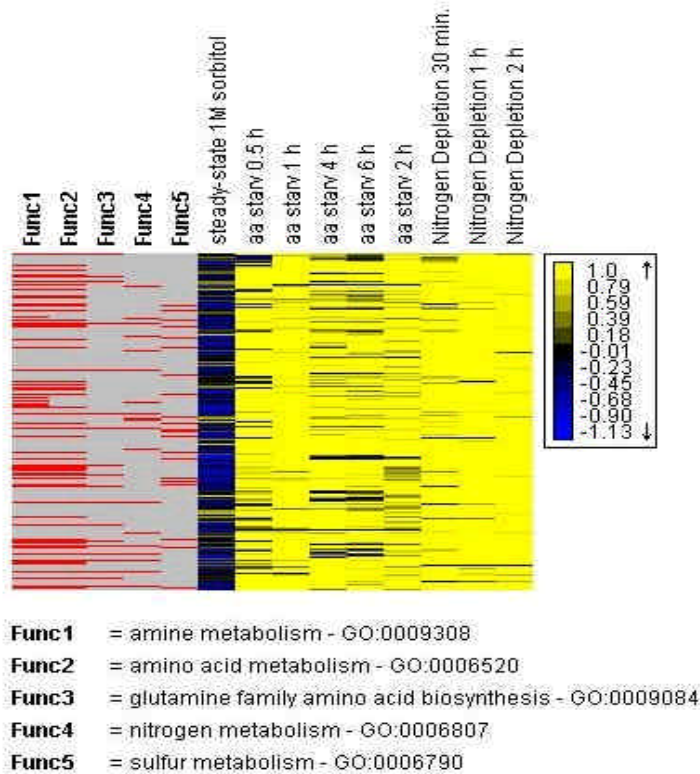


Figure 7.2.7.d: The expression matrix of bicluster #30. Yellow indicates over-expression and blue indicates under expression.

7.2.6 Performing Promoter analysis on biclusters

Promoter analysis was performed on the biclusters. The whole dataset was used as background set and threshold p-value was set to 5×10^{-8} without a correction for multiple tests (Figure 7.2.8.a).

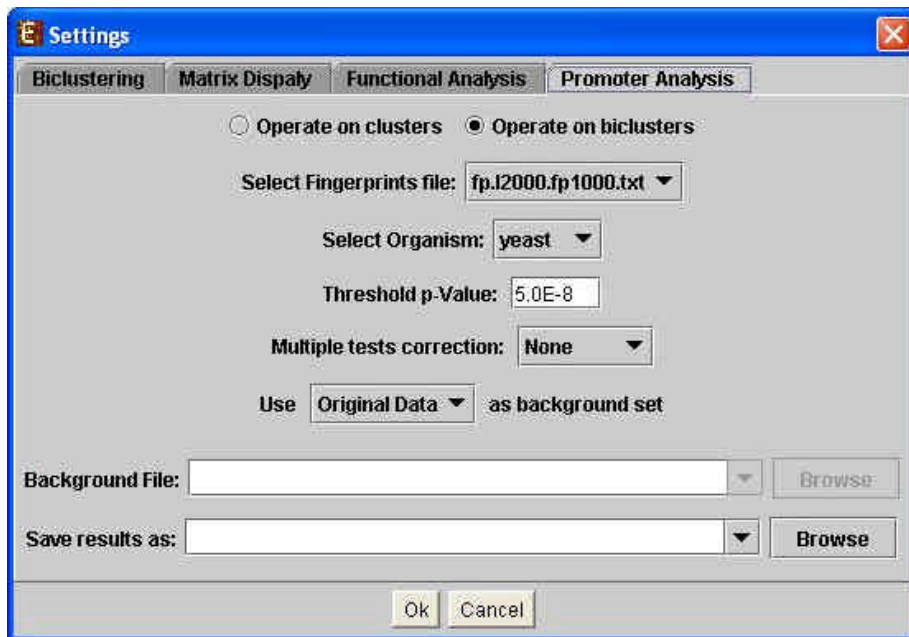


Figure 7.2.8.a: Promoter analysis settings that were used to analyze biclustering results.

43 PWMs corresponding to yeast TF motifs, extracted from the 'Transfac' database (version 7.4, April 2004), were used for this analysis. The resulting display contained a histogram for each bicluster which was found to be enriched with at least one TF binding site.

Thirty four biclusters were found to be significantly enriched with at least one TF binding site.

Bicluster #99 was found to be highly enriched with RAP1 motif, with a p-value = 1.83×10^{-19} (Figure 7.2.8.b). Recall that our functional analysis identified this bicluster as related to ribosomal proteins. These results are in correlation with previous studies, demonstrating that the repression of ribosomal protein genes is regulated by the transcription factor Rap1p (Moehle and Hinnebusch 1991; Li et al. 1999).

Bicluster #103 was not found to be enriched with any of the tested motifs.

Bicluster #2 was found to be enriched with STRE (Stress Response Element) motif, with a p-value = 1.74×10^{-18} (Figure 7.2.8.c). These results are in agreement with previous studies that have identified STRE sequences in many stress-induced genes (Kandror et al. 2004; Boorsma et al. 2004). Two transcription factors, Msn2p and Msn4p, are involved in STRE-mediated gene expression (Martinez-Pastor et al., 1996). Both factors bind to STRE in vitro and in vivo and are required for the

induction of an *STRE-LEU2-lacZ* reporter gene in response to different forms of stress (Martinez-Pastor *et al.*, 1996).

Bicluster #30 was found to be enriched with GCN4, CBF1 and AP-1 motifs, p-values are shown in Figure (Figure 7.2.8.d). GCN4 is a transcription factor that is known to play a key role in the regulation of amino acid metabolism in yeasts (Hinnebusch 1984). It has been shown to bind degenerate variants of the pseudo palindrome 5'ATGACTCAT3' known as the AP-1 site (Suckow *et al.* 1994) (i.e., the motifs identified as AP-1 binding sites in this bicluster are probably the same motifs identified as CGN4 binding sites). CBF1 is a transcription factor that is necessary for the expression of genes involved in methionine biosynthesis, and deletion of *CBF1* renders *S. Cerevisiae* methionine auxotrophic (Kuras and Thomas 1995). These results support the suggestion presented by Gacsh *et al.* that ESR regulation is both gene specific and condition specific, and that the expression of genes in ESR is regulated by different transcription factors depending on the conditions (Gasch *et al.* 2000).

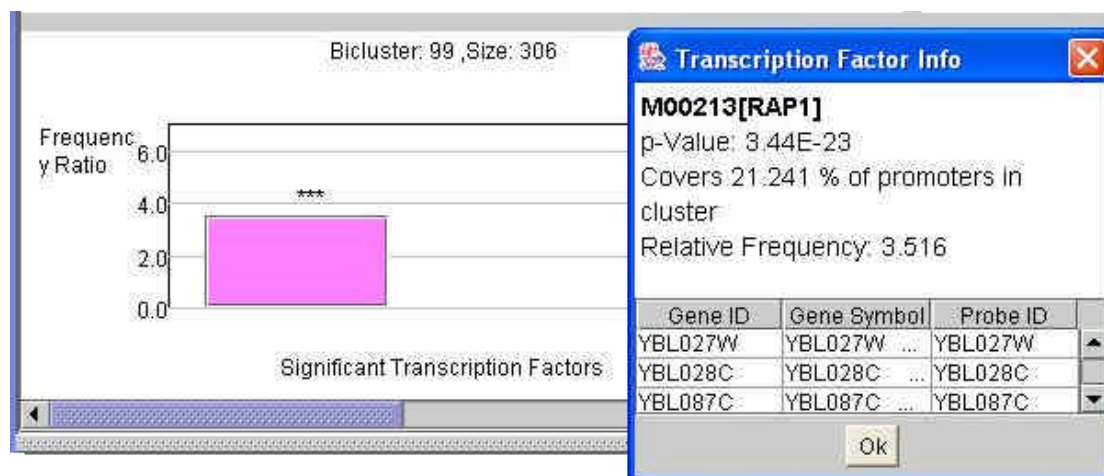


Figure 7.2.8.b. Promoter analysis results for bicluster #99. Upon clicking a column in the histogram, an info dialog box appears.

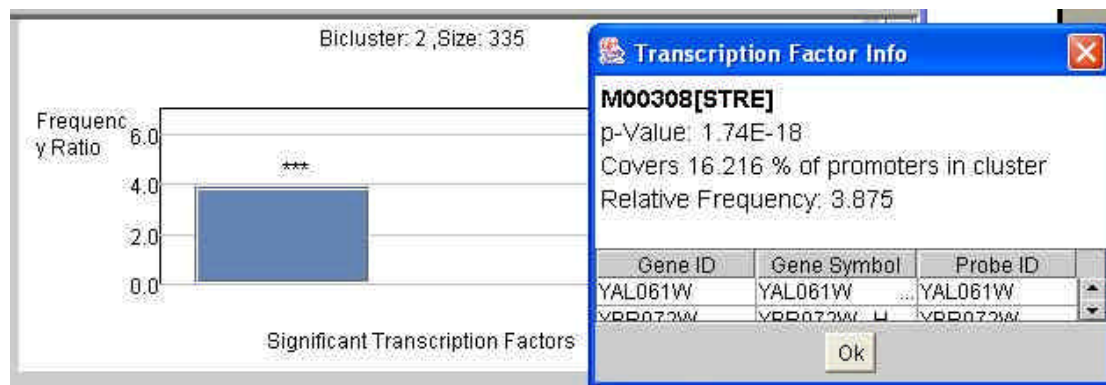


Figure 7.2.8.c. Promoter analysis results for bicluster #2.

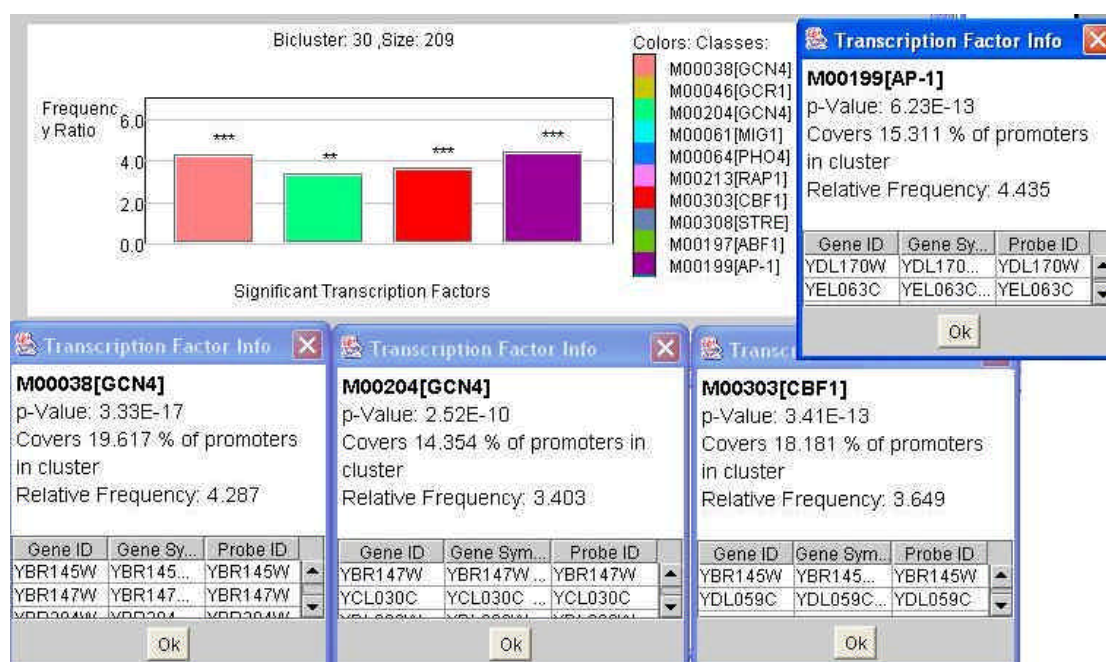


Figure 7.2.8.d. Promoter analysis results for bicluster #30.

7.2.7 Discussion

The genomic expression programs characterized in the study of Gasch et al. and in this analysis reveal that yeast cells respond to environmental changes by altering the expression of thousands of genes, creating a genomic expression program that is customized for each environment (Gasch et al. 2002).

Our biclustering analysis detected several biclusters which were found to be highly enriched for genes that encode ribosomal proteins. As would be expected, bicluster expression matrices indicate down regulation of these genes under stressful conditions and upregulation when environmental conditions improve (e.g. 37° to 25°, hypo

osmotic shock etc.). One such bicluster (#99) was found to be significantly enriched for genes that contain a RAP1 binding site in their promoters. It has been shown that the majority of ribosomal protein (RP) genes and a number of the translation factors genes contain binding sites for the essential Rap1 protein in their upstream regions (*Shore 1994*).

Another type of biclusters that were detected in our analysis was highly enriched for genes that are involved in ribosome biogenesis and assembly and transcription from *pol1* promoter. As in the biclusters described above, expression matrices indicate down regulation of these genes under stressful conditions and upregulation under normal environmental conditions.

Our biclustering algorithm also detected a bicluster of genes that are highly expressed under amino acid starvation and nitrogen depletion conditions. This bicluster was found to be highly enriched for genes that are involved in amine and amino acids metabolism and biosynthesis. It was found to be significantly enriched for genes that contain binding sites for GCN4, CBF1 and Ap1 in their promoters. GCN4 is a known transcriptional activator of amino acid biosynthetic genes (*Sattlegger et al. 2004*) and CBF1 is a transcription factor that is necessary for the expression of genes involved in methionine biosynthesis (*Kuras and Thomas 1995*). The reason for the enrichment in Ap1 binding sites is not clear. One possible reason is that GCN4 binds degenerate variants of the pseudo palindrome 5'ATGACTCAT3' known as the AP-1 site (*Suckow et al. 1994*).

Our analysis recovered the key conclusions reported by Gasch et al. (2000). A key difference in our methodology is the elimination of the need to use prior biological knowledge and of the subjective pre-selection of conditions for the analysis. All biclusters and their condition sets (including biclusters containing a small set of conditions, such as bicluster #30), were automatically detected by the SAMBA biclustering algorithm from the entire dataset.

7.3 Example 3: Analysis of cDNA microarray data associated with cell cycle progression in human cells

7.3.1 The data

cDNA microarrays were used to measure gene expression in human cancer cell line, HeLa cells (*Whitfield et al. 2002*). Prior to the microarray preparation cell cultures were synchronized in three different ways: a double thymidine block, a thymidine – nocodazole block and a mitotic shake off (a physical method) (*Whitfield et al. 2002*). Altogether 114 arrays were prepared for different cell cycle stages and using different synchronization methods. Whitfield et al. identified 874 genes (represented by 1134 elements or probes) as periodically expressed during the cell. Our analysis focused on this set only.

7.3.2 Loading the data

The raw data were edited to fit the required EXPANDER input format and contents (some of the columns from the original data file were excluded and array names were changed to be unique). The edited dataset was loaded, along with a conversion file that converted UUIDs to LocusLink IDs. Missing values were automatically set to 0, since in cDNA microarrays values are expected by EXPANDER to be given as log ratios so the value 0 indicates a normal expression level (ratio = 1).

7.3.3 Preprocessing the data

Data were filtered using an input file containing 1134 IDs of elements representing the 874 genes that were reported as periodically expressed (this file was downloaded from <http://genome-www.stanford.edu/Human-CellCycle/Hela/index.shtml>). Since we used an external clustering solution generated by Whitfield et al. (see below), no additional filtering was performed.

7.3.4 Loading a clustering solution

Whitfield et al. (2002) partitioned the cell cycle regulated genes according to their expression periodicity patterns into five clusters, corresponding to cell cycle phases G1/S(1), S(2), G2(3), G2/M(4), and M/G1(5). This partition was loaded as a clustering solution.

Figure 7.3.4.a shows the clustering results dialog that also reports the homogeneity and separation of the clustering. These values were calculated by EXPANDER according to the expression patterns.

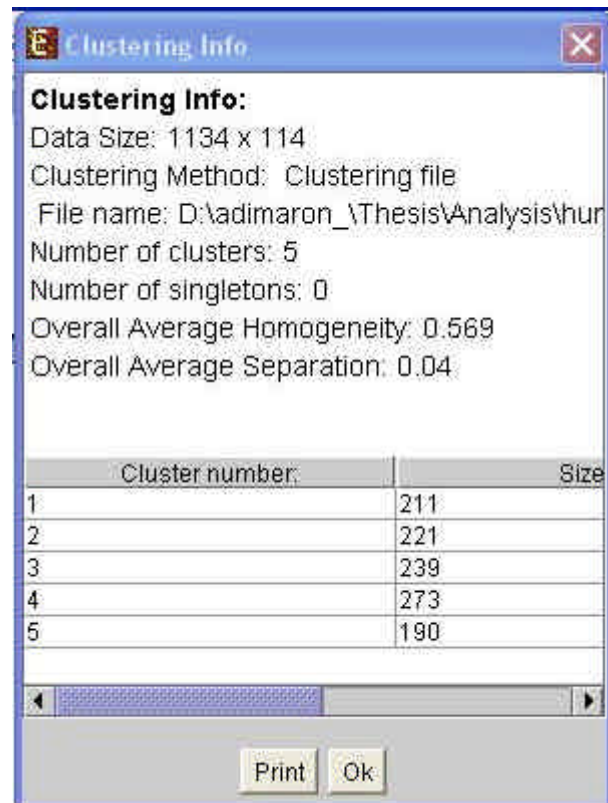


Figure 7.3.4.a. Clustering results dialog produced by EXPANDER after loading the gene partition into clusters as provided by Whitfield et al.

7.3.5 Viewing clustered data

Figure 7.3.5.a shows the clustered expression matrix display. Patterns of genes which were clustered together appear next to each other.

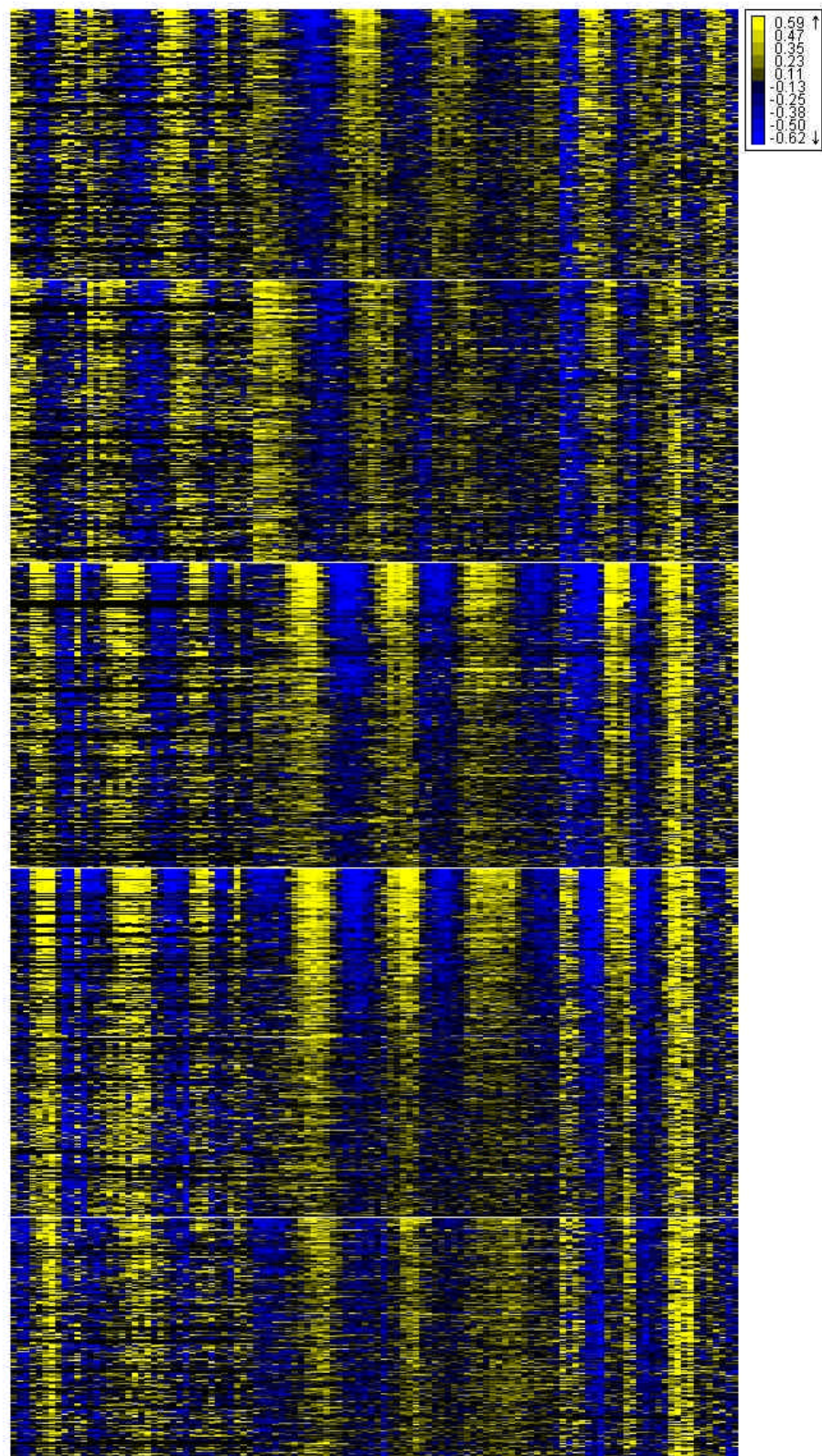


Figure 7.3.5.a. A clustered expression matrix display of the data. The display contains 1134 elements representing 874 genes, which were identified as periodically expressed. Clusters are separated by white lines. The clustering solution was provided by Whitfield et al.

Figure 7.3.5.b shows a PCA visualization of the data. Each element is represented as a point on an XY scatter plot. Elements from the same cluster appear in the same color (Note that different probes of the same gene are displayed separately). In this Figure we can see that the similarities and distances between patterns (vectors) in the data are preserved reasonably well in the projection to two dimensions.

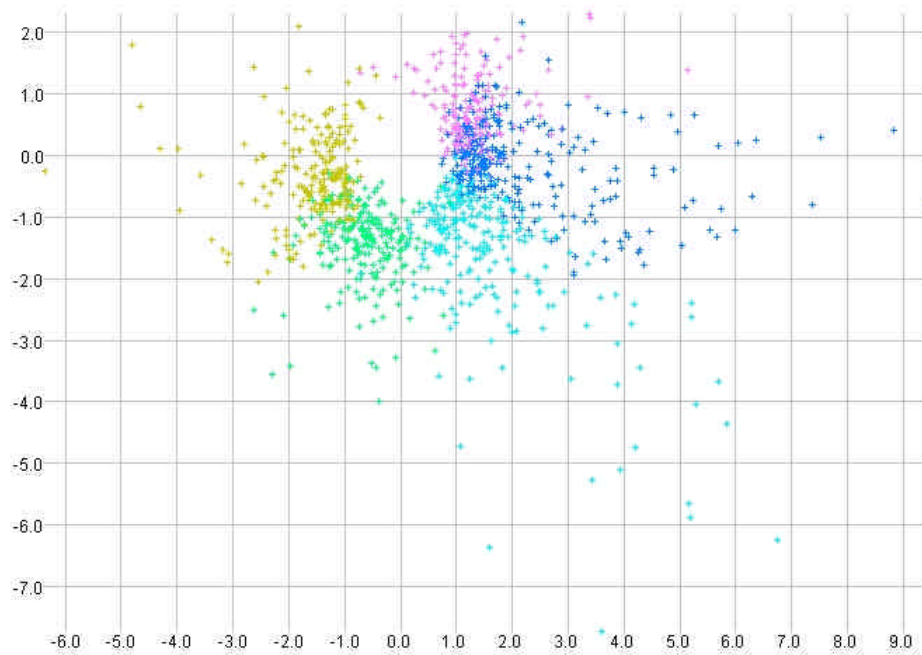


Figure 7.3.5.b. A PCA visualization of the data. Elements from the same cluster appear in the same color.

7.3.6 Functional analysis

To identify functional classes that are specifically enriched in each of the cell cycle phases, functional analysis was performed twice: (1) using only the 874 genes identified as periodically expressed during cell cycle as the background set. (2) using the whole genome as the background set.

Figure 7.3.6.a shows the visualization of the first analysis. Cluster 2 (S phase) was found to be highly enriched for genes of the functional class 'DNA metabolism' ($p = 2 \times 10^{-11}$) and 'S phase of mitotic cell cycle' ($p = 6 \times 10^{-7}$). It was also found to be

enriched for genes of the 'DNA replication and chromosome cycle', 'replisome' and 'replication fork' functional classes ($p < 0.0005$). Cluster 3 (G2) was found to be enriched for genes of the 'mitosis' functional class ($p = 1.5 \cdot 10^{-4}$). Cluster 4 (G2/M) was found to be enriched for genes of the functional classes 'cytoskeleton' ($p = 4 \cdot 10^{-5}$) and 'microtubule cytoskeleton' ($p = 2 \cdot 10^{-4}$). The results are summarized in table 7.3.6.a.

The results in the second analysis support the results in the first, and the p-values obtained had even higher significance, but they contain additional classes that were not identified in the first analysis (results not shown). Some of those classes (e.g., 'mitotic cell cycle' and 'regulation of cell cycle') were identified in several of the clusters. Other classes were identified in only one or two of the clusters, e.g., 'DNA repair' in clusters 1 and 2, 'response to DNA damage stimulus' in clusters 1 and 2.

The reason to these differences in results is the significant over-representation of genes of cell cycle related functional classes in the entire set of 874 genes. Random partitions of the dataset are also expected to yield clusters where some of these functional classes are enriched with respect to the entire genome. Thus, it is preferable to use the filtered dataset as the background set if one wishes to detect phase-specific functionalities, and to use the unfiltered background for finding general cell-cycle functions.

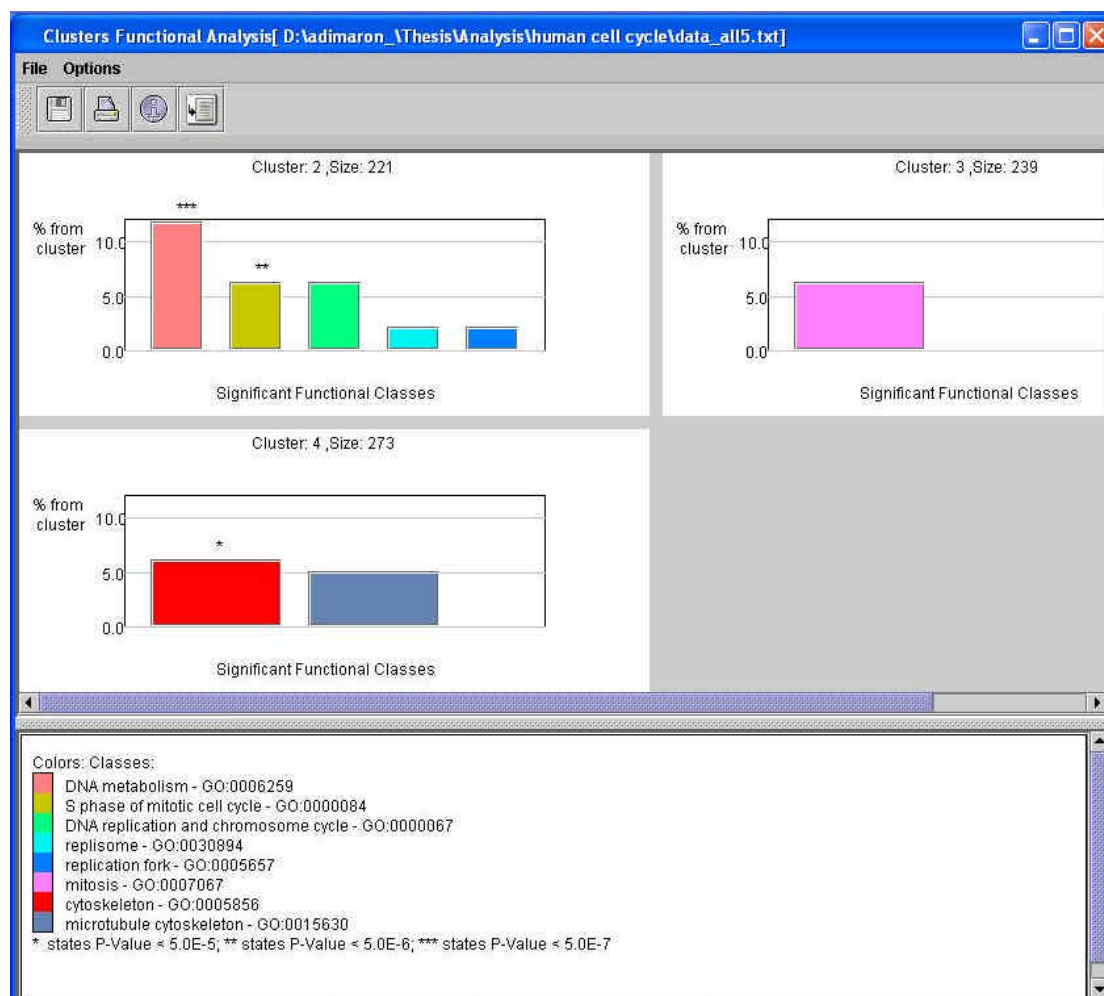


Figure 7.3.6.a. The functional analysis visualization. Threshold p-value was set to 0.0005. Only genes identified as periodically expressed during cell cycle were used as background. Clusters 2 (S phase), 3 (G2) and 4 (M/G1) were identified as significantly enriched with several functional classes.

Cluster (phase):	Enriched functional class:	p-value:
2 (S)	DNA metabolism	2.317E-11
	S phase of mitotic cell cycle	6.358E-7
	DNA replication and chromosome cycle	8.734E-5
	replisome	2.081E-4
	replication fork	2.081E-4
3 (G2)	mitosis	1.498E-4
4 (G2/M)	cytoskeleton	4.143E-5
	microtubule cytoskeleton	2.234E-4

Table 7.3.6.a: Functional analysis results when using the filtered dataset as background.

7.3.7 Promoter analysis

In previous work (Elkon *et al.* 2002), promoter analysis was performed on the above clusters using the PRIMA software. At the time of the analysis the available promoter set contained sequences for 568 of the 874. The human fingerprint file that is currently used by EXPANDER contains TF fingerprints for promoter sequences of 717 out of the 874 cell cycle responding genes, so the richer set was used in the current analysis.

The promoter analysis was performed twice: (1) using only the 874 genes identified as periodically expressed in cell cycle as the background set; (2) using the whole genome as the background set. We do not show again the visualization but rather concentrate on the results. These are summarized in Tables 7.3.7.a and b for the analysis (1) and (2), respectively.

In analysis (1), only promoters from cluster 1 (G1/S) were found to be significantly enriched with TF-binding sites. All TF binding sites that were identified as enriched in analysis (1) in cluster 1 (G1/S) were also detected in analysis (2), with higher p-values. This is due to the significant overrepresentation of these motifs in the entire set of 874 genes (see table 7.3.7.c). The same analysis was previously performed by Elkon *et al.* and yielded no results with p-value $< 5 \times 10^{-4}$. The improvement is clearly due to the updated fingerprint files.

Both Arnt and YY1 PWMs that were previously identified by Elkon *et al.* as significantly enriched ($p < 0.001$) in promoters of genes which are expressed in G1/S and M/G1, respectively, were not detected in this analysis. This is probably due to the differences in the fingerprint files that were used for the analyses. Since the current analysis is based on substantially more promoters, we believe that Arnt and YY1 were false positive detections.

Cluster	TF	Number of promoters with hits	Number of hits	p-value:
1 (G1/S)	E2F	26	31	1.41×10^{-7}
	Sp1	36	51	4.27×10^{-4}
	Ncx	20	21	1.56×10^{-4}

Table 7.3.7.a. Promoter analysis results when using filtered data as background

Cluster	TF	Number of promoters with hits	Number of hits	p-value:
1 (G1/S)	E2F	19	24	3.1×10^{-11}
	NF-Y	39	54	1.07×10^{-7}
	Ncx	20	21	5.9×10^{-6}
	Sp1	36	51	2.96×10^{-5}
2 (S)	E2F	17	20	1.21×10^{-9}
	NF-Y	26	40	6.69×10^{-4}
3 (G2)	NF-Y	44	69	3.86×10^{-8}
	Sp1	60	86	6.23×10^{-5}
4 (G2/M)	NF-Y	55	81	5.52×10^{-9}
5 (M/G1)	NF-Y	32	45	3.44×10^{-5}
	CREB	25	31	9.1×10^{-4}

Table 7.3.7.b. Promoter analysis results when using the unfiltered data as background

To explore the distribution of binding sites in the filtered dataset in relation to the whole genome, a different clustering file was loaded to EXPANDER, classifying all 874 genes into one cluster. Promoter analysis was then performed using the entire genome as background set. The results are summarized in table 7.3.7.c. All TF binding site motifs that were identified as enriched in the entire set of cell cycle responding genes in the previous study (*Elkon et al. 2002*) were detected in this study as well. Two additional motifs, Alpha-CP1 and ETF were identified in this study.

TF	Number of promoters with hits	Number of hits	p-value:
E2F	55	64	9.84×10^{-23}
NF-Y	174	267	3.83×10^{-13}
Sp1	225	322	1.12×10^{-7}
Alpha-CP1	83	105	1.15×10^{-5}
ETF	234	378	4.46×10^{-5}
CREB	99	118	8.47×10^{-5}
ATF	110	127	1.68×10^{-4}
Nrf-1	106	134	4.94×10^{-4}

Table 7.3.7.c. Promoter analysis results on the entire set of cell cycle periodically expressed genes, when using the whole genome as background.

7.3.8 Discussion

The E2F family is well documented as a prime regulator of the mammalian cell-cycle. Pathways that modulate the activity of E2F are frequently disrupted in human cancers, leading to mis-regulated cellular proliferation (*Nevins 2001*). The E2F PWM obtained highly significant enrichment scores in all the analyses performed by Elkon et al. (2002), and also in the analysis described above, which was performed using more data, demonstrating the sensitivity of PRIMA in revealing true signals. As in the analysis performed by Elkon et al., E2F was found to be highly enriched in promoters of genes that are expressed in G1/S and in S phases.

Three TFs, E2F, SP1 and Ncx were detected as enriched ($p\text{-value} < 5 \times 10^{-4}$) in cluster 1 (G1/S) in this analysis, when using only the cell cycle responding genes as background set. These were not detected in the analysis performed by Elkon et al., probably due to the differences in the fingerprint files that were used for the analyses. The Sp1 has been previously shown to be involved in cell cycle regulation (*Clem et al. 2003*). Ncx is known to be expressed in neural crest derived tissues (*Iitsuka et al. 1999*). No evidence that connects Ncx to cell cycle regulation has been found.

All the TFs that were previously identified by Elkon et al. as highly enriched in promoters of all 874 genes (in comparison to their prevalence in promoters of the whole genome), were detected again in this analysis. In addition, two new TFs, Alpha-CP1 and ETF were detected as highly enriched. ETF is a known transcriptional activator of p53 (*Hale and Braithwaite 1999*). Alpha-CP1 is a transcription factor that belongs to a group of factors which are known to bind to the sequence CAATT (*Alonso et al. 1996*).

8 Bibliography

Alon, U., Barkai N., et al. Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *Proc Natl Acad Sci U S A* 96(12), 6745-50, 1999.

Alonso, C.R., Pesce, C.G. and Kornblihtt, A.R. The CCAAT-binding proteins CP1 and NF-I cooperate with ATF-2 in the transcription of the fibronectin gene. *Biol Chem.* 271(36):22271-9, 1996.

Amundson, S.A., Bittner, M. and Fornace, A.J.Jr. Functional genomics as a window on radiation stress signaling. *Oncogene.* 22(37):5828-33, 2003.

Ashburner, M., Ball, C.A., Blake, J.A., Botstein, D., Butler, H., Cherry, J.M., Davis, A.P., Dolinski, K., Dwight, S.S., Eppig, J.T., Harris, M.A., Hill, D.P., Issel-Tarver, L., Kasarskis, A., Lewis, S., Matese, J.C., Richardson, J.E., Ringwald, M., Rubin, G.M. and Sherlock, G. Gene ontology: tool for the unification of biology. The Gene Ontology Consortium. *Nat Genet.* 25(1):25-9, 2000.

Banin, S., L. Moyal, S. Shieh, Y. Taya, C. W. Anderson, L. Chessa, N. I. Smorodinsky, C. Prives, Y. Reiss, Y. Shiloh, and Y. Ziv. Enhanced phosphorylation of p53 by ATM in response to DNA damage. *Science* 281:1674-7, 1998.

Ben-Dor, A., Shamir, R., Yakhini, Z. Clustering gene expression patterns. *J Comput Biol* 6(3-4):281-97 1999.

Bolstad, B. M. Irizarry, R. A. Astrand, M. and Speed, T. P. A Comparison of Normalization Methods for High Density Oligonucleotide Array Data Based on Variance and Bias. *Bioinformatics* 19(2):185-193, 2003.

Boorsma, A., de Nobel, H., ter Riet, B., Bargmann, B., Brul, S., Hellingwerf, K.J. and Klis, F.M. Characterization of the transcriptional response to cell wall stress in *Saccharomyces cerevisiae*. *Yeast* 21(5):413-27, 2004.

Braxton, S. and Bedilion, T. The integration of microarray information in the drug development process. *Curr Opin Biotechnol* 9(6), 643-9, 1998.

Brazma, A., Hingamp, P., Quackenbush, J., Sherlock, G., Spellman, P., Stoeckert, C., Aach, J., Ansorge, W., Ball, C.A., Causton, H.C., Gaasterland, T., Glenisson, P., Holstege, F.C., Kim, I.F., Markowitz, V., Matese, J.C., Parkinson, H., Robinson, A., Sarkans, U., Schulze-Kremer, S., Stewart, J., Taylor, R., Vilo, J. and Vingron, M. Minimum information about a microarray experiment (MIAME)-toward standards for microarray data. *Nat Genet.* 29(4):373, 2001.

Chong, M. J., Murray, M. R., Gosink, E.C., Russell, H. R., Srinivasan, A., Kapsetaki, M., Korsmeyer, S. J., McKinnon, P. J. . Atm and Bax cooperate in ionizing radiation-induced apoptosis in the central nervous system. *Proc Natl Acad Sci U S A.* 97:889-894, 2000.

Clarke, P. A., te Poele, R. et al. Gene expression microarray analysis in cancer biology, pharmacology, and drug development: progress and potential. *Biochem Pharmacol* 62(10), 1311-36, 2001.

Clem, A.L., Hamid, T. and Kakar, S.S. Characterization of the role of Sp1 and NF-Y in differential regulation of PTTG/securin expression in tumor cells. *Gene*. 322:113-21, 2003.

Dresen, I.M., Husing, J., Kruse, E., Boes, T. and Jockel, K.H. Software packages for quantitative microarray-based gene expression analysis. *Curr Pharm Biotechnol*. 4(6):417-37, 2003.

Dudoit, S., Gentleman, R.C. and Quackenbush, J. Open source software for the analysis of microarray data. *Biotechniques*. Suppl:45-51, 2003.

Dysvik, B. and Jonassen, I.J-Express: exploring gene expression data using Java. *Bioinformatics* 17(4):369-70 2001.

Dyrskjot, L., Thykjaer, T., Kruhoffer, M., Jensen, J.L., Marcussen, N., Hamilton-Dutoit, S., Wolf, H. and Orntoft, T.F. Identifying distinct classes of bladder carcinoma using microarrays. *Nat Genet*. 33(1):90-6, 2003.

Eisen, M. B., Spellman, P. T. et al. Cluster analysis and display of genome-wide expression patterns. *Proc Natl Acad Sci U S A* 95(25), 14863-8, 1998.

Elkon, R., Linhart, C. Sharan, R. Samir, R. and Shiloh, Y. Genome-Wide In Silico Identification of Transcriptional Regulators Controlling the Cell Cycle in Human Cells. *Genome Research*, Vol. 13(5), pp. 773-780, 2003.

Everitt, B. 1993. Cluster analysis. London: Edward Arnold, third edition.

Fredrickson, H. L., Perkins, E. J. et al. Towards environmental toxicogenomics -- development of a flow-through, high-density DNA hybridization array and its application to ecotoxicity assessment. *Sci Total Environ* 274(1-3), 137-49, 2001.

Gasch, A.P., Spellman, P.T., Kao, C.M., Carmel-Harel, O., Eisen, M.B., Storz, G., Botstein, D., Brown, P.O. Genomic expression programs in the response of yeast cells to environmental changes. *Mol Biol Cell*. 11(12):4241-57, 2000.

Glanzer, J.G., Haydon, P.G. and Eberwine, J.H. Expression profile analysis of neurodegenerative disease: advances in specificity and resolution. *Neurochem Res*. 29(6):1161-8, 2004.

Golub, T. R., Slonim, D. K. et al. Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science* 286(5439), 531-7, 1999.

Hale, T.K. and Braithwaite, A.W. The adenovirus oncoprotein E1a stimulates binding of transcription factor ETF to transcriptionally activate the p53 gene. *J Biol Chem*. 274(34):23777-86, 1999.

Hansen, P. and Jaumard, B. Cluster analysis and mathematical programming. *Mathematical Programming* 79:191-215, 1997.

Hartigan, J. Clustering Algorithms. John Wiley and Sons, 1975.

Hieter, P. and Boguski, M. Functional genomics: it's all how you read it. *Science* 278(5338), 601-2, 1997.

Hinnebusch, A.G. Evidence for translational regulation of the activator of general amino acid control in yeast. *Proc Natl Acad Sci U S A.* 81(20):6442-6, 1984.

Hughes, T.R., Mao, M., Jones, A.R., Burchard, J., Marton, M.J., Shannon, K.W., Lefkowitz, S.M., Ziman, M., Schelter, J.M., Meyer, M.R., Kobayashi, S., Davis, C., Dai, H., He, Y.D., Stephanian, S.B., Cavet, G., Walker, W.L., West, A., Coffey, E., Shoemaker, D.D., Stoughton, R., Blanchard, A.P., Friend, S.H., Linsley, P.S. Expression profiling using microarrays fabricated by an ink-jet oligonucleotide synthesizer. *Nat Biotechnol.* 19(4):342-7, 2001.

Iitsuka, Y., Shimizu, H., Kang, M.M., Sasagawa, K., Sekiya, S., Tokuhisa, T. and Hatano, M. An enhancer element for expression of the *Ncx* (*Enx*, *Hox11L1*) gene in neural crest-derived cells. *J Biol Chem.* 274(34):24401-7, 1999.

Irwin, R.D., Boorman, G.A., Cunningham, M.L., Heinloth, A.N., Malarkey, D.E. and Paules, R.S. Application of toxicogenomics to toxicology: basic concepts in the analysis of microarray data. *Toxicol Pathol.* 32 Suppl 1:72-83, 2004.

Ishida, S., Huang, E. et al. Role for E2F in control of both DNA replication and mitotic functions as revealed from DNA microarray analysis. *Mol Cell Biol* 21(14), 4684-99, 2001.

Jelinsky, S. A., Estep, P. et al. Regulatory networks revealed by transcriptional profiling of damaged *Saccharomyces cerevisiae* cells: Rpn4 links base excision repair with proteasomes. *Mol Cell Biol* **20**(21), 8157-67, 2000

Kandror, O., Bretschneider, N., Kreydin, E., Cavalieri, D. and Goldberg, A.L. Yeast adapt to near-freezing temperatures by *STRE/Msn2,4*-dependent induction of trehalose synthesis and certain molecular chaperones. *Mol Cell.* 13(6):771-81, 2004.

Kuras, L. and Thomas, D. Identification of the yeast methionine biosynthetic genes that require the centromere binding factor 1 for their transcriptional activation. *FEBS Lett.* 367 pp.15-18 ,1995.

Li, B., Nierras, C.R. and Warner, J.R. Transcriptional elements involved in the repression of ribosomal protein synthesis *Mol Cell Biol.* 19(8):5393-404 (1999).

Li, C. and W. H. Wong. Model-based analysis of oligonucleotide arrays: model validation, design issues and standard error applications. *Genome Biology* 2(8), 1–11 ,2001.(a)

Li, C. and Wong, W. H. Model-based analysis of oligonucleotide arrays: Expression index computation and outlier detection. *PNAS*, 98(1),31-36,2001.(b)

- Li, H., Singh, A.K., McIntyre, L.M. and Sherman, L.A. Differential gene expression in response to hydrogen peroxide and the putative PerR regulon of *Synechocystis* sp. strain PCC 6803. *J Bacteriol.* 186(11):3331-45, 2004.
- Li, N., S. Banin, H. Ouyang, G. C. Li, G. Courtois, Y. Shiloh, M. Karin, and G. Rotman. ATM is required for I κ B kinase (IKK κ) activation in response to DNA double strand breaks. *J Biol Chem* 276:8898-903, 2001. (c)
- Liu, J., Blackhall, F., Seiden-Long, I., Jurisica, I., Navab, R., Liu, N., Radulovich, N., Wigle, D., Sultan, M., Hu, J., Tsao, M.S. and Johnston, M.R. Modeling of lung cancer by an orthotopically growing H460SM variant cell line reveals novel candidate genes for systemic metastasis. *Oncogene*. 2004 Jul 12 [Epub ahead of print].
- Lord, P.G. Progress in applying genomics in drug development. *Toxicol Lett.* 149(1-3):371-5, 2004.
- Martinez-Pastor, M.T., Marchler, G., Schuller, C., Marchler-Bauer, A., Ruis, H. and Estruch, F. The *Saccharomyces cerevisiae* zinc finger proteins Msn2p and Msn4p are required for transcriptional induction through the stress response element (STRE). *EMBO J.* 15(9):2227-35, 1996.
- Marton, M. J., DeRisi, J. L. et al. Drug target validation and identification of secondary drug target effects using DNA microarrays. *Nat Med* 4(11), 1293-301, 1998.
- Matys, V., Fricke, E., Geffers, R., Gossling, E., Haubrock, M., Hehl, R., Hornischer, K., Karas, D., Kel, A.E., Kel-Margoulis, O.V., Kloos, D.U., Land, S., Lewicki-Potapov, B., Michael, H., Munch, R., Reuter, I., Rotert, S., Saxel, H., Scheer, M., Thiele, S. and Wingender, E. TRANSFAC: transcriptional regulation, from patterns to profiles. *Nucleic Acids Res.* 31(1):374-8, 2003.
- McGall, G.H., Christians, F.C. High-density genechip oligonucleotide probe arrays. *Adv Biochem Eng Biotechnol.* 77:21-42, 2002.
- Mirkin, B. *Mathematical Classification and Clustering*. Kluwer, Dordrecht 1996.
- Moehle, C.M. and Hinnebusch, A.G. Association of RAP1 binding sites with stringent control of ribosomal protein gene transcription in *Saccharomyces cerevisiae*. *Mol Cell Biol.* 11(5):2723-35, 1991.
- Nevins, J.R. The Rb/E2F pathway and cancer. *Hum. Mol. Genet.* 10: 699-703, 2001.
- Nuwaysir, E. F., Bittner, M. et al. Microarrays and toxicology: the advent of toxicogenomics. *Mol Carcinog* 24(3), 153-9, 1999.
- Quackenbush, J. Computational analysis of microarray data. *Nat Rev Genet* 2(6), 418-27, 2001.
- Pilpel, Y., Sudarsanam, P. et al. Identifying regulatory networks by combinatorial analysis of promoter elements. *Nat Genet* 29(2), 153-9, 2001
- Piret, B., S. Schoonbroodt, and J. Piette. The ATM protein is required for sustained activation of NF- κ B following DNA damage. *Oncogene* 18:2261-71, 1999.

Raychaudhuri, S., Stuart, J.M., Altman, R.B. Principal components analysis to summarize microarray experiments: application to sporulation time series. *Pac Symp Biocomput.*:455-66, 2000.

Reich, M., Ohm, K., Angelo, M., Tamayo, P. and Mesirov, J.P. GeneCluster 2.0: an advanced toolset for bioarray analysis. *Bioinformatics* [Epub ahead of print],2004.

Ripley, B.D. The R project in statistical computing. *MSOR Connections*. The newsletter of the LTSN Maths, Stats & OR Network., 1(1):23-25, 2001.

Saito, S., A. A. Goodarzi, Y. Higashimoto, Y. Noda, S. P. Lees-Miller, E. Appella, and C. W. Anderson. ATM mediates phosphorylation at multiple p53 sites, including Ser(46), in response to ionizing radiation. *J Biol Chem* 277:12491-4, 2002.

Sakaki, K., Tashiro, K., Kuhara, S. and Mihara, K. Response of genes associated with mitochondrial function to mild heat stress in yeast *Saccharomyces cerevisiae*. *J Biochem (Tokyo)*. 134(3):373-84, 2003.

Sattlegger, E., Swanson, M.J., Ashcraft, E.A., Jennings, J.L., Fekete, R.A., Link, A.J., Hinnebusch, A.G. YIH1 is an actin-binding protein that inhibits protein kinase GCN2 and impairs general amino acid control when overexpressed. *J Biol Chem*. 2004

Schadt, E., C. Li, B. Eliss, and W. H. Wong. Feature extraction and normalization algorithms for high-density oligonucleotide gene expression array data. *J. Cell. Biochem*. 84(S37),120–125, 2002.

Sgal, E., Yelensky, R., Kaushal, A., Pham, T., Regev, A., Koller, D. and Friedman, N. GeneXPress: A Visualization and Statistical Analysis Tool for Gene Expression and Sequence Data. *Proceedings of the 11th Inter. Conf. on Intelligent Systems for Molecular Biology (ISMB)*, 2004.

Shalon, D., Smith, S. J., et al. A DNA microarray system for analyzing complex DNA samples using two-color fluorescent probe hybridization. *Genome Res* 6(7), 639-45, 1996

Shannon, W., Culverhouse, R. and Duncan, J. Analyzing microarray data using cluster analysis. *Pharmacogenomics*. 4(1):41-52, 2003.

Sharan, R. and Shamir, R. CLICK: a clustering algorithm with applications to gene expression analysis. *Proc Int Conf Intell Syst Mol Biol* 8, 307-16, 2000.

Sharan, R., Maron-Katz, A. and Shamir, R. CLICK and EXPANDER: A System for Clustering and Visualizing Gene Expression Data. *Bioinformatics* Vol. 19 No. 14 pp. 1787--1799, 2003.

Shore, D., RAP1: a protean regulator in yeast. *Trends Genet*. 10 pp. 408–412, 1994.

Spellman, P. T., Sherlock, G., et al. Comprehensive identification of cell cycle-regulated genes of the yeast *Saccharomyces cerevisiae* by microarray hybridization. *Mol Biol Cell* 9(12), 3273-97, 1998.

- Sturn, A., Quackenbush, J. and Trajanoski, Z. Genesis: cluster analysis of microarray data. *Bioinformatics*. 18(1):207-8, 2002.
- Suckow, M., Schwamborn, K., Kisters-Woike, B., von Wilcken-Bergmann, B. and Muller-Hill B. Replacement of invariant bZip residues within the basic region of the yeast transcriptional activator GCN4 can change its DNA binding specificity. *Nucleic Acids Res.* 22(21):4395-404, 1994.
- Tamayo, P., Slonim, D., Mesirov, J., Zhu, Q., Kitareewan, S., Dmitrovsky, E., Lander, E. S., and Golub, T. R. Interpreting patterns of gene expression with self-organizing maps: methods and application to hematopoietic differentiation. *Proc Natl Acad Sci U S A*, 96, 2907-2912, 1999.
- Tanay, A. Sharan, R. and Shamir, R. Discovering statistically significant biclusters in gene expression data. *Bioinformatics*, 18(1), 136-144, 2002.
- Tanay, A., Sharan, R., Kupiec, M. and Shamir, R. Revealing modularity and organization in the yeast molecular network by integrated analysis of highly heterogeneous genomewide data. *PNAS* 101 (9) 2981-2986, 2004.
- Tavazoie, S., Hughes, J. D., Campbell, M. J., Cho, R. J., and Church, G. M. Systematic determination of genetic network architecture. *Nat Genet*, 22: 281-285, 1999.
- van de Vijver, M.J., He, Y.D., van't Veer, L.J., Dai, H., Hart, A.A., Voskuil, D.W., Schreiber, G.J., Peterse, J.L., Roberts, C., Marton, M.J., Parrish, M., Atsma, D., Witteveen, A., Glas, A., Delahaye, L., van der Velde, T., Bartelink, H., Rodenhuis, S., Rutgers, E.T., Friend, S.H. and Bernards, R. A gene-expression signature as a predictor of survival in breast cancer. *N Engl J Med.* 347(25):1999-2009, 2002.
- Waddell, S.J., Stabler, R.A., Laing, K., Kremer, L., Reynolds, R.C. and Besra GS. The use of microarray analysis to determine the gene expression profiles of *Mycobacterium tuberculosis* in response to anti-bacterial compounds. *Tuberculosis (Edinb)*. 84(3-4):263-74, 2004.
- Warner, J.R. The economics of ribosome biosynthesis in yeast. *Trends Biochem Sci.* 24(11):437-40, 1999.
- Whitfield, M.L., Sherlock, G., Saldanha, A.J., Murray, J.I., Ball, C.A., Alexander, K.E., Matese, J.C., Perou, C.M., Hurt, M.M., Brown, P.O., Botstein, D. Identification of genes periodically expressed in the human cell cycle and their expression in tumors. *Mol Biol Cell.* 13(6):1977-2000, 2002.
- Wodicka, L., Dong, H., et al. Genome-wide expression monitoring in *Saccharomyces cerevisiae*. *Nat Biotechnol* 15(13), 1359-67, 1997.
- Yang, C.R., Wilson-Van Patten, C., Planchon, S.M., Wuerzberger-Davis, S.M., Davis, T.W., Cuthill, S., Miyamoto, S. and Boothman, D.A. Coordinate modulation of Sp1, NF-kappa B, and p53 in confluent human malignant melanoma cells after ionizing radiation. *FASEB J.* 2000 Feb;14(2):379-90.
- Zhao, R., Gish, K. et al. Analysis of p53-regulated gene expression patterns using oligonucleotide arrays. *Genes Dev* 14(8), 981-93, 2000.

