Doctoraatsproefschrift nr. 887 aan de faculteit Bio-ingenieurswetenschappen van de K.U.Leuven

PREPROCESSING AND BICLUSTERING OF HIGH-THROUGHPUT EXPRESSION DATA

Hui ZHAO

Dissertation presented in partial fulfilment of the requirements for the degree of Doctor in Bioscience Engineering

March 2010

ISBN 978-90-8826-128-2
D/2010/11.109/6

Acknowledgements

First and foremost, I sincerely thank my promoter, Prof. Kathleen Marchal, for giving me the opportunity to pursue my PhD in the field of bioinformatics four years ago and guiding me in the right track throughout my study. Kathleen is a role model of a researcher for me with great passion on research and deep understanding of the field. What I learned from her is not only knowledge, but also an attitude towards research. She was always there for me whenever there were difficulties. There were so many hours I can recall that we spent together on discussing research problems. I should not forget to mention that now we have a brand new topic, babies. Also, with the team work atmosphere created in our group by Kathleen, I never felt alone at work. Kathleen, without you, I would not have reached this stage. Thank you for everything!

I also want to express my gratefulness to my co-promoter, Prof. Bart De Moor, for his trust on me. His comments were always encouraging and kept me on the right direction.

I would also like to thank two assessors of this dissertation, Prof. Jos Vanderleyden and Prof. Mieke Verstuyf, for their valuable feedbacks on different stages of my research and on polishing this dissertation.

In addition, it is an honour for me to have Prof. Jan Ramon and Prof. Florence d'Alché-Buc in my doctoral committee. With their suggestions, the quality of this dissertation is significantly improved. detailed

I would like to acknowledge my colleagues who I have worked with during all these years of my PhD study. Kristof, thank you for helping me gain the fundamentals of microarrays, especially when I was a beginner, and for giving me the insightful explanations whenever I had questions. Sigrid and Inge, from our collaboration I obtained more concrete understanding on microarray data and got familiar with the biologists' way of thinking. Tim, I can not count how many of our meetings starting with you saying “I did some thinking in the car….” Thank you for the pleasant cooperation. Riet, thank you for your patience on validating the results and for your time on all the discussions we had. Lore, thank you for all your efforts to help me out of a lot of deadlines.

I also want to thank my other colleagues, Abeer, Aminael, Carolina, Fu Qiang, Ivan, Karen, Lyn, Peyman, Pieter, Sun Hong, Thomas, Valerie and Wu Yan. I had such nice time working with you guys. I will definitely miss the conferences, brain storms and those Friday lunches we had together.

Papa, mama, you have always believed in me and supported my decision unconditionally. Without your understanding and support, I can not go this far for my study.

Finally, I would like to give my special thanks to my husband, Xinliang. Your unchanging love and support were with me from the beginning. Your patience helped me get through the last stressful periods. Thank you for sharing with me all the wonderful moments with our lovely daughter Youran.

Abstract

As nowadays microarray experiments have gained common ground in many laboratories, inferring transcriptional networks from high-throughput expression data is becoming increasingly feasible. Both data normalization and module inference are crucial steps in the inference process.

Therefore, we first investigated how to properly preprocess microarray data from different experiments, different laboratories or different techniques. To overcome problems with current data normalization methods, we designed the BioConductor package CALIB which estimates absolute expression levels from two-color microarray data using external spikes. In addition we developed an automated analysis flow to analyse in house generated data. The analysis flow consists of a quality assessment of the generated microarray data, data normalization, the identification of differentially expressed genes and/or clustering of the expression profile and validation of the results by exploiting information in curated public databases.

Secondly, we contributed to the network inference problem itself. State of the art network inference problems reduce the complexity of the inference problem by exploiting the modularity of gene networks: in contrast to assigning an individual program to each single gene as is done with direct network inference methods, module-based network inference procedures assign a regulatory program to pre-grouped sets of coexpressed genes (modules). This drastically lowers the number of interactions that needs to be evaluated during the inference process. Module inference is thus a crucial step in the inference problem. Module inference is solved by biclustering microarray datasets. We developed a novel biclustering framework, *Pro*Bic based on the Probabilistic Relational Model framework that has advantages over current state of the art bicluster algorithms. An evaluation on both synthetic and biological data illustrates the performance of *Pro*Bic in identifying sets of coexpressed genes.

Lastly, we studied whether the outcome of different transcriptional network inference algorithms is affected by the methods to generate the microarray expression compendia on which the algorithms were applied.

Korte inhoud

Afgeleid uitgebreide regulerende netwerken van high-throughput data is een van de belangrijkste uitdagingen van de moderne systeembiologie. Als high-throughput expressie profilering experimenten hebben Vaststaat, opgedaan in vele laboratoria, zijn verschillende technieken voorgesteld om regulerende netwerken van hen af te leiden en veel inspanning gaat naar de ontwikkeling van algoritmen die de structuur van regulatorische netwerken afleiden uit deze gegevens. Biclustering algoritmen hebben het voordeel van het ontdekken van genen die coexpressed in een subset van de gemeten voorwaarden. Biclustering past bij de noodzaak voor ontdekking van de regelgeving modules, die van essentieel belang aanwijzingen voor het openbaren van regulerende netwerken bieden.

In dit onderzoek, enerzijds, onderzochten we hoe goed Preprocessen microarray data van verschillende experimenten, verschillende laboratoria of verschillende technieken ter verbetering van de vergelijkbaarheid van deze gegevens. We ontwierpen een Bioconductor pakket CALIB absolute expressie niveaus schatten uit twee kleuren microarray data met behulp van externe spikes. De methode voorkomt de Global Normalization Assumption waarop de meeste van normalisatie methoden vertrouwen en stelt het voordeel ten opzichte van log-ratio gebaseerde benaderingen. We ontwikkelden ook een microarray data analyse stroom die bijdraagt tot de kwaliteitsbeoordeling van microarray data, data normalisatie, de identificatie van differentieel tot expressie genen, clustering van de expressie profiel en de analyse van het cluster resultaten. Ten tweede, ontwikkelden we een nieuw biclustering model ProBic, die is gebaseerd op de Probabilistische relationele model kader. Een evaluatie van zowel synthetische en biologische gegevens illustreert de kracht van het model werkt als een query-gedreven biclustering model. Ten derde, bestudeerden we de invloed van het gebruik van verschillende microarray compendium normalisatie benaderingen over de resultaten van verschillende regelgevende algoritmen netwerk gevolgtrekking.

Tot slot concludeerde we de belangrijkste onderzoeks-resultaat van ons werk en stelde een vooruitzichten voor toekomstig onderzoek op dit gebied.

Acronyms

AIC	Akaike Information Criterion
ANOVA	Analysis Of VAriance
AQBC	Adaptive Quality-Based Clustering
BIC	Bayesian Information Criterion
cDNA	complementary DNA
ChIP	Chromatin ImmunoPrecipitation
CLR	Context Likelihood of Relatedness
CPD	Conditional Probability Distribution
DNA	Deoxyribonucleic acid
EM	Estimation Maximization
FDR	False Discovery Rate
FN	False Negatives
FP	False Positives
GAN	Gene Aging Nexus database
GEO	Gene Expression Omnibus
GNA	Global Normalization Assumption
GO	Gene Ontology
IM	Ideal Missmatch
ISA	Iterative Signature Algorithm
ITTACA	Integrated Tumor Transcriptome Array and Clinical data Analysis
JPD	Joint Probability Distribution
LIMMA	Linear Models for Microarray Data
LOESS	Locally Estimated Scatter plot Smoother
LOWESS	Locally WEighted Scatter plot Smoother
M3D	Many Microbe Microarrays Database
MAP solution	Maximum A Posteriori solution
MAS	MicroArray Suite
MBEI	Model-Based Expression Index
MIAME	Minimum Information About a Microarray Experiment
MM	MissMatch

mRNA	messenger RNA
PCR	polymerase chain reaction
PM	Perfect Match
PRMs	Probabilistic Relational Models
RMA	Robust Multi-array Average convolution
RNA	Ribonucleic acid
ROC	Receiver Operating Characteristic curves
SAM	Significance Analysis of Microarrays
SMD	Stanford Microarray Database
SOM	Self-Organizing Maps
TN	True Negatives
TP	True Positives

Table of Contents

Chapter 1

Introduction

1.1 High-throughput Data

The discovery of DNA structure in 1953 was the starting point of a real scientific and cultural revolution. The discovery and use of enzymes that copy, cut and join DNA molecules in cells were the next step in this revolutionary course. With the development of techniques like the manual DNA sequencing appeared in 1975, the polymerase chain reaction (PCR) discovered in 1985 and the automation of DNA sequencing succeeded since 1986, the processes of replication, transcription and translation of the genetic material have been extensively studied. The study of biology that deals with the nature of biological phenomena at a molecular level is called molecular biology. Molecular biology has uncovered a multitude of biological facts, such as the interactions between the various systems of a cell, including the interactions between DNA, RNA and protein as well as how these interactions are regulated. However, just like if we want to understand the complexity underlying the engineered object of an airplane, only looking at the list of components is not sufficient even though the airplane is actually comprised of these components. A biological system is not just an assembly of genes or proteins and their interconnections. It can be understood at different levels such as cells, tissues, organs and organisms. These are all systems of components whose specific interactions have been defined by evolution. Therefore, while the understanding of individual genes and proteins continues to be important, the system-level understanding will help us gain more insight in the complexity of a biological system. Systems biology emerged in this respect. Systems biology does not investigate individual genes or proteins one at a time. It investigates the behaviour and interactions of all the components in a particular biological system while it is functioning [1]. The development of systems biology has been driven by obtaining, integrating and analyzing high-throughput data from various experimental sources. These high-throughput biological experiments supply thousands of measurements per sample and generate high-throughput data to be used to enhance inference at a system level. The research presented in this dissertation is based entirely on the preprocessing and integration high-throughput data.

1.2 Microarrays and Microarray Compendia

Traditional molecular biological techniques have provided valuable biological insights, but they are limited by the scale of data that can be obtained from a single experiment. In the past decade, the development of cDNA and oligonucleotide microarrays [2] have made it possible to measure the expression levels of thousands of genes simultaneously and produce huge amounts of valuable data. Microarray technique has become one of the most commonly used high-throughput experiments to understand the roles various genes play in different biological processes.

Different microarray platforms exist such as cDNA microarrays, Affymetrix, Agilent, Codelink or in-house microarrays. Each different platform requires its own optimized sample preparation, labelling, hybridization and scanning protocol and concurrently also a specific preprocessing procedure. Preprocessing of the raw, extracted intensities aims to remove consistent and systematic sources of variation to ensure comparability of the measurements, both within and between arrays.

Microarray experiments are made publicly available in specialized databases such as Gene Expression Omnibus [3], Stanford microarray database [4] or ArrayExpress [5]. Exacting information from these public databases remains tedious. Information is not only stored in different formats and data models, but is also redundant, incomplete and/or inconsistent. In order to fully exploit the large source of information offered by these public databases, various species-specific microarray compendia are constructed that combine all the experiments on one particular organism. The experiments in a microarray compendium are normalized individually and then normalized and annotated as a whole to improve the comparability.

1.3 Reconstruction of transcriptional networks from Microarray data

One of the most challenging tasks of systems biology is to reconstruct structures and mechanisms of interaction between components of cellular systems from the available experimental data. A transcriptional network is a collection of DNA segments in a cell which interact with each other and with other substances in the cell, thereby governing the rates at which genes are transcribed into mRNA. Genes can be viewed as nodes in such a network, with input being proteins such as transcription factors and outputs being the level of gene expression. One gene can affect the expression of another gene by binding of the gene product of one gene to the promoter region of another gene. Looking at more than two genes, we refer to the transcriptional network as the regulatory interaction between the genes. Therefore, with the development of the microarray techniques and the advent of microarray compendia, reconstructing transcriptional networks can often be formulated as a problem of gene network inference from microarray expression data. The classical approaches for network reconstruction from gene expression data aimed at inferring the interactions between all genes. Many methods are available nowadays for inferring modules from microarray expression data. For example Boolean models, Bayesian networks, differential equations and hybrids of those have been described (for exhaustive overviews we refer to D'Haeseleer *et al.*[6], van Sommeren *et al.* [7] and de Jong *et al.* [8]).

1.4 Objectives

We have now introduced the principle of microarray analysis and transcriptional networks, which allow us to define our goal for this dissertation. Several issues need to be addressed concerning microarray data and reconstruction of transcriptional networks from microarray data.

At first because of the properties of the technical procedure of an array analysis, consistent sources of variation (technical variation) might obscure detecting the true biological variation in which one is interested in. To remove these consistent sources of variation requires the adequate preprocessing procedures. Currently most microarray normalization methods rely on the Global Normalization Assumption (GNA), which is proven less appropriate when the expression patterns between two tested biological samples are expected to differ considerably. Moreover, the use of log-ratios required for most of the common normalization methods makes the interpretation of the results depend on the choice of the reference sample. To overcome these problems we proposed a novel normalizing procedure for two-color microarray data that avoids the global normalization assumption and the use of ratio's as estimates of the differential expression level.

Secondly, microarray data are very informative for studying biological mechanism on a global scale. However, it is essential to assess the quality of the data, and to remove as much as possible the technical sources of variation. In this respect, systematic preprocessing procedures are necessary to check and remove all the possible technical sources of variation from the raw microarray data, so that in the ideal world, the variation in the data is only explained by biology. Then normalized microarray data need to be further analyzed to reveal the biological results. Therefore we developed in this work a standard microarray analysis flow that can be applied to a common microarray experiment.

Thirdly, as described in Section 1.3, traditional methods for network inference from gene expression data consider every gene as an individual node in the network, and their goal is to infer all interactions between these nodes. Because of the large search space when training all genes as individual nodes, most of these methods have extensive data requirements obviating their practical usage. However, for a biologist, the primary interest does not lay so much in reconstructing interactions between all genes but in recovering the interactions between the regulators and their target genes. Therefore, conceptual simplifications that reduce the complexity of the inference problem are possible. One such simplification is the modular description of the transcriptional network [9]: in contrast to assigning an individual program to each single gene as is done with direct network inference methods, module-based network inference procedures assign a regulatory program to pre-grouped sets of coexpressed genes (modules). This drastically lowers the number of interactions that needs to be evaluated during the inference process. Module inference is thus a crucial step in the inference problem. Module inference is solved by biclustering microarray datasets. We developed a biclustering algorithm to identify overlapping biclusters in gene expression data using Probabilistic Relational Models, which is able to incorporate biological prior information in the form of a set of seed genes.

Fourthly, the emergence of microarray compendia introduced a new impulse to transcriptional network inference, namely inferring large-scale transcriptional networks. Thus far, a multitude of algorithms have been proposed for inferring transcriptional network from a microarray compendium. Generating an expression compendium from public data is not trivial and can be done by applying a whole range of normalization methods to make the data from different public experiments mutually comparable. As using a different normalization methods can influence the results of different transcriptional network inference algorithms, we studied how different normalization methods can affect the inference results.

1.5 Overview of this dissertation

Figure 1.1 shows how the dissertation is organized and how the different chapters are related with each other. Hereafter, a brief introduction of each chapter is given.

Chapter 2 is a survey of microarrays and microarray compendia. Section 2.2 gives an introduction on microarray technology, on preprocessing of raw microarray data and on the methods available for postprocessing of microarray data. Section 2.3, following the similar structure as Section 2.2 provides an introduction on how integrated microarray compendia can be made, how data within a compendium need to be preprocessed to guarantee their mutual comparability. Finally an overview is given of analysis methods that can be applied on a microarray compendium.

Chapter 3 presents in detail a BioConductor package CALIB designed for estimating absolute expression levels from two-color microarray data. In this package, a novel cDNA microarray normalization method is implemented. The method avoids the Global Normalization Assumption on which most of normalization methods rely and poses the advantage over log-ratio based approaches. This package has been published:

Zhao H., Engelen K., De Moor B., and Marchal K. CALIB: a BioConductor package for estimating absolute expression levels from two-color microarray data. Bioinformatics 2007; 23(13) 1700-1701. doi: 10.1093/bioinformatics/btm159

Chapter 4 describes a designed cDNA microarray data analysis work flow which contributes to the quality assessment of microarray data, data normalization, the identification of differentially expressed genes, clustering of the expression profile and analysis of the cluster results. The work flow is explained by the study of AI-2 mediated quorum sensing in *Salmonella* Typhimurium. The work flow is also applied on other biological cases. The articles describing the cases on which the developed work flow was applied are as follows:

Janssens JC, Steenackers H, Robijns S, Gellens E, Levin J, Zhao H, Hermans K, De Coster D, Verhoeven TL, Marchal K, Vanderleyden J, De Vos DE, De Keersmaecker SC. 2008. Brominated furanones inhibit biofilm formation by Salmonella enterica serovar Typhimurium. Appl Environ Microbiol. 74: 6639-6648.

Thijs IVM, De Keersmaecker S, Fadda A, Engelen K, Zhao H, McClelland M, Marchal K, Vanderleyden J. Delineation of the Salmonella Typhimurium HilA regulon through genome-wide location and transcript analysis. J Bacteriol. 2007 Jul; 189(13) 4587-96

Thijs IVM, De Keersmaecker S, Fadda A, Engelen K, Zhao H, McClelland M, Marchal K, Vanderleyden J. Combining omics data to unravel the regulatory network controlling Salmonella invasion of epithelial cells. Commun Agric Appl Biol Sci. 2007. 72: 55-59.

Thijs IM, Zhao H, De Weerdt A, Engelen K, Schoofs, G, De Coster D, McClelland M, Vanderleyden J, Marchal K, De Keersmaecker SCJ. The AI-2 dependent regulator LsrR has a limited regulon in Salmonella Typhimurium. Under revision.

De Keersmaecker SCJ, Zhao H, Sonck KAJ, Thijs IM, De Coster D, van Boxel N, Engelen K, Northen T, Vanderleyden J, Marchal K. Integrating high-throughput data reveals important role for AI-2 in Salmonella enterica serovar Typhimurium flagellar phase variation. In preparation.

Chapter 5 introduces a novel biclustering model *Pro*Bic, which is based on the Probabilistic Relational Model framework. The model itself is explained in detail in this chapter. An evaluation on both synthetic and biological data illustrates the strength of the model working as a query-driven biclustering model. This work is ready to be published:

Zhao H., Van den Bulcke T., De Smet R., Cloots L., Engelen K., De Moor B., and Marchal K. Efficient query-driven biclustering of gene expression data using Probabilistic Relational Models. In preparation.

Chapter 6 studies the influence using different microarray compendium normalization approaches on the outcome of different transcriptional network inference algorithms.

Finally, Chapter 7 concludes the main research result of our work and proposes an outlook for future research in this domain.

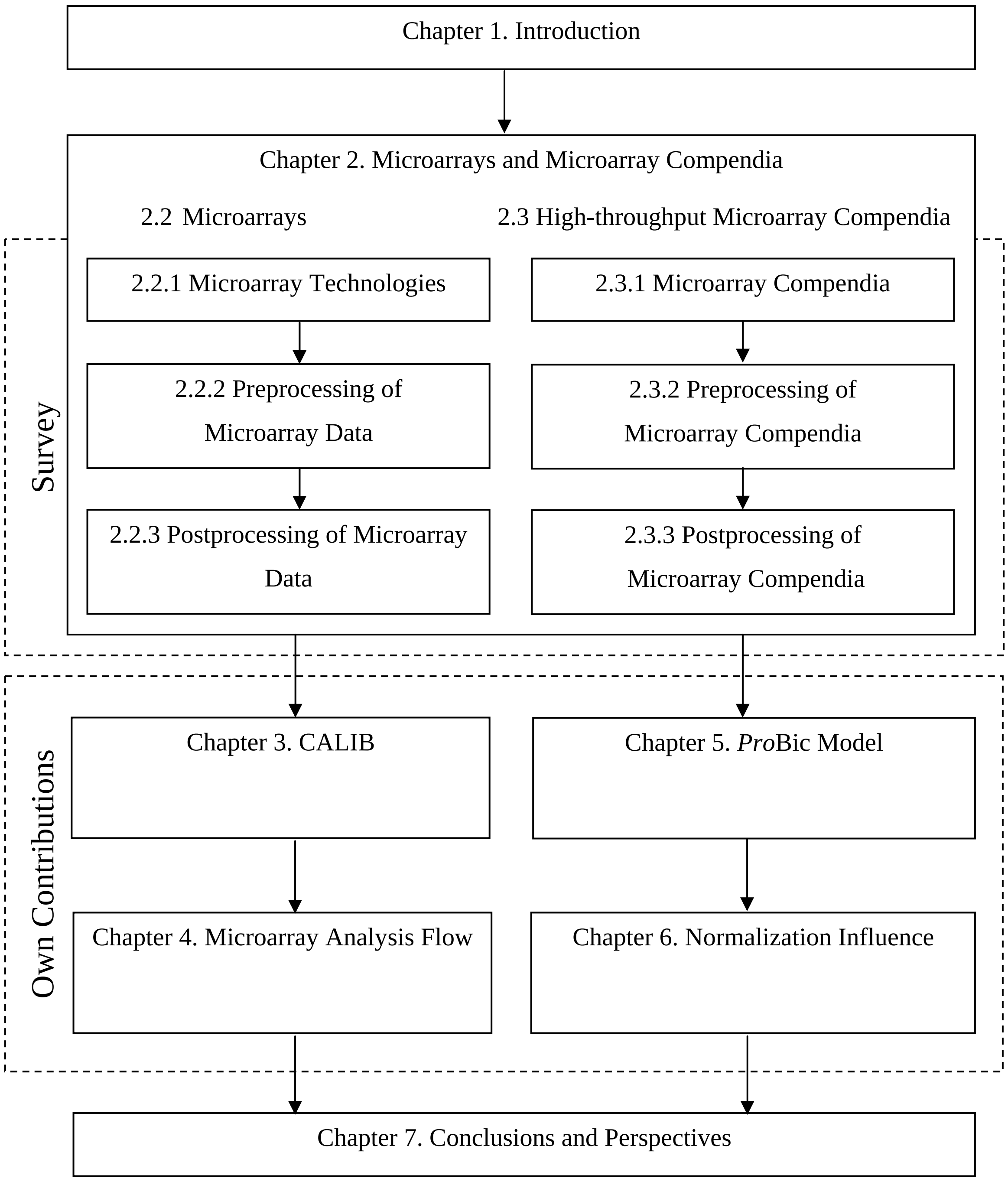

Chapter 1. Introduction
Chapter 2. Microarrays and Microarray Compendia
2.2 Microarrays
2.3 High-throughput Microarray Compendia
2.2.1 Microarray Technologies
2.2.2 Preprocessing of Microarray Data
2.2.3 Postprocessing of Microarray Data
2.3.1 Microarray Compendia
2.3.2 Preprocessing of Microarray Compendia
2.3.3 Postprocessing of Microarray Compendia
Survey
Own Contributions
Chapter 3. CALIB
Chapter 4. Microarray Analysis Flow
Chapter 5. *Pro*Bic Model
Chapter 6. Normalization Influence
Chapter 7. Conclusions and Perspectives

Figure 1.1: Organization of this dissertation.

Chapter 2

Microarrays and Microarray Compendia

High-throughput experiments allow measuring the expression levels of mRNA (genomics), protein (proteomics) and metabolite compounds (metabolomics) for thousands of entities simultaneously. They provide wealth of data that can be used to develop a global insight into the cellular behaviour. The most powerful experimental designs consist of surveying a biological system in a wide array of responses, phenotypes or conditions. The combination of these experimental data and right computational analysis tools can lead to powerful new finding with various applications. One of the main contributors to the high-throughput applications is the development of microarray technologies.

In this chapter, we provide a survey concerning microarrays and microarray compendia. We start with an overview of different technologies that are used to perform microarray experiments followed by a survey of various preprocessing methods that help to remove the systematic effects which arise from variation in the techniques rather than from tested biological samples. Possible microarray data analysis methods are listed after the survey of microarray preprocessing techniques. Next, we make a survey of different categories of microarray compendia, preprocessing of a microarray compendium and the different algorithms applied on a compendium.

2.1 Introduction

A microarray is a chip (i.e. array) on the surface of which single-stranded DNAs (called probes) are bound in grid. When exposed to an RNA or cDNA sample obtained from a certain biological study, a microarray is able to capture a snapshot of the transcription levels (i.e. the mRNA levels) of tens of thousands of genes or even a whole genome under the assessed experimental condition. By performing microarray experiments under different conditions, biologists can simultaneously monitor the behaviour of the genes at the transcription level. The transcriptional behaviour of a gene is thus described by its expression profile, which is made up of the expression levels of the gene under different conditions.

There are different technologies available for the manufacturing of microarray chips. However, the main mechanism is similar for all the technologies. Microarrays used for gene expression profiling contain probes representing target genes for the study. The probes are then labelled, usually with fluorescent dyes, and finally exposed to the chip. The measurement of the expression level of a gene relies on the binding of its corresponding labelled samples to the probes representing the genes on the chip. Once the hybridization is finished, the unhybridized materials are washed away, and the chip is scanned so that the intensity of the fluorescence for each probe is read out.

However, from the building of the chips and the preparation of the probes, to the array hybridization and the final scanning procedure, every step involved in a microarray experiment introduces noise and artefacts to the readout intensity data. Thus the raw data obtained from a microarray experiment needs to go under various preprocessing procedures to remove the systematic variations before any further analysis can be carried out.

When integrating microarray experiments independently performed by different laboratories and on different platforms into a species-specific microarray compendium, choosing and performing preprocessing procedures to generate the compendium becomes even more crucial. Such compendia then can be used to contribute to the understanding of a certain organism at a more global level.

2.2 Microarrays

2.2.1 Microarray technologies

The mainstream microarray technologies can be classified into two categories – spotted arrays and *in situ* synthesized arrays. In spotted arrays, pre-synthesized DNA probes, which are typically oligonucleotides (usually 50-80 bases), long complementary DNA strains (100-1000 bases) or small fragments of PCR products corresponding to specific genes are spotted on the solid surface, such as glass, plastic or silicon slides. In *in situ* synthesized arrays, the probes, which are short sequences designed to represent a single gene or family of gene splice-variants, are synthesized directly onto the array surface. The probes can be longer (60 bases) or shorter (25 bases) depending on the desired purpose. Because oligonucleotides are typically used as probes for in situ synthesized arrays, these arrays are often referred to as oligonucleotide arrays. In the following section, we will use cDNA arrays (a type of spotted arrays) and Affymetrix

Gene Chip (a type of *in situ* synthesized arrays) as the examples to have further explanation and discussion. The microarrays mentioned in Chapter 4 of this dissertation function on a similar principle to cDNA arrays in that labelled probes are hybridized to unlabelled probes fixed on to a solid surface. They differ in the nature of the probes. 70-mer oligonucleotides are used rather than cDNA. Other than that, the mechanism is the same as a cDNA microarray experiment. All preprocessing procedures and methods for subsequent analysis applied for cDNA microarrays are also applicable for them.

2.2.1.1 cDNA microarrays

The probes on cDNA microarrays are cDNAs fragments that correspond to mRNAs. The probes are synthesized prior to deposition on the array surface and are then spotted onto the slides by a set of spotting pins. The pins draw fluid containing the probe DNAs from a microtiter plate and then spot the probes on the substrate surface by directly contact with the slides. Each spot on the printed microarray contains one probe representing one gene. cDNA microarrays sometimes contain control probes designed to hybridize with RNA spike-ins. An RNA spike-in is an RNA transcript used to calibrate measurements in a cDNA microarray experiment. Manufacturers of commercially available microarrays typically offer companion RNA spike-in "kits". Known amounts of RNA spike-ins are mixed with the experiment sample during preparation. Subsequently the measured degree of hybridization between the spike-ins and the control probes can be used to normalize the hybridization measurements of the sample RNA. In addition to normalization purposes, the spike-in experiments allow to quantify the accuracy of microarrays, i.e. to compare the observed measurement with the real concentration of RNA for the spikes probes.

In a cDNA microarray experiment, a mixture of mRNA samples derived from two conditions (e.g. one is from wild type and the other is from mutant) is hybridized to one microarray. Each sample is labelled with a different fluorescent dye, Cy3 or Cy5. After hybridization, the slide is scanned in a microarray scanner at the wavelengths for Cy3 and Cy5. Relative intensities of each fluorophore at each spot reflect the expression level of the corresponding gene. Figure 2.1 illustrates the procedure of measuring by using a cDNA microarray.

2.2.1.2 Affymetrix Gene Chip

Affymetrix uses a combined technology of photolithography and combinatorial chemistry to synthesize nucleotides to the multiple growing chains of oligonucleotides on the surface of the chip [11]. Figure 2.2 illustrates the manufacturing procedure of Affymetrix Gene Chip.

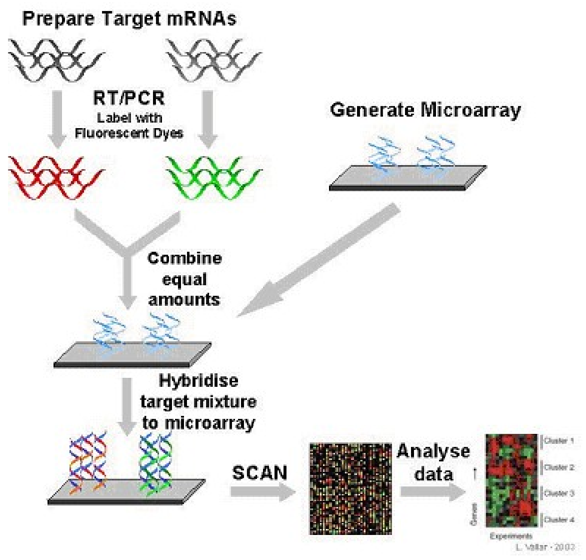


Figure 2.1: cDNA microarray experiment (figure source from http://www.crp-sante.lu/files/images/Profiling%20schema_0.jpg). mRNAs are extracted from the two chosen cell samples of interest and reverse transcribed to complementary DNAs (cDNAs). Fluorescent dyes (Cy3 and Cy5) are labelled to the two cDNA samples. The labelled cDNA samples are called probes. Equal amount of cDNA probes are tested by hybridizing them to a pre-generated cDNA microarray. Then the microarray is scanned to determine the amount of each probe is bound to each spot. The intensities provided by the scanned image will be quantified. After being preprocessed, the data can be used to do further analysis.

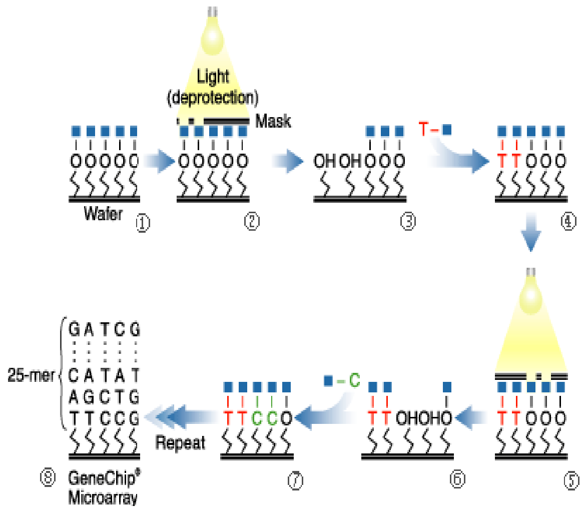


Figure 2.2: The manufacturing procedure of Affymetrix Gene Chip (figure source: http://cnx.org/content/m12388/latest/). (1) The surface is coated with a covalent linker molecule coated with a light-sensitive agent that prevents further nucleotide binding to them until they are subsequently exposed to light. (2)(5) The surface is overlayed with the first mask and

exposed to the light source. (3)(6) Linker molecules at the unprotected position are activated. (4)(7) The array surface is flooded with the nucleotide linked to the light-sensitive agent. The nucleotides are linked to the activated end. (8) The whole procedure is repeated using a set of designed masks until the probes reaches the length of 20-30 bases.

On the Affymetrix Gene Chip, each gene is represented by a probe set. A probe set usually contains 11 to 20 oligonucleotide probe pairs of length 25 bases. Each pair consists of a perfect match (PM) oligonucleotide and a mismatch (MM) oligonucleotide. A PM probe is complementary to the gene sequence of interest while a MM probe is identical to its PM counterpart except for one mismatch base inserted at its central position. The MM probe can be used to estimate the signal of non-specific hybridization, i.e. the binding to the given probe of other sequences, which are only partially complementary to it [12]. Gene expression levels can be calculated based on the weighted average of difference in PM and MM intensities across the entire set of probes.

2.2.2 Preprocessing of Microarray Data

Microarray data are very informative to be used to study biological mechanism in a global scale. However, the noise level in microarray data is quite high. The high noise level is caused by technical variation which exists in every step of one microarray experiments, from mRNA sample preparation to all the experimental variables which include slide and spot morphology, hybridization specificity, differences in the efficiency of labeling reactions, signal strength, background fluorescent noise etc. [13] The technical variation might obscure the biological variation in which one is interested in. Therefore, it is essential to assess the quality of the data, and remove as much as possible the technical noise and sources of variation. In this respect, preprocessing procedures are designed to check and remove all the possible technical sources of variation from the raw microarray data, so that in the ideal world, the variation in the data is only explained by biology.

2.2.2.1 Replicates and Experimental Design

Replication is essential for identifying and reducing the technical noise in microarray. Single measurements are subject to random effects. It is also necessary to repeat experiments to rule out noise. There are two types of replicates, biological and technical. Biological replicates use RNA independently derived from distinct biological sources and provide both a measure of the natural biological variability in the samples under study. Random noise can be caused in the procedure of sample preparation or in the process of a microarray experiment. Technical replicates include replicated probes for a particular gene within a single array and replicated probes on different arrays. Biological replicates are independent samples that are varying all the variables that a person in the lab could not control. It is usually preferable to have biological replicates than technical replicates.

Replicates are easy to design for Affymetrix since only one biological sample is measured on a single array. The experimental design for cDNA microarray is more complex because two biological samples are measured simultaneously on one array. Therefore, the issue of experimental design is addressed to ensure that the microarray data are amenable to further analysis.

A simple and effective design for the direct comparison between two samples is dye-swap [14, 15]. This design uses two microarrays to compare two samples A and B. One the first array, sample A is labelled with the red (Cy5) dye and sample B is labelled with green (Cy3) dye while on the second array, the dye assignments are reversely. Dye-swap is widely applied in the current microarray experiments to reduce the technical sources of variation, particularly compensating for the dye bias.

If one aims to study the expression profiles of genes under multiple conditions rather than comparing differential expression in two biological samples, two experimental designs can be used as describing in Figure 2.3: the reference design and the loop design.

- The reference design (as in Figure 2.3(left)), each biological sample is hybridized against a common reference sample, which can be the wild type or the biological controls. Dye-swap can be applied between each pair of comparison. The advantage of this design is that when a common reference is preserved, comparison between different biological samples is very straightforward and the design can easily be extended to other experiments is very easy. When considering integration of microarray data from different laboratories, this design is relatively robust to the difference caused by the laboratory effects. However, when using this design, half of the resources is consumed on the common reference, which mostly likely is not the sample of biological interest.

- The loop design (as in Figure 2.3(right)), each biological sample is hybridized to each of two other samples in two different dye orientations. Dye-swap can also be applied between each pair of comparison. The advantage is that this design utilizes a large number of direct sample to sample comparisons if the samples are of equal interest. As shown in Figure 2.3(right), using a loop design each sample occurs four times on the arrays, rather than two times under a reference design, at the cost of six microarray experiments in total for both designs. The obvious drawback is that if one array fails or has a bad quality, the influence on the comparison is significant. From data integration point of view, it is also difficult to integrate microarray data from various sources without careful normalization if the loop design is used.

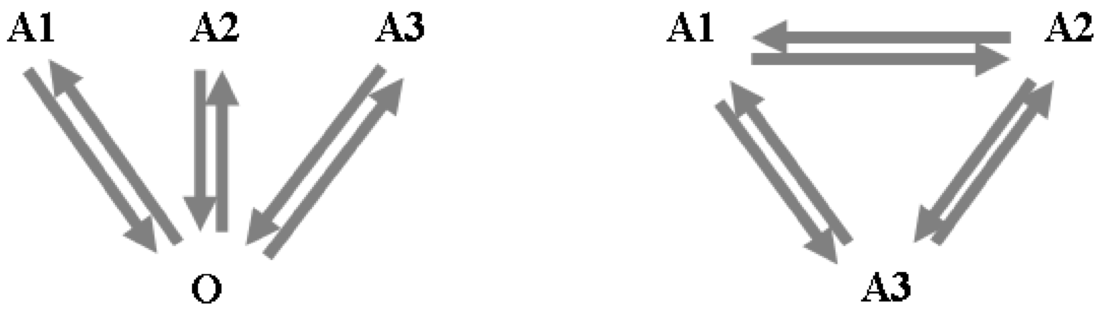


Figure 2.3: Examples of Microarray design. A1, A2, A3 represent different biological samples, O represents common reference and arrows represent hybridizations between the mRNA samples and the microarray. The sample at the tail of the arrows is labelled with red (Cy5) dye, and the sample at the head of arrows is labelled green (Cy3) dye. The left and right panel explains a reference design and a loop design respectively.

The reference design and the loop design are just two of many existing microarray experimental designs. Other designs like a saturation design or different combinations of type of design, number of replication and use of dye-swap [14, 15] can also result in useful analysis. How to choose a “right” design depends on many factors, as the goals of experiments, the resources, the cost, the method of analysis and so on.

2.2.2.2 Quality Assessment

After the microarray experiments are designed and performed, the first step of preprocessing the data is to decide whether the data obtained from one microarray experiment or from one particular microarray in a set of experiments is beyond correction and should be removed from further analysis. There are numerous ways to quantitatively assess the quality of microarray data and then to filter out the data with bad quality. For instance, one of the proposed criterions is to exclude any spot with intensity lower than the background plus two standard deviations [16, 17]. However, the threshold for this type of quality assessment usually lacks consensus among different analyzers and is quite arbitrary in most of the cases. A simpler and more straightforward way for assessing microarray data is to visualize the data and interfere manually.

Visualization techniques facilitate to assess the microarray qualities in two respects: within-array quality assessment and between-array quality assessment. Within-array assessment focuses on the quality of one microarray. For both cDNA microarrays and Affymetrix, it could reveal the spatial non-uniformity (due to such as damage or contamination on the surface of the microarray, plate effects or print-pin effects). In the case of cDNA microarrays, other aspects such as low contrast between the foreground and the background intensities, and abnormality in the size or the shape of spots can also be checked by the visualization techniques. Between-array assessment aims at assessing the homogeneity between replicated experiments, including both biological replicates and technical replicates.

2.2.2.3 Background Correction

The motivation for background correction of microarray data is the belief that a spot's measured intensity does not only result from the hybridization of the target to the probe, but also to non-specific hybridization or spatial heterogeneity across the arrays.

Most of the image analysis software for cDNA microarrays returns foreground and background intensities obtained for each spot after segmentation of the image [18, 19]. Background correction estimates the effect of background in a local neighbourhood of each segmented spot and is used to subtract this estimate from the foreground signal. In the case of Affymetrix, PM probes are used for measuring specific binding while MM probes are used to measure optical noise and non-specific binding directly. Hence, background correction can be simply performed by the MM intensity from the corresponding PM intensity [20, 21].

However, the subtraction of these measured background signals has been under debate. The subtraction background correction strategy commonly used in cDNA microarray relies on the assumption that local background generates additive bias to the foreground signal [22]. However, if noise was additive, background subtracted log-intensity ratios would show less variability compared to the ones for which background correction was not performed. Results of different researches [18, 19, 23] indicate that this is not the case, so the assumption that background noise is additive is too simplistic. Therefore, whether background correction for cDNA microarray analysis is performed depends on a personal choice. No background correction is applied in the microarray analysis flow presented in Chapter 4 of this dissertation.

For Affymetrix, the simple subtraction is not always applicable. Irizarray *et al*. (2003) [24] have illustrated that the transformation PM – MM produces an expression estimate with large variance. Moreover, for about 1/3 of probes the intensity obtained for MM probes is larger than PM intensity. Consequently, several popular Affymetrix analysis approaches propose different alternatives to perform background correction:

- MAS 5.0 [20]: Affymetrix proposed the use of ideal mismatch (IM), a corrected MM intensity which is guaranteed to be smaller than the corresponding PM intensity. IM is obtained by adjusting the biweight specific background (SB) calculated from the robust average over the log-ratios between the corresponding PMs and MMs in the probe set. The adjusted PM intensity is based on the difference PM – IM.

- Model-Based Expression Index (MBEI) [25]: A statistical model of how the probe intensity values respond to changes of the expression levels of a gene is proposed. The expression level estimates are constructed from all the intensity values for the PM and MM probes corresponding to this gene, assuming that within the same probe pair, the PM intensity will increase at a higher rate than the MM intensity.

- Robust Multi-array Average convolution (RMA) [26]: The background correction of the RMA method only uses the PM intensities. It assumes that observed PM value is composed of two terms, one generated from a normal distribution (Bg) which explains the background noise, and the other being an exponential signal component (S). The normal distribution is truncated at zero to avoid negative background signals.

2.2.2.4 Log transformation

Before normalization, a logarithm transformation is often performed. Taking log transformation is successful at reducing the intensity-dependent variation [26, 27], and makes the noise of the microarray data additive. Moreover, cDNA microarrays are often used to compare expression levels between two biological samples. These comparisons are typically expressed for each gene as the ratio of the intensities measured from each dye labelled sample. Even though the ratios provide an intuitive measure for comparing two expression levels, they have the disadvantage of treating up- and down-regulated expression ratios differently [28]. Up-regulated ratios range from one to infinity, while down-regulated ratios range from zero to one. By taking log transformation previously non-symmetrical data is transformed into a more symmetrical data distributed around zero. This means that up- and down-regulated genes are treated in similar fashion.

2.2.2.5 Normalization

Normalization attempts to adjust individual hybridization intensities to remove as many as possible those effects which arise from variation in the techniques rather than from tested different biological samples. There are a number of reasons why normalization is essential, including unequal amount of starting RNA, difference in print-tip group on one slide, dye bias in labelling or detection efficiencies and other systematic biases in the measured expression level.

Various methods are developed for normalizing microarray data. Most of them are platform dependent. For cDNA microarrays, theoretically, the systematic noises originated from spot effect, print-tip effect, plate effect and array effect can be divided away using log ratios. Hence, the normalization methods mentioned in this section mainly focus on the removal of dye-related discrepancies from the log-ratios.

These normalization methods can be summarized into two major categories: within-array normalization and between-array normalization. Firstly, within-array normalization has to be performed to adjust for artificial differences in intensities of the two labels for each individual microarray. The following approaches are available in this sense:

- Global normalization: It assumes that the red and green intensities are related by a constant factor overall intensity. Namely, $R = kG$. Different choices for the constant factor are advised. Chen *et al.* (2002) [27] propose an iterative method is used for estimating the constant normalization factor k and cut-offs for the red and green intensity ratio *R/G*.

- Intensity dependent linear normalization: In many cases, the dye bias appears to be dependent on spot intensity, as revealed by the MA-plot (shown in Figure 2.4(left)). The MA plot uses M as the y-axis and A as x-axis where M and A are usually defined as $M = \log_2 R - \log_2 G$ and $A = \frac{1}{2} \times (\log_2 R + \log_2 G)$ given (logR, logG) be the red and green (background-corrected) intensities. Linear normalization assumes the relation between M and A is linear, in the form of $M = \beta_0 + \beta_1 A$, where (β_0, β_1) can be estimated by least squares estimation.

- Intensity dependent nonlinear normalization: These normalization methods consider the relation between M and A is in the form of M = *c*(A), rather than a linear relation. The most popular and widely used nonlinear normalization approach was first described by Yang *et al* (2002) [29]. The estimation of *c*(A) is made by using a LOWESS (locally weighted scatter plot smoother) function [30] to perform a local scatter plot smoothing to the MA-plot. The scatter plot smoother, a type of regression analysis, performs robust locally linear fits by calculating a moving average along the A axis. Robust in this context means that the curve is not affected by a small moderate percentage of differentially expressed genes that appear as outliers in the MA-plot. The outcome of the LOWESS normalization is revealed in Figure 2.4(right).

The above normalization methods are mainly for controlling dye discrepancies of logarithmically transformed intensities. They are often applied to each individual microarray. Yang *et al*. (2001) [19] also propose scale normalization methods, which is a representative of between-array normalization approaches. Scale normalization is a simple scaling of the M values from a series of arrays so that each array has the same median absolute deviation. The main purpose of scale normalization is to control between array variability and to facilitate comparison and integration across different microarrays.

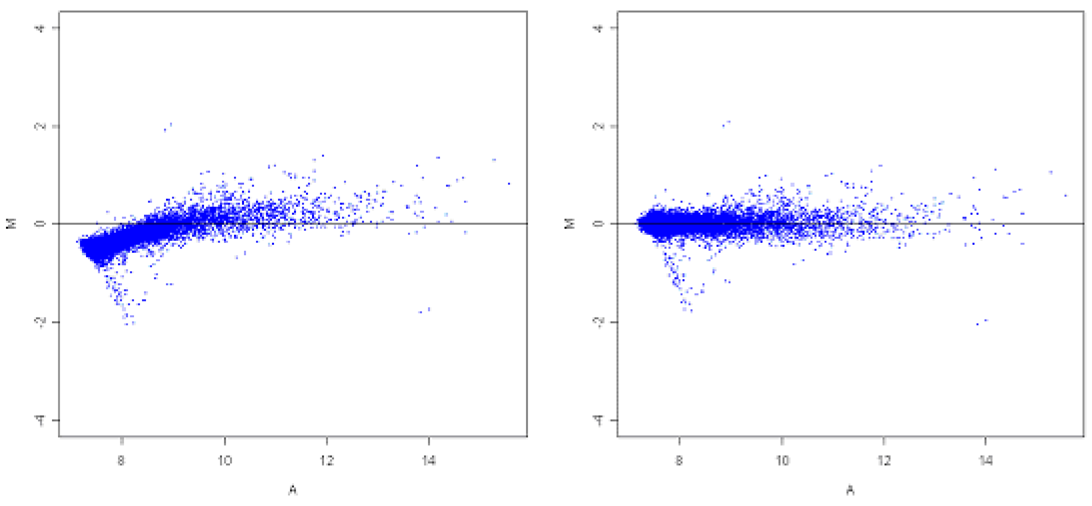


Figure 2.4: Intensity dependent normalization. Left: MA-plot showing the dye bias is dependent on spot intensities. The x-axis presents the intensity of the genes by the A (add) values, which are the added log-intensities of the red (R) and green (G) channels. The y-axis gives the M (minus) values of the genes, which are the log-ratios of the two channel intensities. Right: MA-plot after LOWESS normalization.

For Affymetrix, normalization aims at removing the array effects and the probe effects. Similar as for cDNA microarrays, both linear and nonlinear methods exist for the normalization of Affymetrix:

- Linear Normalization methods: The simplest linear approach is to re-scale each array in an experiment by its total intensities. Forcing array distributions to have the same central tendency (arithmetic mean, geometric mean, median) can be accomplished by a scaling factor in this scaling method.

- Nonlinear Normalization methods: A popular representative is quantile normalization [31], whose goal is to impose the same empirical distribution of intensities to each array. For convenience, the pooled distribution of probes on all arrays is taken as the empirical distribution. The algorithm first ranks the probe intensities from the lowest to highest for each array, so that each rank represents a quantile. Then, the average intensity value across all the arrays is calculated within each quantile. Finally, the measured intensity in a given quantile in an array is replaced by the calculated average intensity for that quantile. The transform is $x_{norm} = F_2^{-1}(F_1(x))$, where F_1 is the distribution function of the actual array, and F_2 is the empirical distribution.

After the normalization procedure, the data measured by different probe sets need to be summarized to produce one measure for the expression level of each gene on each array. Common summarization methods include average difference (which simply computes the average difference between PM and MM intensities over all probe sets of one gene), one-step Tukey's biweight estimate [20], median polish fit [32] to a linear model describing the log-intensities as a three-term-addition (with one term being the true log expression level, another one describing the probe effect and the third one for normally distributed noise) [24].

2.2.2.6 Discussion

Various published microarray preprocessing methods are mentioned in the previous sections. Three issues concerning microarray preprocessing procedures are essential to be addressed.

Firstly, although preprocessing can not remove all systematic variations, it can not be denied that the whole preprocessing procedure plays an important role in the earlier stage of microarray data analysis because the resulting preprocessed expression data can significantly vary when different preprocessing approaches are used. Subsequent analysis, such as differential expression detecting, classification and clustering are quite dependent on a choice of the whole preprocessing procedure. More detailed introduction about the subsequent analyses will be in Section 2.2.3.

Secondly, in Section 2.2.2.4, it is mentioned that the log transformed ratios in cDNA microarray is widely used to control several sources of experimental variation. However, conventional microarray ratios have several properties that limit subsequent computational analysis and also the amount of information that can be extracted from these high-throughput data. The first aspect is that reporting expression data as ratios between the tested biological sample and the reference clearly says the difference between these two samples on the same spot. However, the ratios can not provide information about the absolute expression levels carried by the individual spot intensity. Thus the ratios obscure the essential differences in levels of gene expression between different genes. The second aspect is that the use of the ratios largely dependent on the choice of the reference sample, which is uncharacterized and not easily reproduced. This will hamper comparison and integration between data sets using different references.

Finally, a range of cDNA microarray normalization approaches are introduced in Section 2.2.2.5. These methods assume that the majority of the genes on the array are non-differentially expressed between the two samples and that the number of over-expressed genes approximately equals the number of under-expressed genes, an assumption referred as Global Normalization Assumption (GNA). These assumptions can be inappropriate for when testing two drastically different biological samples or when working with custom arrays. In such cases, these normalization methods may yield unreliable results.

2.2.2.7 Absolute Expression Level Estimation

In the previous section, it is stated that reliable normalization is essential since expression data can significantly vary from different normalization approaches. Even though the use of log-ratios of the measured intensities has its limitation, calculation of the log-ratios is still the main stream in microarray preprocessing approach. And such normalization methods are largely dependent on the satisfaction of the GNA, which have been shown violation more frequently than currently believed [33, 34]. Therefore, novel normalization methods which differ in spirit from previously published normalization strategies are expected.

Engelen *et al.* (2006) [35] proposed a different way of normalizing cDNA microarray data that avoids the GNA and poses advantages over log-ratio based approaches. This normalization approach is based on a physically motivated model. The model consists of the following two components:

- a hybridization reaction, which models the hybridization of transcript targets to their corresponding DNA probes in form of

  $$\frac{x_s}{x_0(s_0 - x_s)} = K_A$$

  where x_s represents the amount of hybridized target; x_0 represents the concentration of the corresponding transcript in the hybridization solution; K_A is the hybridization constant and s_0 is the spot size or the maximal amount of available probe.

- a dye saturation function, which the relation between the measured fluorescence intensities and the amount of hybridized, labelled target in form of

  $$y = p_1 x_s e^{\varepsilon_m} + p_2 + \varepsilon_a$$

where y represents the measured intensities and models as a linear equation incorporating an additive and multiplicative intensity error, respectively represented by $\varepsilon_a \sim N(0, \sigma_a)$ and $\varepsilon_m \sim N(0, \sigma_m)$.

The parameters in this calibration model are estimated by the known concentration in the hybridization solution and measured intensity of external control spike. As mentioned in Section 2.2.1.1, spikes are genuine calibration points and can be served for quality control and normalization. These estimated parameters can then be used to obtain absolute expression levels for every gene in each of the test biological conditions. More details see Engelen *et al*. (2006) [35].

To increase this method's usability and accessibility, we implemented it as a user-friendly BioConductor package, CALIB. Details about the package are presented in Chapter 3 of this dissertation.

2.2.3 Postprocessing of Microarray Data

After microarray data have been normalized, they can be explored in order to extract biologically meaningful results. The biological questions to be addressed can be quite diverse. With the lack of uniform strategies, numerous techniques and algorithms from statistics, data mining and machine learning have found their ways into the microarray data analysis field. Results from such analyses have been fruitful and have provided powerful tools for studying the mechanism of gene interactions, gene regulations and other studies. This section lists some of popular methods among them.

Identification of differential gene expression

A microarray experiment measures the expression levels from thousands of genes in parallel. Genes that show little or no change in expression levels are typically of no biological relevance. Therefore, a detection of the genes shows a variable expression between two tested biological samples is often a crucial step in the analysis of any microarray experiment. Many methods are proposed to identify the significantly differentially expressed genes, most of which attempt to fit a model to estimate the relative gene expression and the error terms. Details of some of these methods are demonstrated and discussed through a case study in Chapter 4 of this dissertation.

Classification

Classification methods can be used to study microarray data in the space of tested biological samples, for example in pharmaceutical or clinical settings: drug discovery, disease management, toxicogenomics, etc. A typical set of microarray experiments of such usually have a very limited number of samples. Namely, the number of gene expressions, which are seen to be the input variables, is substantially large. In order to study the point of interest, a classification process usually has two steps:

- In the first step, the original gene expression data is fed into a dimensionality reduction algorithm, which reduces the number of input variables by either filtering out a larger amount of irrelevant input variables or building a small number of linear or nonlinear combinations from the original set of input variables. The former approach is often known as variable selection while the latter is often known as feature selection.

- In the second step, methods for class discovery or class prediction can be applied on the dimension reduced microarray data set. Class discovery (unsupervised) is to subdivide the tested biological samples into classes based on the characteristic expression. Class prediction (supervised) is to predict the class membership of new samples based on a classifier model trained on the known data set.

Clustering

Clustering is a useful exploratory technique for suggesting resemblances among groups of genes or a group of tested biological samples. It is essentially a grouping technique that aims to find patterns in the data that are not predicted by the experimenter’s current knowledge or pre-conceptions.

Different procedures emphasize different types of similarities, and give different resulting clusters. Most cluster programs offer several distance measures (Euclidean, Manhattan distances), some relational measures (correlation, and sometimes relative distance), and mutual information. Standard clustering techniques, such as hierarchical clustering, K-means, and self-organizing maps, are applied to group together the gene profiles with similar patterns across the conditions [36]. Moreover, advanced algorithms have also been developed which are specifically fine-tuned for biological applications.

Henceforward, we give a detailed introduction of two clustering algorithms: hierarchical clustering and AQBC. They will be used in Chapter 4 of this dissertation.

Hierarchical clustering was first applied in biology for the construction of phylogenetic trees. Early applications of the method to gene expression data analysis [37, 38] had proved its usefulness. Hierarchical clustering has almost become the *de facto* standard for gene expression data analysis, probably because of its intuitive presentation. The whole clustering process is presented as a tree called a dendrogram. The original data are often reorganized in a heatmap demonstrating the relationships between the genes or the conditions.

Two approaches to hierarchical clustering are possible:

- Divisive clustering (a top-down approach as is used in Alon (1999) [39]): In divisive clustering, all the gene expressional profiles are treated as belonging to one cluster; in each step, a cluster is divided so that the resulting clusters are as far away from each other as possible. Different techniques for dividing the clusters are available for divisive hierarchical clustering [40, 41].

- Agglomerative clustering (a bottom-up approach, see for example in Eisen 1998 [37]): In agglomerative clustering, each expression profile is initially assigned to one cluster; in each step, the distance between every pair of clusters is calculated and the pair of clusters with the minimum distance is merged; the procedure is carried on iteratively until a single cluster containing all the expression profiles is obtained.

After the dendrogram is obtained, the determination of the final clusters is achieved by cutting the tree at a certain level, which is similar to putting a threshold on the pair wise distance between clusters.

As we know that hierarchical clustering in general have several drawbacks. It can never repair a decision (to split in the divisive one or to merge in the agglomerative one) made in previous steps [41]. It is, after all, based on a stepwise optimization procedure rather than looking for *k* optimal clusters globally. Also, the decision of the final cluster partition is rather arbitrary. Another disadvantage of hierarchical clustering is that the algorithm obliges every gene to belong to a cluster. However, in general, a considerable number of genes included in the microarray experiments do not contribute to the studied biological process. Including these “noisy” genes or spurious expression profiles in one of the cluster will contaminate the content of the cluster and make the cluster less suitable for further analysis.

Previously mentioned disadvantages are not only of the hierarchical clustering but also often of most of the first generation clustering algorithms, such as k-means [40] and self-organizing maps (SOM) [42], while Adaptive quality-based clustering method (AQBC) [43] is one of the algorithms which can overcome these disadvantages. It is a heuristic, two-step approach that defines the clusters sequentially. The first step locates a cluster centre (quality-based approach) and the second step derives the quality of this cluster from the data (adaptive approach). The number of clusters is not known in advance, so it is not a parameter of the algorithm. The algorithm decides the number of clusters automatically in cooperate with two user-defined parameters: the minimal number of genes in a cluster and the least similarity between an expression profile and the cluster centre it supposes to belong to. So the number of clusters is not decided arbitrarily anymore. The algorithm only accepts valid clusters and allows a certain gene does not belong to any of the clusters. Therefore, the resulting clusters don't contain many noisy gene expression profiles anymore. This will help further biological interpretation of the clusters. This method is

2.3 High-throughput Microarray Compendia

As previously mentioned in Section 2.2, microarray technology has become an indispensable technology for studying large scale transcriptional gene expression in genomics research. Increased accessibility, lowered cost and improved technology result in more comprehensive studies, under more diverse and larger sets of conditions and a rapid expansion of available gene expression data. Mining from the integrated large scale microarray data offers the molecular biologists the possibility to view their own small scale analysis in the light of what is already available and also the opportunities towards studying in network and pathway perspective. A first step towards integration the large scale microarray data from different laboratories in different conditions is to generate microarray compendia. The rest of this section gives an introduction of microarray compendia, preprocessing and standardization of different experiments in a microarray compendium, the available subsequent analysis on a microarray compendium.

2.3.1 Microarray Compendia

In the early days of microarray technology, small-scale experiments were performed containing only few hybridizations. Soon the scale of the experiments increased dramatically. Nowadays, transcriptome is becoming viewed as a separate biological entity. Unlike the genome, which is roughly fixed for a given cell line, the transcriptome can be specific to a certain cell and varied external environmental conditions. Similarly as for the genome, collections of transcriptomes are being gathered from many species and called gene expression compendia or microarray compendia.

One of the first data sets can be regarded as a compendium was produced by Hughes et al. in 2000 [44]. This compendium contains 300 cDNA microarrays in *Saccharomyces cerevisae*. Using these microarrays, 300 expression profiles were generated in which transcript levels of a mutant or compound-treated culture were compared to that of a wild-type or mock-treated culture. In their paper the authors clearly put forward the benefits of applying a compendium approach to study gene expression and make advances in functional genomics. Microarrays measuring whole genome expression levels can provide molecular phenotypes of cell even for conditions for which no conventional phenotypes exist. This circumvents the dependence of easily scored phenotypes when studying e.g. mutants. As stated in Hughes et al, a fundamental advantage of the compendium approach over the conventional assays available at that time was thus that it substitutes a single genome-wide expression profile in place of many conventional assays that measures only a single cellular parameter [44]. Furthermore, the same compendium used to characterize mutants can also be used to characterize other perturbations, such as treatments with pharmaceutical compounds. Soon after this first compendium, many efforts were started to generate compendia of various ranges and types [45-47].

Currently, public databases, such as GEO [3] or ArrayExpress [5] offer a central repository of MIAME-compliant [48] microarray data. Although these databases are an extremely rich source of information, containing thousands of experimental data sets for a particular model organism, they do not directly allow for an integrated exploration of the data between experiments. An additional conversion step is needed: to derive compendia by combine all the experiments on one particular organism from the public resources.

Presently, microarrays can be manufactured in different ways. For both spotted arrays and in situ synthesized arrays, there are many subcategories of either type on the market, for example, cDNA microarrays, Affymetrix, Agilent, Code-link and in-house microarrays [49]. As a consequence, two main types of compendium exist: single platform compendia and cross platform compendia [50]

- Single platform compendia: They combine all data on a particular organism that were obtained from one specific microarray platform. Most of the single platform compendia focus on Affymetrix as it turned out to be one of the more robust and reproducible platforms [51, 52]. For example, Genevestigator [53] initially developed for *Arabidopsis*, but now being extended to other species such as human and mouse, and M3D [45], which offers Affymetrix based compendia for three microbial organisms (*E.coli*, yeast and *Shewanella oneidensis*). Such single platform compendia are more straightforward to be normalized and used for subsequent analysis.

- Cross platform compendia: They include data from different platforms and often combine data from both one and two channel microarrays. These compendia are topic-specific, collecting all the publicly available experimental information related to the topic of interest. ITTACA [54] and ONCOMINE [55], for instance, focus on cancer in human, GAN [56] on aging in several species. Because of the heterogeneity in platforms, more attention has to be paid when applying normalization and subsequent analysis on them.

2.3.2 Preprocessing of a Microarray Compendium

When integrating microarray experiments into a compendium, comparability is the most essential issue to be considered. Two main aspects impede the comparability mostly:

- Firstly, microarrays may be generated on a range of platforms, for example, cDNA microarray, Affymetrix, Agilent, Code-link or sometimes even in-house microarrays. Each platform requires its own optimized sample preparation, labelling, hybridization and scanning protocol, and concomitantly also a specific normalization procedure.

- Secondly, despite of the heterogeneity in platforms, microarrays may come from different laboratories. There is also great diversity in the protocols used by different laboratories for RNA preparation and labelling, as well as in the instrumentation and software used for these procedures.

Therefore, the comparability of microarray data between laboratories and across platforms obliges careful preprocessing approach applied on the compendium.

Considering single platform compendia, on one hand, the best agreement between results of different experimental setups could be reached when different laboratories used standardized protocols for both experimental work and data analysis [51]. On the other hand, using alternative preprocessing approaches instead of the default methods offered by the manufactures increases the comparability of the results [52]. For instance, mentioning an Affy-based single platform compendium, within-array normalization methods such as MAS 5.0, Li & Wang method and RMA (Section 2.2.2.3 and 2.2.2.5) can be chosen as the optimized preprocessing approach to remove the probe effect within one Affymetrix and subsequently, between-array normalization methods including linear normalization and quantile normalization (Section 2.2.2.5) should be applied. They have the similar principle which is to make different array distributions to be comparable. In detail, linear normalization focuses on forcing different array distribution to have the same central tendency while quantile normalization is to impose the same empirical distribution of intensities to each array. Either of them serves as a between-array normalization method which aims at removing the array effect across all the arrays. As such, these normalization methods control between array variability and facilitate comparison and integration across different microarrays.

Obviously, data obtained from microarrays performed on different experimental platforms are usually less comparable than when the same platform is used [52]. Normalization methods, especially for between-array normalization, are not as straightforward as on single platform compendia. Suppose that a cross platform compendium combines data from both one and two channel microarrays. Different optimized within-array normalization should be chosen according to the different platforms. For two channel microarrays, as described in Section 2.2.2.5, global normalization, intensity dependent linear normalization and intensity dependent nonlinear normalization exist. Among these normalization methods, LOWESS is one of the most widely used methods. Two channel microarrays are designed for comparing expression levels between two biological samples. These comparisons are typically expressed for each gene as the ratio of the intensities measured between each dye-labelled sample. Therefore, the normalized microarray data by the above normalization methods are all in the format of logarithm transformed intensity ratios between two channels. While considering the normalization methods for one channel microarray, MAS 5.0, Li & Wang method and RMA come into widespread use. In the final step of these normalization procedures, the data measured by different probe sets are summarized to produce one measure for the expression level of each gene on each array. Thus, in order to exploit a cross platform compendium as an integrated data set, the log ratios from two channel microarrays and the absolute expression levels from one channel microarrays have to be converted into one uniform format. The most straightforward way towards the uniformity is to calculate the log ratios from the absolute expression levels using a carefully chosen reference. The reference can be a pooled reference from a set of one channel microarrays or one of the arrays in a set of one channel microarrays. The homogeneity in data format definitely benefits the subsequent analysis of a compendium.

Apart from choosing platform-specific within-array normalization methods, between-array normalization is also essential when generating cross platform compendia. For instance, one condition may change the expression of many genes by a very large factor while another condition affects mainly the same genes but with only a small amount. Although, these two conditions are related, this relation is not explicit in the expression data of the compendium without between-array normalization. Moreover, from the microarray technology point of view, between-array normalization aims to remove individual array variations to guarantee the comparability of the arrays within a compendium. As an example, scale normalization is a simple scaling of the data from a series of arrays so that each array has the same median absolute deviation. Consequently, array variability is controlled and comparison and integration across different microarrays is made possible.

In summary, the preprocessing procedure for a microarray compendium is necessary to enhance the comparability of the compendium. The procedure includes choosing optimized platform-specific within-array normalization methods and applying between-array normalization methods on the compendium. In case of cross platform compendia, more attention needs to be drawn. Such as all the data have to be converted into a uniform format and the reference used for the conversion has to be chosen carefully. Choosing different references for different experiment sets also complicates the way to normalize between arrays. For instance, the compendium can be scaled across all the arrays, across the arrays with the same reference, or across the arrays from the laboratories and so on. Whether these different normalization methods have an influence on different subsequent analysis will be studied in Chapter 6 of this dissertation.

2.3.3 Postprocessing of a Microarray Compendium

Once a microarray compendium is generated, the information contained in the compendium is enormously rich. Extracting comprehensive knowledge out of the compendium is then the next issue. All the methods applicable on an individual microarray or a relatively small set of microarrays are also valid on a microarray compendium. As mentioned in Section 2.2.3, these methods fall in several main categories: identification of differential gene expression, classification and clustering. Single-platform compendia are more straightforward to use these methods directly. Namely, the methods are applied to a compendium as a whole. Due to the heterogeneity of cross platform compendia, each set of experiments can be analyzed separately and independent analyses are subsequently combined or compared across all the arrays. This can be seen as the indirect analysis compared to the direct analysis of single platform compendia [50].

However, the results of standard clustering algorithms to genes are sometimes limited when clustering a microarray compendium [57]. The limitation results are due to the existence of numerous experimental conditions in the compendium where the activity of genes in uncorrelated. A similar limitation exists when clustering conditions. For this reason, biclustering algorithms have been proposed to date. In addition to discovering the relationships between the genes or between the conditions as in conventional clustering algorithms, biclustering methods explore the relationships in both the gene dimension and the condition dimension. Biclustering is a technique to cluster both the genes and the conditions at the same time. The general understanding of cellular processes leads us to expect that subsets of genes will only be co-regulated and co-expressed under certain experimental conditions, but behave independently under other conditions. Therefore, biclustering is more applicable and meaningful to deal with large-scale, integrated microarray compendia, in which the number of conditions to be concerned is numerous. We also developed a biclustering algorithm called *Pro*Bic, whose details can be found in Chapter 5 of this dissertation.

Analysis of microarray compendia can also contribute to the inference of regulatory (or genetic) networks [58-61], which is one of the major challenges of the field of interdisciplinary biology.

In a systems biology viewpoint, a cell is considered a system that continuously interacts with its environment. The cell receives dynamically changing environmental cues and transduces these signals into observed behaviour (i.e. change of phenotype or change of physiological response). This signal transduction is mediated by the transcriptional network. A complete transcriptional network can be seen as consisting of proteins interacting with each other, with DNA or with metabolites to constitute a complete signalling pathway [62-65]. Transcriptional network inference became a big research topic as microarray technology made its way into mainstream biological research, but the underdetermined nature of the data made the construction of biologically relevant transcriptional networks a daunting task. With the advent of microarray compendia however, the research in network inference has received a new impulse to obtain biologically plausible results. Many algorithms are available now for inferring transcriptional networks. For instance, learning transcriptional networks from a microarray compendium can either focus on the detection of modules (module inference) or on the detection of a regulatory program (regulatory program inference). An important property of the transcriptional network is its modularity: the network consists of overlapping modules of functionally related genes that all act in concert under certain conditions. Module inference algorithms are useful on their own as they tell us which genes are coexpressed. The biclustering algorithm proposed in Chapter 5 is one of the algorithms that can be used to learn about these modules.

Chapter 3

CALIB: a BioConductor package for estimating absolute expression levels from two-color microarray data

Normalization attempts to adjust individual intensities to remove the effects which arise from variation in the technology rather than from biological difference between tested samples. It is an essential step in microarray data analysis. Previously proposed normalization methods mostly rely on Global Normalization Assumption (GNA), which is proven less appropriate when the expression patterns between two tested biological samples are expected to differ considerably. The usage of log-ratios in most of these normalization methods obscure the essential differences in levels of gene expression levels between genes and are biased according to the choice of the reference sample. Different ways of normalizing cDNA microarray data that avoids the GNA and poses advantages over log-ratios based approaches have been proposed.

In this chapter we introduce the BioConductor package CALIB designed for estimating absolute expression levels from two-color microarray data. We start from the introduction of the package. Next, we elaborate the detailed usage of the package.

3.1 Introduction

Normalization of microarray measurements is an essential in a microarray analysis flow. It aims at removing consistent sources of variation arises from the technician rather than from the tested biological samples. Reliable normalization is crucial because expression data can significantly vary from different normalization approaches. For normalization of two-color microarrays, different methods have been described. Although some approaches inherently work with absolute intensities, for example ANOVA [66], in general, preprocessing of two-color microarrays largely depends on the calculation of the log-ratios of the measured intensities. A common normalization step is the removal of the nonlinear intensity-dependent discrepancy between Cy3 and Cy5 intensities, for example loess [29]. This normalization assumes the distribution of gene expression to be balanced and showing little change between the biological samples tested, an assumption which is referred to as global normalization assumption or GNA. Global mRNA changes that result in an uneven distribution of expression changes, however, have been shown to occur more frequently than currently believed [33, 34] and could have a significant impact on the interpretation of data normalized according to the GNA.

Recently, a different way of normalizing two-color microarray data was proposed that avoids the GNA and poses several advantages over ratio-based approaches [35]. Briefly, the normalization is based on a physically motivated model, explicitly modelling the hybridization of transcript targets to their corresponding DNA probes, and the relation between the measured fluorescence and the amount of hybridized, labelled target. The parameters of this model and incorporated error distributions are estimated from external control spikes: targets that are added to the hybridization solution in known concentrations. This, together with the inherent non-linearity of the model, allows for normalizing the data without making any assumptions on the distribution of gene expression, as opposed to procedures relying on the GNA. More importantly, since the model links target concentration to measured intensity, estimating absolute expression levels of transcript targets in the hybridization solution becomes possible.

To increase this method’s usability and accessibility, we implemented it as a user-friendly BioConductor package [67], CALIB. This package allows normalizing two-color microarray, using the method mentioned above. A spike-based calibration model is used to estimate absolute transcript levels for each combination of a gene and tested biological condition, irrespective of the number of microarray slides or replicate spots on one slide.

3.2 Algorithm Description

As stated in Engelen *et al*. 2006, the principle of the normalization procedure proposed in CALIB package is as follows: intensity measurements of external control spikes serve to estimate the parameters of a calibration model. These parameters can then be used to obtain absolute expression levels fro every gene in each of the tested biological conditions. The calibration model consists of two components, a hybridization reaction and a dye saturation function.

The hybridization reaction describes the relation between the amount of hybridized target (x_s) and the concentration of the corresponding transcript in the hybridization solution (x_0). It is modelled as

$$\frac{x_s}{x_0(s_0 - x_s)} = K_A \tag{3.1}$$

In Equation 3.1, the spot capacity s_0 is assumed to follow a certain distribution around an average spot capacity. Two possible distributions are provided: either the spot capacity is additive $\mu_s : s_0 = \mu_s + \varepsilon_s$ (errormodel ="A"), or the spot capacity error is multiplicative $s_0 = \mu_s e^{\varepsilon_s}$ (errormodel ="M") in both cases with the spot error $\varepsilon_s \sim N(0, \sigma_s)$.

The second component of the calibration model is the dye saturation function, which describes the relationship between the measured intensity (y) and the amount of labelled target (x_s), hybridized to a single spot on the microarray:

$$y = p_1 x_s e^{\varepsilon_m} + p_2 + \varepsilon_a \tag{3.2}$$

This dye saturation function is a simple linear equation incorporating an additive and multiplicative intensity error, respectively represented by $\varepsilon_a \sim N(0, \sigma_a)$ and $\varepsilon_m \sim N(0, \sigma_m)$.

In all, there are three different error distributions that are assumed to influence intensity measurements: additive intensity error ε_a , multiplicative intensity error ε_m and spot capacity error ε_s . The parameters of the saturation function and the variances of the intensity error distributions are considered specific for all measurements of a single array and dye combination. The parameters of the hybridization reaction and variance of the spot error on the other hand apply to all measurements of a single array. Intensities of the external control spikes are used to estimate all these model parameters.

In the next step, the obtained parameter values can be used to estimate the absolute expression level of a single gene in a single biological condition based on all measurements that were obtained for this combination of gene and condition.

3.3 Package Description

Below we give a short description of the data structures adopted by the CALIB package, the implemented normalization approach and the corresponding visualization tools.

3.3.1 Data Structure

In the CALIB package, microarray data are stored in a structure called *RGList_CALIB*, an extension of the *limma::RGList* [68]. The advantage of constructing *RGList_CALIB*, as an extension of the *limma::RGList* enhances its usability for users already familiar with normalization methods such as *normalizeWithinArrays()* in *limma*, allowing for maximal flexibility in using CALIB alongside other packages available within BioConductor. The CALIB microarray data structure *RGList_CALIB* as well as other CALIB specific data structures, such as *SpikeList* (to store spike related data) and *ParameterList* (to store estimated calibration parameters), are all inherited directly from the R data type *LIST*. Therefore, any method that works on *LIST* will also work on these CALIB defined classes.

3.3.2 Example Data

In order to illustrate the workings and principles of the CALIB method and the usage of the functions in the package, we included a test set containing two out of fourteen hybridizations of a publicly available benchmark data set [69]. The experimental design of these two arrays consists of a color-flip of two conditions. The usage of the package is illustrated in this guide by means of this test example.

3.3.3 Normalization

The normalization method implemented in the CALIB package consists of two major steps: (1) estimation of the calibration parameters: *estimateParameter()* and (2) normalizing the data accordingly: *normalizeData()*.

The function *estimateParameter()* takes the objects of *RGList_CALIB* and *SpikeList* as input arguments and uses spike intensity measurements and corresponding concentrations to estimate the parameters of the calibration models. Since each array has a different set of parameters, the estimation is performed separately for each array. The output of this function is an object of the *ParameterList* class, containing all the model parameters (representing the intensity saturation characteristics of both dyes, and the hybridization of target to complementary probes) of the pecified arrays.

The function *normalizeData()* takes the raw microarray data *(RGList_CALIB* object) and the estimated model parameters (*ParameterList* object) as input arguments. The parameters of the spike-based calibration model are used to estimate the absolute expression levels for each combination of a gene and condition in the design of the microarray experiment, irrespective of the number of microarray slides or replicate spots on one slide. When required, this function can be used on individual genes, or on a selected group of genes instead of on the entire gene set.

These functions are written in C++. A dynamic link library, compiled from the C++ code is used as a plug-in for the R wrapper functions.

3.3.4 Visualization

The CALIB package provides different visualization functions that allow quality control and data exploration. As the estimation of model parameters depends to some extent on the quality of the external control spikes, it is advisable to check the quality of the external controls and the estimated parameters prior to running the normalization function by using these visualization functions. Additional functions are also provided to view the final results of the normalization procedure.

The functions *plotSpikeCI()* and *plotSpikeRG()* can be used to check the quality of external controls before running the estimation function as well as to evaluate the model fit after parameter estimation. *plotSpikeCI()* compares measured intensities to actual spike concentrations. *plotSpikeRG()* compares Cy3 and Cy5 measurements from the same spot. Other functions can only be called after estimating the model parameters: *plotSpikeHI()* plots the estimated amount of hybridized target against corresponding intensities of calibration controls (i.e. with a 1:1 ratio) and serves to evaluate the calibration model, and *plotSpikeSpotError()* plots the distribution of estimated spot capacity errors for all spikes of an array (as a histogram, box plot or density plot).

The functions *plotSpikeACEC()* and *plotNormalizedData()* can be called after data normalization. *plotSpikeACEC()* compares the actual concentrations of calibration controls (i.e. with a 1:1 ratio) with the estimated concentrations. Ratio controls are not included by default, as they do not necessarily reflect the design of the experiment. *plotNormalizedData()* allows comparing the estimated expression levels of two selected conditions.

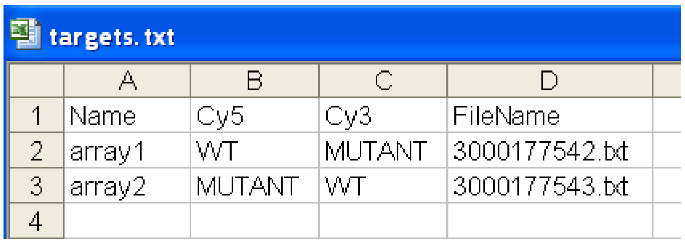
3.4 Package Usage

3.4.1 Reading Data

3.4.1.1 Reading two-color microarray data

Since the *RGList_CALIB* used to store microarray data in the CALIB package is an extension of the *limma::RGList*, the functions used for reading raw data into an *RGList* object in *limma* are also applicable in the CALIB package.

Following the steps in *limma*, at first a target file needs to be created. It should be a tab-delimited text file with the default name “targets.txt”, and basically describe the experiment design, listing for each array, its name, the file where the data corresponding to this array can be found and the condition/dye combinations measured on this array. Figure 3.1 gives an example of the target file.



	A	B	C	D
1	Name	Cy5	Cy3	FileName
2	array1	WT	MUTANT	3000177542.txt
3	array2	MUTANT	WT	3000177543.txt
4				

Figure 3.1: An example of the target file “targets.txt” describing the experiment design. It contains the following information for each array: name, the file where the data corresponding to this array can be found and the condition/dye combinations measured on this array.

The following command is used to read the target file into R.

```
> targets <- readTargets()
```

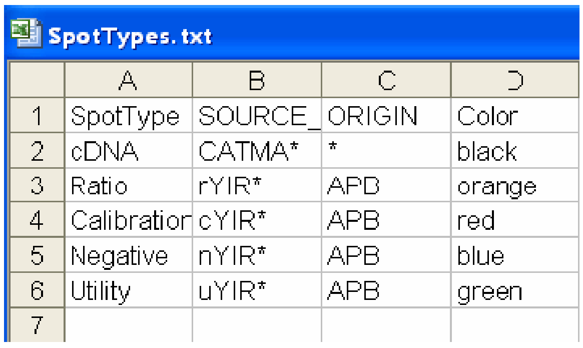
The *targets* object contains a column labelled “FileName”. The entries in this column correspond for each array to the name of the image analysis output file. This column is used as input argument of the function to read in the intensity data:

```
> RG <- read.rg(targets$FileName, source=”<imageanalysisprogram>”,path=”<filedirectory>”)
```

where <imageanalysisprogram> is the name of the image analysis program used to generate the raw datafile and <filedirectory> is the full path of the directory containing the raw data files. At present time, the CALIB package supports three image analysis programs: “genepix”, “genepix.median” and “quantarray”. *RG* is an *RGList_CALIB* object containing the imported microaray data.

After importing the raw data into R, a specific spot type will be assigned to each of the different spots on the array (or different rows of the *RG*). This is done by first reading in the spot file (an example is given in Figure 3.2) which specifies the different user specified spot types (descriptive expression used to make a distinction the regular cDNA probes and the different types of control spots (ratio, calibration,…)) and has the default name “SpotType.txt” and secondly, by setting the status of each spot on the array to one of the user specified spot types. This allows discriminating in the *RG* between regular cDNA spots, as opposed to control spots from which the model parameters will be estimated. Note that while these steps are optional in *limma* package, they are required in the CALIB package.

The spot type file uses simplified regular expressions to match patterns. For example, AA* means any string starting with AA, AA. means AA followed by exactly one other character and AA\. means AA followed by a period and no other characters. The status of the calibration controls, the ratio controls and the negative controls should be Calibration, Ratio and Negative respectively.



	A	B	C	D
1	SpotType	SOURCE_	ORIGIN	Color
2	cDNA	CATMA*	*	black
3	Ratio	rYIR*	APD	orange
4	Calibratior	cYIR*	APB	red
5	Negative	nYIR*	APB	blue
6	Utility	uYIR*	APB	green
7				

Figure 3.2: An Example of spot type file "SpotType.txt". In this file, the simplified regular expressions represented different spot types are listed.

The function used to read user-specified spot type files is *readSpotType()*. The function used to control the status of each spot on the array is *controlStatus()*. They are used as follows:

```
> spottypes <- readSpotType()

> RG$genes$Status <- controlStatus(spottypes, RG)
```

The function *readSpotType()* takes the spot type file name as input argument. In the example, default name "SpotType.txt" is given to the spot type file. Therefore, *readSpotType()* just takes the default argument.

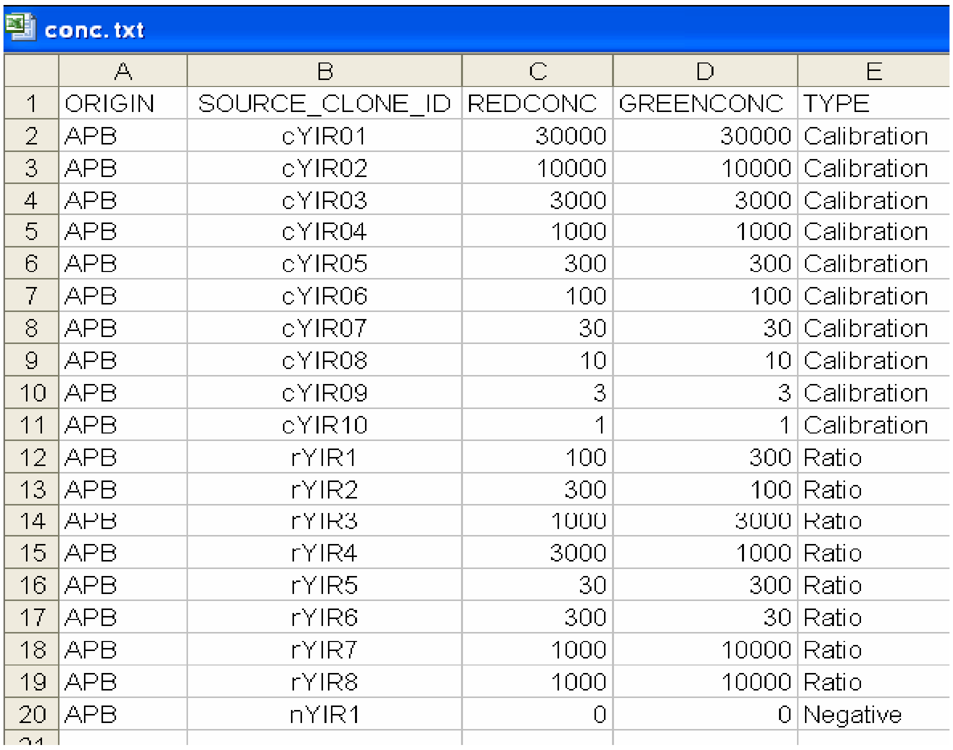
For details regarding the usage of these functions and the format of the target file and the spot type file and for some more optional functions, we refer to the *limma* user's guide in the *limma* package.

3.4.1.2 Reading spike data

Spike data are stored in a *SpikeList* object. The data in the *SpikeList* object correspond to a subset of the data stored in the *RGList_CALIB* object, i.e. to these data that correspond to the measurements of the externally added control spikes as specified by their user defined spot status.

In addition, the *SpikeList* object contains two more fields: *RConc* and *GConc*. The entries in these fields correspond to the absolute concentrations of labeled mRNAs for each of the control spikes. *RConc* and *GConc* are set by reading information from a user-specified concentration file.

An example of a concentration file is given below in Figure 3.3: it is a tab-delimited text file in which each row corresponds to a specific control spike, identified by the rows' entry in an identifier column, in this case with the name "SOURCE_CLONE_ID". This name of the identifier column should match the name of a column in the *RG$genes* (where *RG* is an *RGList_CALIB* object) of which the entries contain patterns or regular expressions sufficient to uniquely identify each different spike. The entries of each row in the columns REDCONC and GREENCONC by default contain the absolute concentrations of the spike, labelled with respectively the red and the green dye added to the hybridization solution. For each row, the entry in the TYPE column indicates the type of the particular spike. The names used to indicate spike types in the TYPE column should match the names used to indicate the spike status (or spot type) in the *RGList_CALIB* object.


conc.txt

	A	B	C	D	E
1	ORIGIN	SOURCE_CLONE_ID	REDCONC	GREENCONC	TYPE
2	APB	cYIR01	30000	30000	Calibration
3	APB	cYIR02	10000	10000	Calibration
4	APB	cYIR03	3000	3000	Calibration
5	APB	cYIR04	1000	1000	Calibration
6	APB	cYIR05	300	300	Calibration
7	APB	cYIR06	100	100	Calibration
8	APB	cYIR07	30	30	Calibration
9	APB	cYIR08	10	10	Calibration
10	APB	cYIR09	3	3	Calibration
11	APB	cYIR10	1	1	Calibration
12	APB	rYIR1	100	300	Ratio
13	APB	rYIR2	300	100	Ratio
14	APB	rYIR3	1000	3000	Ratio
15	APB	rYIR4	3000	1000	Ratio
16	APB	rYIR5	30	300	Ratio
17	APB	rYIR6	300	30	Ratio
18	APB	rYIR7	1000	10000	Ratio
19	APB	rYIR8	1000	10000	Ratio
20	APB	nYIR1	0	0	Negative

Figure 3.3: An example of concentration file "conc.txt". In this file, different spot types are listed together with their red and green concentrations.

In this example, two identifier columns ORIGIN and SOURCE_CLONE_ID are used.

For a concentration file named conc.txt a *SpikeList* object can be created by

> spike <- read.spike(RG, file = “conc.txt”, path = <filedirectory>)

When the concentration column names differ from the default names “REDCONC” and “GREENCONC” (for example rconc and gconc) the function can be called as follows:

> spike <- read.spike(RG, file = “conc.txt”, conccol = list (RConc = “rconc”, GConc = “gconc”), path = <filedirectory>)

In both previous examples, we assumed that the same spike set (e.g. a commercial kit) was hybridized to all arrays. When this is not the case, for instance, when the same spike set is added in different concentrations on the different arrays, or when different dye configurations are used for the same spike set hybridized to the different arrays (e.g. an experiment where ratio spikes are ‘dye-swapped’ from one array to another), the user has to specify a separate concentration file for each array. For example, if we have two different arrays with different sets of spikes hybridized to them, a single concentration file is defined for each array; say “conc1.txt” and “conc2.txt”. In case the default concentration column names are used, the *read.spike()* function should be called as follows:

> spike <- read.spike(RG, file = c(“conc1.txt”, “conc2.txt”), different = TRUE, path = <filedirectory>)

where conc1.txt and conc2.txt correspond to the concentration files of the first and second array in the RG respectively.

In all previous examples, a *SpikeList* called *spike* is created, containing the spike data of all the arrays.

3.4.2 Quality Control

In the CALIB package, calibration model parameters are estimated from external control spikes. The final normalization thus depends on the quality of spotted spikes on the arrays. Checking the spike quality prior to estimating the model parameters is definitely advisable.

Two functions to visually inspect the quality of the spikes are provided i.e., *plotSpikeCI()* and *plotSpikeRG()*. In the following, we will use the example data set provided by the package to explain how to create plots with these functions.

The function *plotSpikeCI()* plots for one specified array the known concentration of the spikes (corresponding to the amount of externally added labelled cDNA of each spike) against their measured intensity and this for both the red and green channel.

```
> plotSpikeCI(spike)
```

An additional argument array of the function *plotSpikeCI()* takes the first array as default value. Therefore, this function gives the plot (shown in Figure 3.4) of the first array of *spike*. If the second array needs to be plotted, the argument *array* has to be specified.

```
> arraynum<- 2

> plotSpikeCI(spike, array=arraynum)
```

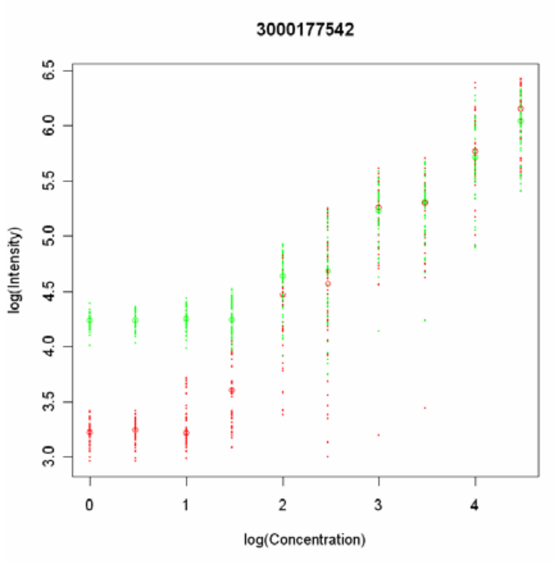


Figure 3.4: A non-linear relationship between measured intensities and corresponding concentrations for all external control spikes on array 3000177542. The red and green spots representing the external control spikes labelled with red dye and green dye respectively.

Figure 3.4 shows a sigmoidal relationship between the measured intensities and added concentrations is to be expected. Indeed, in a certain range the relationship will be linear, but at the highest and lowest concentration levels saturation effects will occur, which might be different for the red and green channel.

The function *plotSpikeRG()* plots for all spikes on a specified array the red versus the green intensities. For example, if we want to plot the red versus green intensities of the first array, the function is used as follows:

```
> plotSpikeRG(spike)
```

The higher intensity levels in Figure 3.5 give an idea of the multiplicative error variance on the measured intensities.

3.4.3 Estimation of model parameters

Once the quality of the spikes is assessed, spike measurements can be used to estimate the parameters of the calibration model. This model describes the relationship between measured intensities and target levels added to the hybridization solution. The function *estimateParameter()* is used to estimate the parameters of the calibration models based on the measured intensities and known concentrations of the external control spikes. It takes the *RGList_CALIB* and *SpikeList* objects as input arguments and estimates a set of parameters for each array. The function can be called as follows

```
> parameter <- estimateParameter (RG, spike)
```

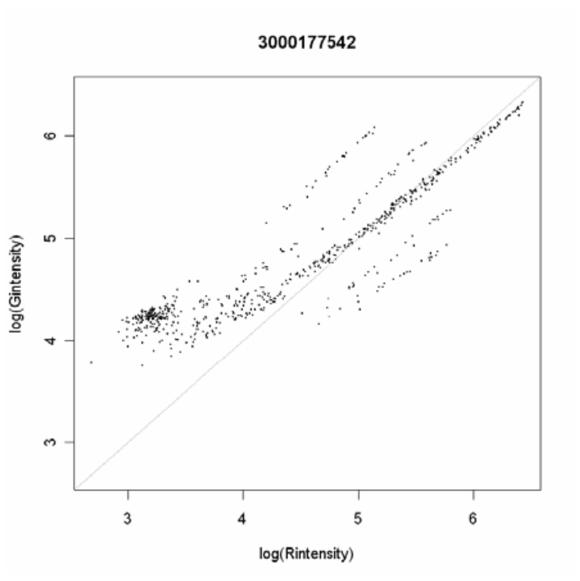


Figure 3.5: Red intensities versus green intensities of all external control spikes (Red/Green ratios 1:10, 1:3, 1:1, 3:1, 10:1).

Besides the required inputs *RGList_CALIB* object and *SpikeList* object, there are three optional input arguments for this function.

- *bc* and *area* (both logical values) are used to specify how intensity values per probe need to be calculated: *bc* indicates whether a background correction is used, *area* indicates whether the foreground intensity will be multiplied with the spot area The default value of these two arguments are *bc* = FALSE (no background correction) and *area* = TRUE (multiplication performed).

- *errormodel* specifies the type of spot error distribution used to model the spot sizes.

If all arguments are given by the user, *estimateParameter()* can be used like

> parameter <- estimateParameter(RG, spike, bc = FALSE, area = TRUE, errormodel = “M”)

The output of this function is an object of *ParameterList* (named *parameter* in the above examples), containing for each array the model parameters and the user specified input arguments (the bc and area values and the type of errormodel).

Results can be accessed by,

> show (parameter)

or

> summary (parameter)

which allow viewing the compacted print-out and the summary of the *parameter* object.

After the parameter estimation, an option is provided to adjust the estimated lower saturation level (indicated by P2 in the parameter list) for either one of both channels. This is needed in these cases, where the lower saturation limit for the spike measurements is seen significantly below the saturation for other data points. This is illustrated in Figure 3.6, where the lower green intensity levels for the spikes (black dots) are generally below those of the other data points (grey dots; the blue curve represents the estimated parameters), indicating that the estimated parameter P2 for the green channel (based on the spikes) is not ideal for normalizing the other data points.

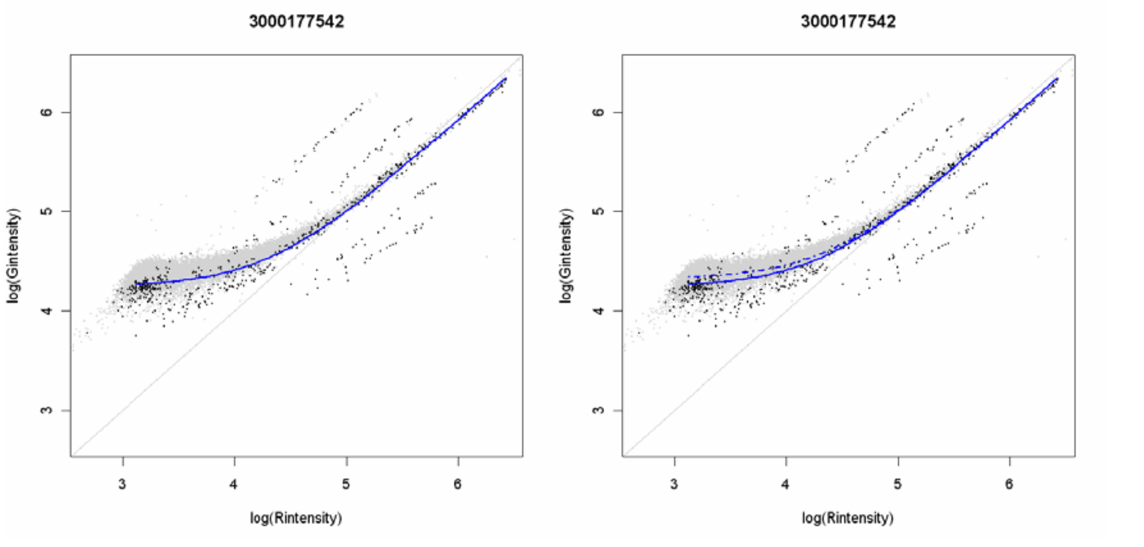


Figure 3.6: Estimated model before (left) and after (right) adjustment of P2 parameter for Cy3. The grey dots indicate the intensities of all the spots on the array. The black dots indicate intensities of the external spike controls. The blue solid line indicates the model estimated from the spikes. The fact that this line does not lie in the centre of the grey dots requires adjustment of P2 parameter. The dot line on the right panel figure shows the P2 adjusted model.

The cause for this discrepancy in lower intensity level between the spike intensities and the intensities of the other data points is not known. A possible explanation might be that fractions of degraded target more or less specifically bind to their corresponding probes raising the background. Industrially synthesized spikes are purer (less degradation), so the effect would be limited to the measurements of the actual gene probes. A higher incorporation rate of Cy3 labels could explain the difference between both channels (i.e. smaller, degraded targets might still get labeled with Cy3, but not with Cy5).

As describing in Section 3.3.2, the function *plotSpikeRG()* can be used to check the quality of the spikes. It can also be used to evaluate the necessity for an adjustment of P2 values if it is called with different arguments as follows:

```
> plotSpikeRG(spike, parameter, RG)
```

This function will give plot shown in Figure 3.6. In the example, Figure 3.6(left) indicates that adjustment of Cy3 is necessary.

When deemed necessary, function *adjustmentP2()* is available for the adjustment. The user can specify the array and channel for which the adjustment will be performed by giving the arguments *arrayindex* and *colorindex* respectively. For the *colorindex*, 1 means red and 2 means green. For example,

```
> parameter -> adjustP2(RG, parameter, arrayindex = c(1,2), colorindex = c(1,2))
```

means that the arrays with index 1 and 2 (according to the data in RG) should be adjusted. For array 1, P2 of the red channel should be adjusted, while for array 2 P2 of the green channel should be adjusted. The *ParameterList* object *parameter* gets an additional field called *AdjustFactor* for the adjustment factor of P2. After adjustment, the evaluation function can also be called to check the result of adjustment. For example,

```
> plotSpikeRG(spike, parameter, RG)
```

Figure 3.6(right) gives the plot created by this function. Figure 3.6 shows that after adjustment the estimated model indicated by dotted line on the plot fits the data better than without adjustment.

3.4.4 Normalization

Once the calibration curves for the red and green channels have been estimated for each array, they can be used to normalize the data. The function *normalizeData()* is used to this and estimates absolute expression levels for each combination of a gene and condition in the experiment design, regardless of the number of replicates. This is conceptually shown for a loop design of three conditions in Figure 3.7.

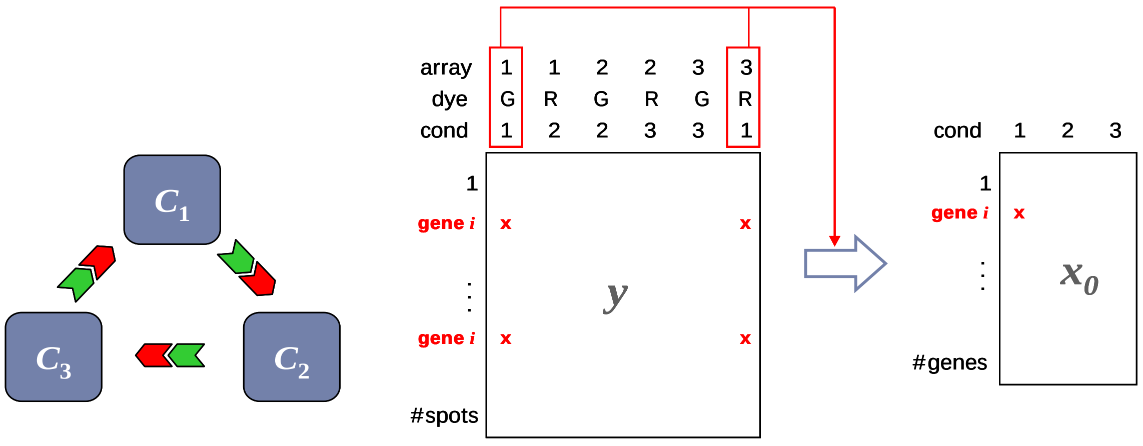
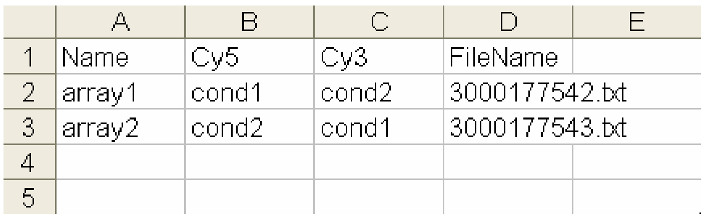


Figure 3.7: Normalization principle. A loop design experiment is used as an example. Left: Experimental design. Three biological samples (C1, C2, C3) are tested using three microarrays. On array 1, C1 and C2 are labelled with green dye and red dye respectively. On array 2, C2 and C3 are labelled with green dye and red dye respectively. On array 3, C3 and C1 are labelled with green dye and red dye respectively. Moreover, each gene is represented by two probes on each array. Right: The absolute expression level (x_0) of gene i is estimated from the measured intensities (y) of all its replicates.

The function *normalizeData()* takes as input arguments the raw data in the form of an *RGList_CALIB* object, and the estimated model parameters in the form of a *ParameterList* object. The settings of the *bc*, *area* and *errormodel*, also needed to perform the normalization are also obtained from the *ParameterList* object.

The design of the array is specified by the input arguments *array*, *condition* and *dye*. Each of them is an integer vector with equal length.



	A	B	C	D	E
1	Name	Cy5	Cy3	FileName	
2	array1	cond1	cond2	3000177542.txt	
3	array2	cond2	cond1	3000177543.txt	
4					
5					

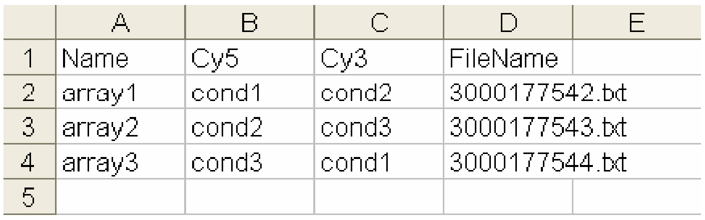
Figure 3.8: An example file indicated a dye-swap design. The file describes the names of the arrays and the conditions measured on these arrays.

For example, for a two array dye-swap design as specified in Figure 3.8, values of *array*, *condition* and *dye* should be given as:

```
> varray <- c(1,1,2,2)

> vcondition <- c(1,2,2,1)

> vdye <- c(1,2,1,2)
```



	A	B	C	D	E
1	Name	Cy5	Cy3	FileName	
2	array1	cond1	cond2	3000177542.txt	
3	array2	cond2	cond3	3000177543.txt	
4	array3	cond3	cond1	3000177544.txt	
5					

Figure 3.9: An example file indicated a loop design. The file describes the names of the arrays and the conditions measured on these arrays.

For or a loop design with three different conditions as specified in Figure 3.9, values of *array*, *condition* and *dye* should be given as:

```
> varray <- c(1,1,2,2,3,3)

> vcondition <- c(1,2,2,3,3,1)

> vdye <- c(1,2,1,2,1,2)
```

From the examples given above, it is clear that:

- Each entry in the vector corresponds to an array channel
- Each entry of *array* indicates on which array this channel was measured. Numbers are duplicated because in a two color design, each array contains two channels.

- Each entry of *condition* gives the numeric representation of the condition measured in this channel

- Each entry in dye indicates the dye used to label this channel. Because we are dealing with two-color arrays, only two numbers are used in vector *dye*. By default 1 represents the red dye and 2 represents the green dye.

The input argument *idcol* specifies the column name of unique identifiers for each probe. It is possible for the same gene to occur multiple times within one array. However, during the calculation, these replicates will be combined and only one normalized value is calculated from these replicates, as indicated in Figure 3.7. The argument is required since different arrays have different annotations. For example, this argument can be specified as follows:

> id_col <- “CLONE_ID”

After specifying all these arguments, the function can be called as follows:

> normdata <- normalizeData(RG, parameter, array = varray, condition = vcondition, dye = vdye, idcol = id_col,)

When the function is called like this, all genes in the *RGList_CALIB* object are normalized. However, since normalization is calculated gene by gene, the normalization can also be performed on individual genes or on a group of interesting genes instead of on the whole gene set. The user can enter the set of selected genes on which normalization has to be performed by another argument of this function – *cloneid*.

For example, if we only want to know the estimated expression level of clone “200001”, we can type

> cloneid_interested <- “200001”

> normdata <- normalizeData(RG, parameter, array = varray, condition = vcondition, dye = vdye, cloneid = cloneid_interested, idcol = id_col,)

Or if we are interested in a group of clones, we can type

> cloneid_interested <- c(“200001”, “200002”, “200003”, “200004”, “200005”)

> normdata <- normalizeData(RG, parameter, array = varray, condition = vcondition, dye = vdye, cloneid = cloneid_interested, idcol = id_col)

In all examples, *normdata* is the output of this function. It is a numeric matrix with rows representing individual genes and columns representing different condition. Namely, every value in this result matrix represents the expression level in a different gene-condition combination (as illustrated in Figure 3.7).

3.4.5 Diagnostics and data visualization

The CALIB package provides different visualization functions that facilitate quality control and data exploration before and after parameter estimation.

The estimation of model parameters is dependent on the quality of the external control spikes. In order to ensure good normalization results, it is advisable to check the quality of the external controls prior to normalization, by using visualization functions described in Section 3.3.2.

Besides checking external control quality, the functions *plotSpikeCI()* and *plotSpikeHI()* can also be used for evaluating the model fit after parameter estimation (if the additional argument “parameter” is added). When including the estimated calibration parameters, both the data and the model fit (red and green curves) are plotted.

The function *plotSpikeCI()* should then be called as follows (where arraynum is the index of the array that will be plotted.):

```
> arraynum <- 1

> plotSpikeCI(spike, parameter, array = arraynum)
```

Figure 3.10 shows the plot created by this function.

The function *plotSpikeHI()* should then be called as follows (again, arraynum is the index of the array that will be plotted.):

```
> arraynum <- 1

> plotSpikeHI(spike, parameter, array = arraynum)
```

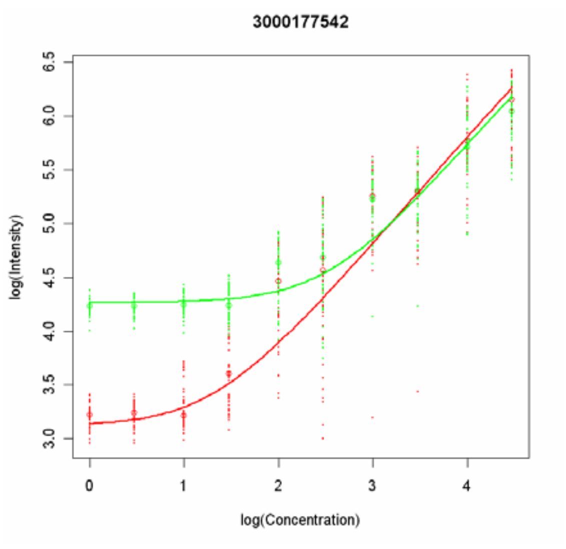


Figure 3.10: Estimated models. The red and green curves represent the estimated calibration models for the red and green channels respectively.

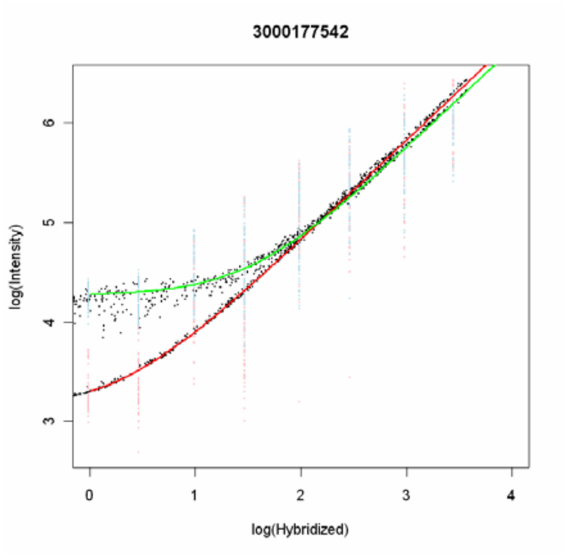


Figure 3.11: Estimated models. The red and green curves represent the estimated calibration models for the red and green channel respectively. Light red and green dots indicate the amount of hybridized target for the measured intensities of the external control spikes if all spot capacities were equal ($\varepsilon_s = 0$). Black dots represent the estimated amount of hybridized target by taking into account the estimated spot capacity errors, i.e. the black dots illustrate how the incorporation of spot errors into the model is an adequate tool for explaining the large variation in measured intensities.

Figure 3.11 shows the plot created by this function, In general, the tighter and smoother (no visible artifacts) the black dots fit the model curves, the more suitable the model is for further normalization.

Note that for both *plotSpikeCI()* and *plotSpikeHI()* the default array index argument *array* is set to 1.

The function *plotSpikeSpotError()* plots the distribution of the estimated spot capacity errors for all spikes of the specified array, and can be used to ascertain the number of outliers or ‘broken’ spots (spots with unusually small spot capacity errors). If there are a substantial amount of these outliers, it is advisable to reestimate the model parameters after omitting these spots from the spike set. Depending on the value of the argument *plottype* (“hist”, “boxplot” and “dens”) this distribution will be plotted as a histogram, a boxplot or a density plot. Resulting plots are shown in Figure 3.12, 3.13 and 3.14.

Function *plotSpikeSpotError()*, can thus be called in the following three ways,

```
> ## plot histogram of the first array, which is the default value of the argument array.

> plotSpikeSpotError(parameter, plottype = "hist")

> ## plot boxplot of both arrays.

> plotSpikeSpotError(parameter, plottype = "boxplot", plotnames = NULL)

> ## plot density function of the first array, which is the default value of the argument array.

> plotSpikeSpotError(parameter, plottype = “dens”, width = 1)
```

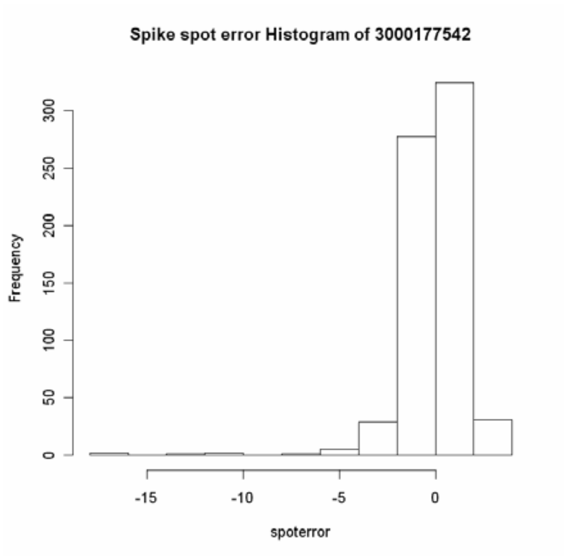


Figure 3.12: Histogram of the estimated spot capacity errors for all spikes.

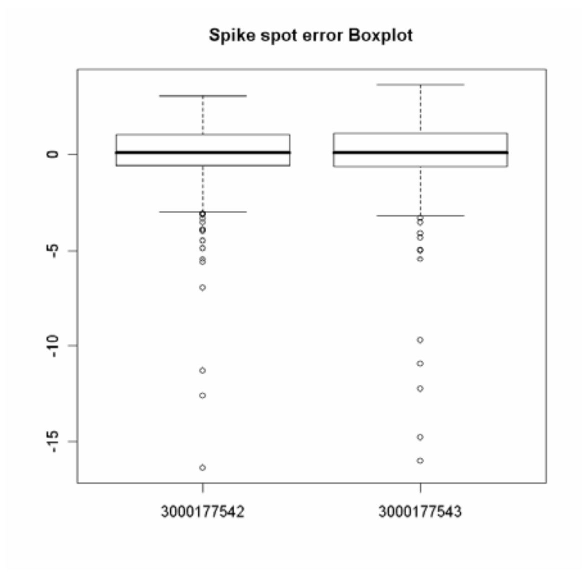


Figure 3.13: Boxplot of the estimated spot capacity errors for all spikes.

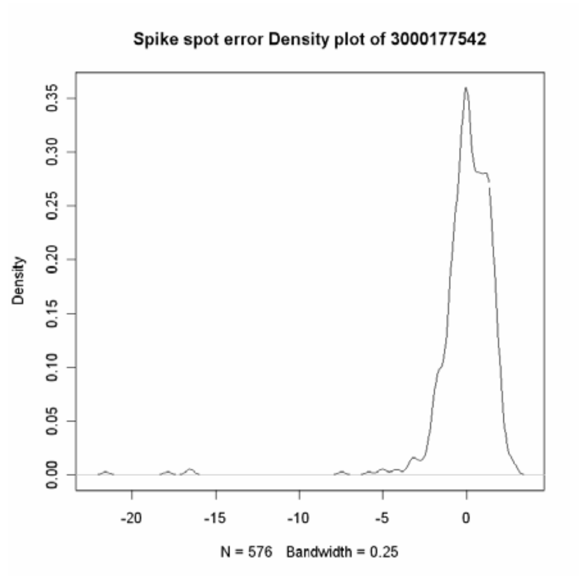


Figure 3.14: Density plot of the estimated spot capacity errors for all spikes.

The function *plotNormalzedData()* can be called after data normalization and allows comparing the estimated expression levels of two selected conditions. It provides an image of the overall similarity between two conditions that were present in the experimental design, i.e. more similar conditions centre more tightly around the bisector, as is the case for the example used in this guide (identical conditions). The function is called as follows:

```
> ## specify the two conditions to be plotted.
> cond <- c(1,2)

> ## use the default values for other parameters.

> plotNormalizedData(normdata,condition = cond)
```

The plot created by this function is shown as Figure 3.15. In this figure, condition 1 and condition 2 in fact represent the same biological sample. The centring of data points around the bisector indicates that typical microarray non-linearities are adequately accounted for.

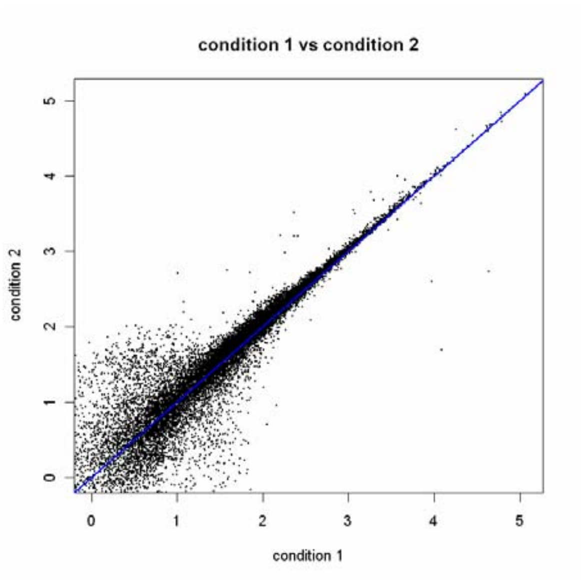


Figure 3.15: Estimated expression levels for condition 1 are plotted against estimated expression levels for condition 3 after normalizing a dye-swap experiment.

3.5 Conclusions

We implemented CALIB, a user-friendly BioConductor package for the normalization of two-color microarray data. The underlying method relies on the presence of external control spikes to estimate the parameters of a calibration model, which are then used to obtain absolute expression levels for all genes. This spike-based normalization procedure provides an alternative solution to the standard ratio-based normalization, and is particularly applicable in cases where, either the GNA is violated and no alternative solution exist, or for applications where absolute expression levels are more convenient than ratios. This normalization procedure has been successfully applied in Smets et al 2008 [70]. Besides the normalization procedure, CALIB provides some convenient visualization tools for quality control of the experimental protocol based on externally added control spikes.

Chapter 4

Microarray Analysis Flow: applied to AI-2 mediated quorum sensing in *Salmonella* Typhimurium

The microarray technology has become one of the most commonly used high-throughput experiments used to study gene expression and transcriptional regulation. Microarray experiments involve three phases: experimental design, measurement and preprocessing, and postprocessing.

In this chapter, we describe the microarray data analysis flow we developed for the analysis of in house generated data. The flow is exemplified by means of dataset generated to study AI-2 mediated quorum sensing in Salmonella Typhimurium. The analysis flow consists of a quality assessment of the generated microarray data, data normalization, the identification of differentially expressed genes and/or clustering of the expression profile and validation of the results by exploiting information in curated public databases.

4.1 Introduction

Microarray technology has been used extensively to survey patterns of gene expression in a range of biological models. The power of microarrays to analyze thousands of genes in parallel increased the speed of experimental progress significantly. Microarrays are used in all fields of biology, for bacteria, plants, animals and humans for a variety of biological questions. Expression profiles can be obtained from diverse developmental stages, under different environmental stress conditions, or in different disease states. The general goal of all these experiments is to find the function, the regulation of the genes and their interaction with other genes. Assessing the function of genes is mainly obtained by making the assumption that the genes which share approximately the same expression patterns are likely to have a similar biological function. Therefore, clustering algorithms, which output a number of clusters, showing genes with a similar behaviour under different conditions, is one of the most important analysis approaches for microarray data. Before analyzing microarray data, preprocessing of these data is an obligatory step because of the variations in technical aspects such as the efficiency of dye incorporation, concentration of DNA on arrays, amount of mRNA, variability in reverse transcription and washing process, among others. Proper normalization is critical for revealing biological results (More detail sees Section 2.2.2).

We carefully studied all the various preprocessing methods and then came to a microarray analysis flow as described by Figure 4.1. This flow consists of four main steps: experimental design, data preprocessing, differential expression detection and clustering analysis. Functions concerning the first three steps and the hierarchical clustering substep are implemented in R (BioConductor), AQBC substep is implemented in Matlab and the cluster annotation analysis is implemented in Perl. All the functions (algorithms) are open sources and user-friendly. This flow has been applied on Salmonella regulatory pathways related to invasion, biofilm formation, mode of action of anti-biofilm molecules and AI-2/LuxS mediated pathway. In this chapter, we will focus on one of these applications: the study of the *Salmonella* Typhimurium AI-2/LuxS mediated pathway.

4.2 Biological background

Infection diseases rank four times in the top 10 causes of death, with diarrhea being one of the main causes. *Salmonella* is one of the main causes of food borne infections. A profound knowledge of the intricate regulatory pathways is required for developing anti-*Salmonella* therapeutics. In this study, we focus on exploring the AI-2/LuxS mediated pathway in *Salmonella* Typhimurium (Figure 4.2). Quorum sensing is a type of decision-making process by decentralized groups to coordinate behaviour. Quorum sensing can regulate a variety of different processes, including virulence and biofilm formation. A variety of different molecules can be used as signals. AI-2 is one of the common classes of signalling molecules, as the AI-2 synthase, LuxS is widely conserved among bacterial species [71]. In *Salmonella* Typhimurium, AI-2 (with DPD as a precursor) is produced and released during exponential growth [72]. AI-2 subsequently is reinternalized by the bacteria via the Lsr (LuxS-regulated) ABC transporter (LsrACDB), encoded by the *lsr* operon. Upon internalization, AI-2 is phosphorylated by LsrK. Phospho-AI-2 then directly binds the repressor LsrR, relieving the repression of the *lsr* operon expression. LsrF and LsrG then are involved in the further processing of the internalized AI-2 [73-75] (Figure 4.2). Until now, besides the *lsr* operon, no other AI-2 regulated genes have been identified in *S*. Typhimurium. Besides as AI-2 synthase, LuxS also plays an important role in the activated methyl cycle [76]. Whether LuxS acts mainly as a metabolic or also as a signal-synthesizing enzyme remains a controversy and how its function is linked to pathogenicity is still unknown. The experiments performed for studying the influence of AI-2 mediated quorum sensing in *Salmonella* Typhimurium (performed by De Keersmaecker et al.) are used as example to illustrate the step by step microarray analysis in this chapter.

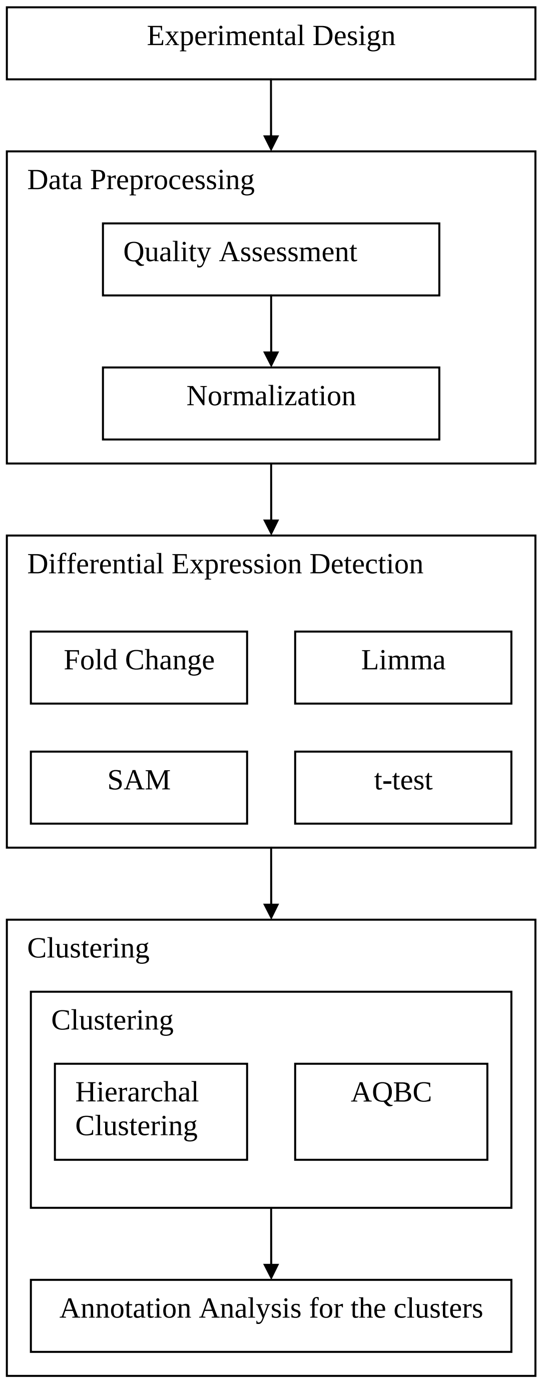


Figure 4.1: Microarray analysis flow.

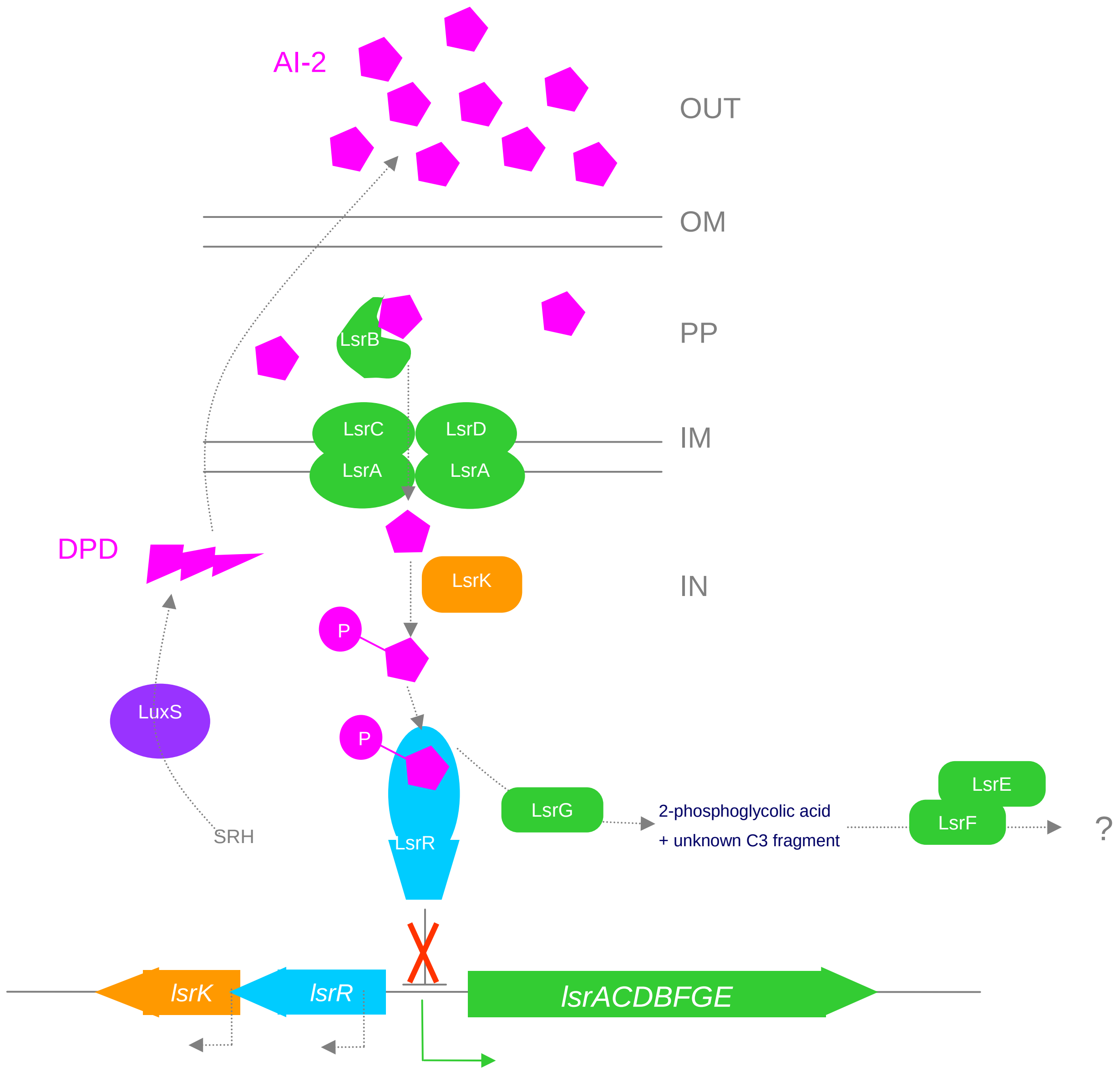


Figure 4.2: AI-2/LuxS mediated pathway in Salmonella Typhimurium. (Figure from S. De Keersmaecker, based on Vendeville et al. 2005 [76].)

4.3 Step by step microarray analysis

4.3.1 Experimental Design

As discussed in section 2.2.2.1, experimental design and replication are essential for identifying and reducing the technical variations in cDNA microarray. Considering the aim of this study and the replications needed for follow-up analysis, seven knockouts were tested: the *lsrG* mutant, the *luxS* mutant, the *luxS*::Km mutant, the *lsrK* mutant, the *lsrFGE* mutant, the *lsrR* mutant and *lsrB* mutant. For each knock out, a dye-swap design was used (2 arrays in total, measuring each sample twice labelled with Cy3 and cy5 respectively). Moreover, some dye-swap experiments of certain knock outs were performed more than once in the same condition.

Table 4.1 All experiments for the study AI-2 mediated quorum sensing in Salmonella Typhimurium.

Date	**Experiment**	**Slide number**	**Conditions**	**Quality (Reason)**	**Chosen for integration (Reason)**
Oct 05	Wt vs lsrG 6h	2004072916 17	Bioreactor – LB – 30% DO – galactose 10 g/l	ok bad (low intensity)	no (no replicate available) no (bad quality)
		2004072931 32		bad (low intensity) bad (low intensity)	no (bad quality) no (bad quality)
	Wt vs lsrG 12h	2004072933 34	Bioreactor – LB – 30% DO – galactose 10 g/l	bad (stationary phrase) bad (stationary phrase)	no (bad quality) no (bad quality)
	Wt vs luxS	2004072935 47	Biofilm– 1/20 TSB – 16°C – 24h	bad () bad (biofilm condition)	no (bad quality) no (bad quality)
	Wt vs luxS::km	2004072946 48	Biofilm– 1/20 TSB – 16°C – 24h	bad (biofilm condition) bad (biofilm condition)	no (bad quality) no (bad quality)
Nov 05	Wt vs lsrG 6h	2006011310 11	Bioreactor – LB – 30% DO – galactose 10 g/l	ok bad (low intensity)	no (no replicate available) no (bad quality)
		2006011314 15		bad (high negative spikes' intensities) bad (high negative spikes' intensities)	no (bad quality) no (bad quality)
	Wt vs lsrG 12h	2006011312 13	Bioreactor – LB – 30% DO – galactose 10 g/l	bad (stationary phrase) bad (stationary phrase)	no (bad quality) no (bad quality)
		2006011316 17		bad (stationary phrase) bad (stationary phrase)	no (bad quality) no (bad quality)
	Wt vs luxS	2006011304 07	Biofilm – 1/20 TSB – 16°C – 24h	bad (technical reasons) bad (technical reasons)	no (bad quality) no (bad quality)
	Wt vs luxS::km	2006011305 08	Biofilm– 1/20 TSB – 16°C – 24h	bad (technical reasons) bad (technical reasons)	no (bad quality) no (bad quality)
Mar 07	Wt vs lsrK	2006011338 40	LB – 6h aerobe – OD 1.4	bad (high negative spikes' intensities) bad (high negative spikes' intensities)	no (bad quality) no (bad quality)
	Wt vs lsrFGE	2006011339 41	LB – 6h aerobe – OD 1.4	bad (high negative spikes' intensities) bad (high negative spikes' intensities)	no (bad quality) no (bad quality)
Sep 07	Wt vs lsrK	2006011347 49	LB – 6h aerobe – OD 1.4	ok Ok	yes yes
	Wt vs lsrFGE	2006011348 50	LB – 6h aerobe – OD 1.4	ok Ok	yes yes
	Wt vs lsrR 4h	2006011343 44	LB – OD 1 – 4h aerobe	ok Ok	yes yes
Jun 08	Wt vs lsrG	2008051903 06	LB – OD 1.4 – 6h aerobe	ok Ok	yes yes
	Wt vs luxS	2008051904 07	LB – OD 1.4 – 6h aerobe	ok Ok	yes yes
	Wt vs luxS::Km	2008051905 08	LB – OD 1.4 – 6h aerobe	ok Ok	yes yes

Oct 08	Wt vs lsrR 6h	2008051909 10	LB – OD 1.4 – 6h aerobe	ok Ok	yes yes
	Wt vs lsrB	2008051911 12	LB – OD 1.4 – 6h aerobe	ok Ok	yes yes

as replicates or in different conditions for comparison (bioreactor samples, biofilm samples, different mutants) (Table 4.1). On all the arrays, each gene is represented by two probes. All the arrays in this experiment were outfitted with the Lucidea^{TM} Universal Scorecard^{TM} (Amersham Biosciences), generating a series of external controls including 10 calibration spikes (added to the labelling reaction in a ratio of 1:1 and spanning up to 4.5 orders of magnitude), eight ratio spikes provided at both low and high concentrations and two negative controls. The entire set of controls was spotted once per pin (32 pins in total). Specific concentrations of these external controls are in Table 4.2.

Table 4.2: Concentrations of the external spikes. Concentration is given in copy number per cell.

	Calibration spikes										**1:3 ratio spikes**		**3:1 ratio spikes**		**1:10 ratio spikes**		**10:1 ratio spikes**	
Concentration of spikes in Cy3	30000	10000	3000	1000	300	100	30	10	3	1	100	1000	300	3000	30	1000	300	10000
Concentration of spikes on Cy5	30000	10000	3000	1000	300	100	30	10	3	1	300	3000	100	1000	300	10000	30	1000

4.3.2 Microarray data preprocessing

The standard analysis of microarray experiment starts from the data which are collected by scanning the signal intensities of the corresponding spots by dedicated fluorescence scanners. The data are mostly in the format of tab-delimited text files. Such files always contain the signal intensities, the information of each probe, including the name, the functional description of the gene represented by that probe and the characteristics of each spot, such as spot size, a quality flag of each spot etc. Before these data can be actually mined for biologically meaningful results, quality assessment and normalization steps are required.

4.3.2.1 Quality Assessment

The quality assessment facilitates to discover possible serious quality problems or even mistakes that occurred during the array printing, the labelling or the hybridization procedures. If the quality assessment step does not report any serious irregularities, the preprocessing step can continue, namely the normalization step should be preformed. In this step, we evaluate the background signal, the intensity ranges of the experiments, the correlation between different replicates and the quality of the spikes spotted on the arrays using visualization techniques. According to the results of the quality assessment, we chose the experiments with good quality used for the follow-up analysis.

Firstly, for all the microarrays the background signal distributions are checked to detect whether there is any region with unreliably high signals. Figure 4.3 uses one randomly chosen microarray as the example to illustrate the visualization of background signal distributions. The figure shows that the background signals in both red and green channels are uniformly distributed. It can be concluded as that there is no spatial non-uniformity. This check is applied on all the microarrays for this study. And all the arrays are with good quality in this respect (figures are not shown).

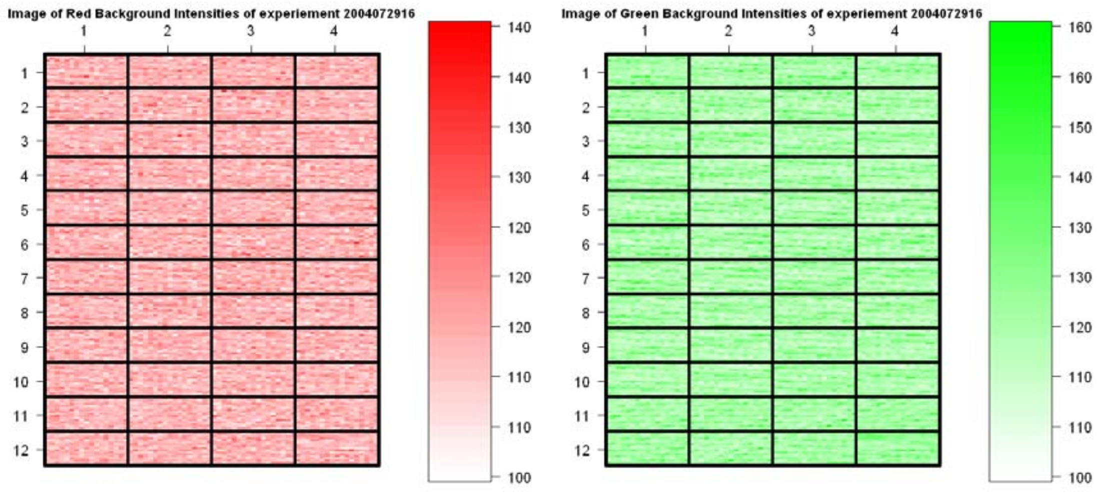


Figure 4.3: Image of red and green background intensities of experiment 2004072916. On the array, the print grids are twelve rows by four columns. The color bars indicate the correspondence between intensities and used labels.

Secondly, we evaluate the intensity range of the microarray signals for each individual microarray. The signals include both background signals and foreground signals. MA-plots are used for this purpose. In a MA-plot, M value is the intensity ratio and A value is the average for a spot on the array. Figure 4.4 shows MA-plots of raw data of four different microarrays. Among these, Experiment 2004072916 (Figure 4.4 (upper left panel)) is of good quality while the other three experiments are of bad quality.

Experiment 2004072916 (Figure 4.4 (upper left panel)) is of good quality in several aspects. The fact that negative spikes (blue dots) are in the low intensity range indicates background signal is low. The fact that the calibration spikes (red dots) with highest concentration corresponds to high intensities and the intensities of cDNA probes (black dots) reach high intensity range indicate the foreground signal is high. Namely, contrast between the foreground and the background is high.

However, the other three experiments in Figure 4.4 are of bad quality for different reasons. Figure 4.4 (upper right panel) shows that for experiment 2004072931 the highest intensities of the calibration spikes are relatively low. This is an indication that the intensities of all the probes are quite low due to a possible intensity measuring problem. Therefore, the data are not reliable because the informative signals and noises are all in the low intensity region and can not be distinguished. Hence, the experiment and several experiments (data not shown) with the same problem were not chosen to use in our follow-up analysis.

The intensities of the calibration spikes with highest concentration of experiment 2004072933 (Figure 4.4 (lower left panel)) are ok. However, as shown in the figure, the calibration spikes (red dots) are not centred into the data points represent cDNA probes (black dots). Since the calibration spikes are of ratio 1:1 for Cy5 and Cy3 labelled two samples, deviating from ratio 1:1 means that most of the genes in the knock out condition have changed their expression levels compared with the wild type. This observation conflicts with the GNA, which says most of genes would not see change in their expression between the two tested conditions and is required by most of the normalization methods. We propose that the reason of why so many genes change their expression levels is because the tested samples on the microarrays are from a stationary phrase. Given that Global Normalization Assumption (GNA, Section 2.2.2.6) is not satisfied, LOWESS normalization can not be performed on this array (the reason why we did not use CALIB to normalize these arrays will be explained below.) Therefore, this array and other arrays (data not shown) also from biofilm condition were not chosen for the follow-up analysis because it is necessary to perform the same normalization method on all the microarrays in a study to facilitate the comparability of the arrays.

For experiment 2006011341 (Figure 4.4 (lower right panel)), the intensities of negative spikes (blue dots) are very high while the intensities of cDNA probes (black dots) are very low. That is to say, contrast between background signals and foreground signals is low. In such case, it is difficult to distinguish informative signals with noises. Arrays with this feature are also excluded from the follow-up analysis.

After checking the quality array by array, we found 18 out of 42 microarrays are of good quality (Table 4.1). However, not all of them are chosen for the follow-up analysis. If one of the arrays of a dye-swap experiment is of bad quality, both of them are not chosen because replicates are necessary when normalizing the data and detecting differentially expressed genes. The last column of Table 4.1 indicates which experiments are chosen for the follow-up analysis. Hereafter, we only consider these experiments for this study.

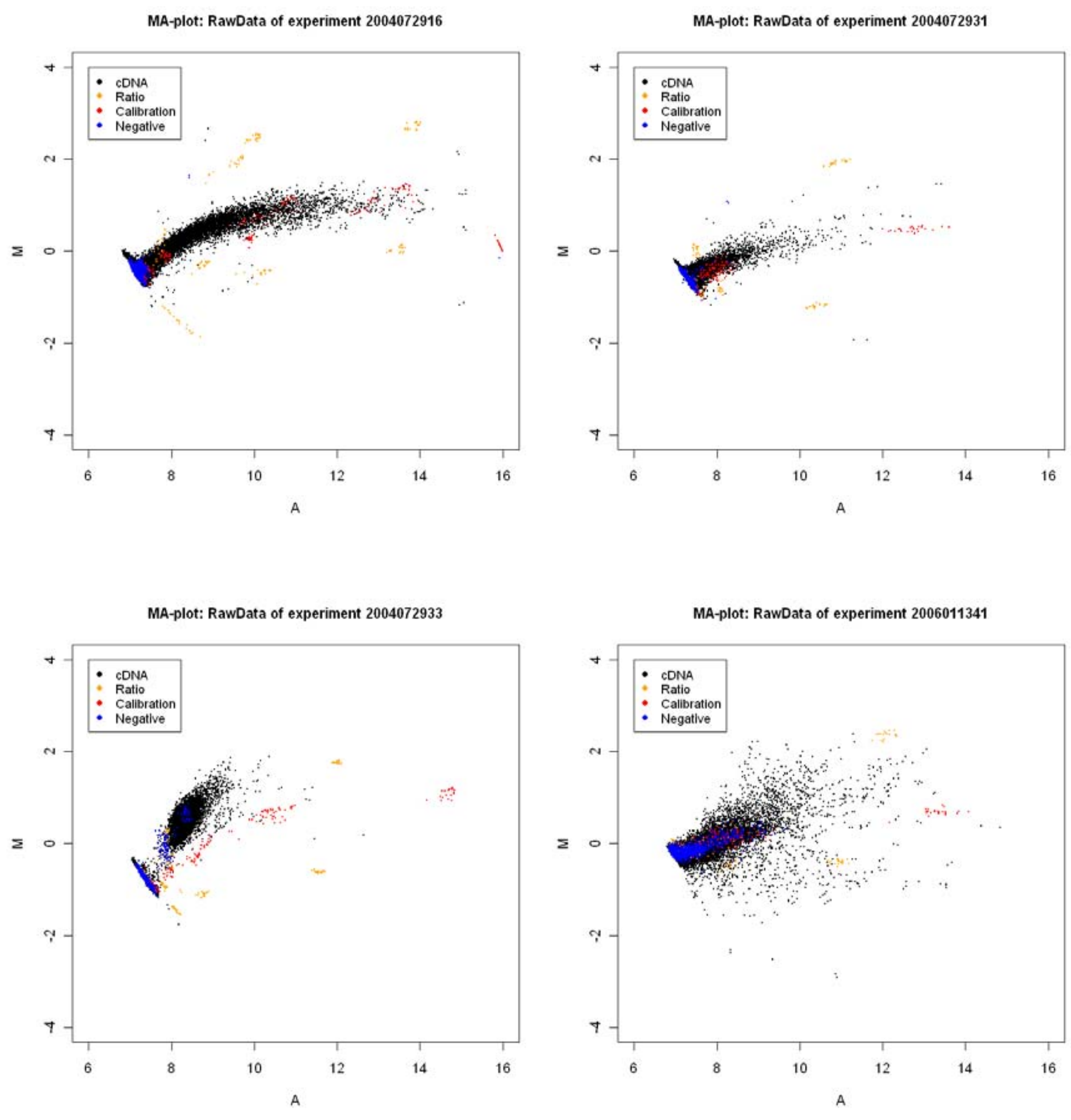


Figure 4.4: MA-pots of the raw data of experiment 2004072916, 2004072931, 2004072933, 2006011341. Experiment 2004072916 is with good quality. The other three experiments illustrate bad quality microarrays of various reasons.

Thirdly, we evaluate the homogeneity of the replicates for every knock out by calculating the correlation between these replicates. *lsrG* mutant experiments 2008051903 and 2008051906 are used as the example. Given that the dye-swap design was used (the same sample are measured on two arrays labelled with Cy3 and Cy5 respectively), the samples labelled red dye on 2008051903 and green dye on 2008051906 are the same biological samples measuring the *lsrG* mutant. In addition, each gene is represented by two probes on each array. Therefore, there are replicates both on one array and on different arrays. Correlations calculated between these replicates are revealed in Figure 4.5. Figure 4.5 (left panel) and (right panel) shows the correlation between the two replicates on one array and on two different arrays respectively. The fact that the intensities of the two replicates on one array are highly correlated indicates that there is no spatial bias on one array. While considering the replicates on two different arrays, the plot deviates from the diagonal. The non-linearity is caused by the bias of the two different dyes. In conclusion, both replicates on the same array and on two different arrays are highly homogenous. Therefore, the replicates are reliable for the follow-up analysis.

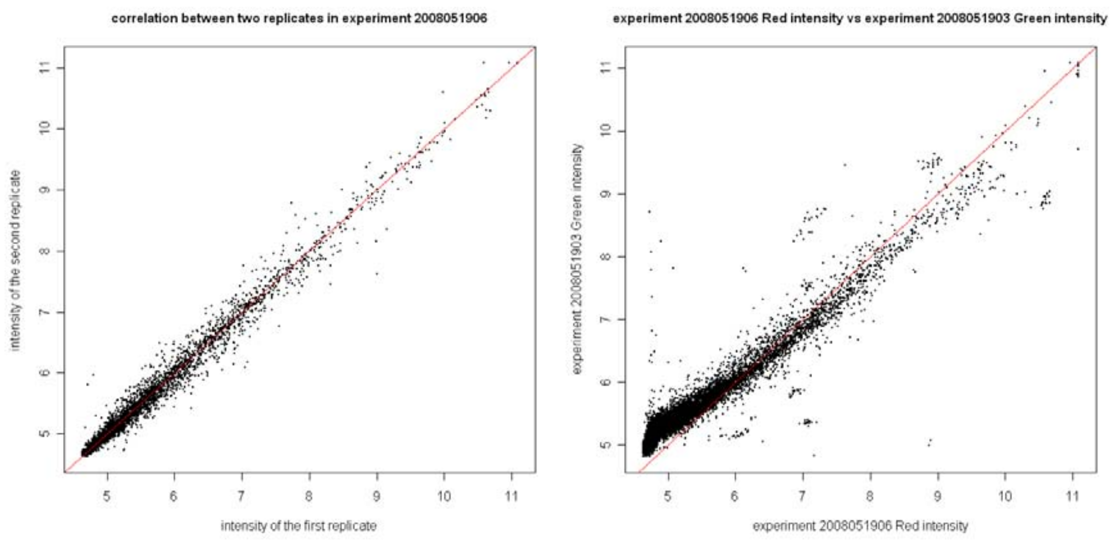


Figure 4.5: Left panel: the correlation between two replicates on experiment 2008051906. Right panel: the correlation between two replicates on two experiments 2008051906 and 2008051903. These two replicates are labelled with red and green dyes respectively in these two experiments.

Fourthly, we evaluate the intensity range of the microarray signals for all the experiments. Figure 4.6 shows that the median and the range of log intensity ratios for all the experiments are different but comparable. The difference can be removed by the normalization procedure explained in next section. The comparability is vital for integration of all the experiments towards obtaining reliable analysis results.

MA-plot is used to visualize not only the whole intensity range of microarray signals but also the intensity-dependent ratio of raw microarray data. In most of the microarray experiments, including the experiments chosen for this study, GNA is satisfied. Therefore, the majority of the points on the y axis (M value) would be located at 0, since Log(1) is 0. Obviously, that is not the case in Figure 4.7 which shows the MA-plot of experiment 2008051903, selected randomly from all the experiments. The shape of the curve is due to the dye effect and the dependence between dye effect and intensities (Section 2.2.3). Normalization is necessary to remove this dye effect.

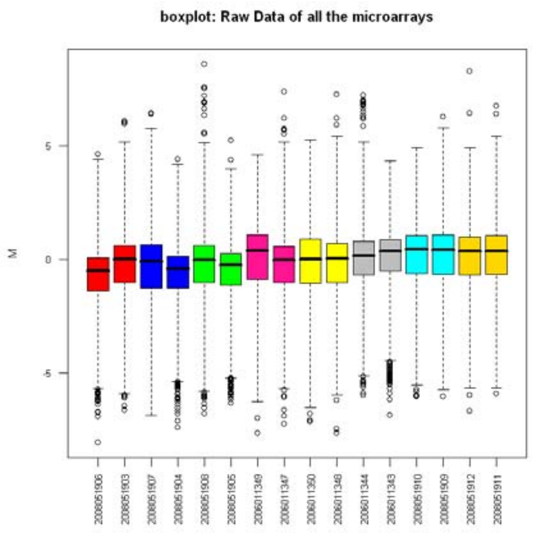


Figure 4.6: Boxplot of the raw log intensity ratios (M-value (Section 2.2.2.5)) between two channels (y-axis) of the spots, calculated for each microarray (x-axis) in the experiment. For every experiment, the lower bound and the upper bound of the box represent the first and third quantile of all the log intensity ratios of this experiment respectively; the bar in the box symbolizes the median log intensity ratios; the whiskers extend to the most extreme data point which is no more than 3 times the inter-quartile range from the box; the circles outside the whisker ranges indicates the outliers. The experiments for the same knock out are with the same color in the figure.

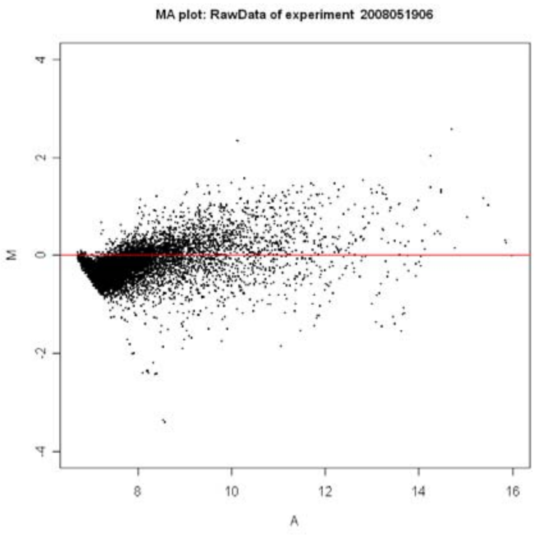


Figure 4.7: MA-plot of raw intensity of experiment 2008051906. The shape of the curve is due to the dye effect and the dependence between dye effect and intensities.

Fifthly, we check the quality of the external control spikes spotted on the arrays to evaluate whether CALIB (Chapter 3) can be used as the normalization method. Figure 4.8 depicts that no matter how much the known concentration is the measured intensities for the spikes are similar and in very low range where the only exception is the spikes with the highest concentration. The feature was found on all the arrays (data not shown). With these poor quality spikes, CALIB package (Chapter 3) is not applicable for normalizing the arrays for this study.

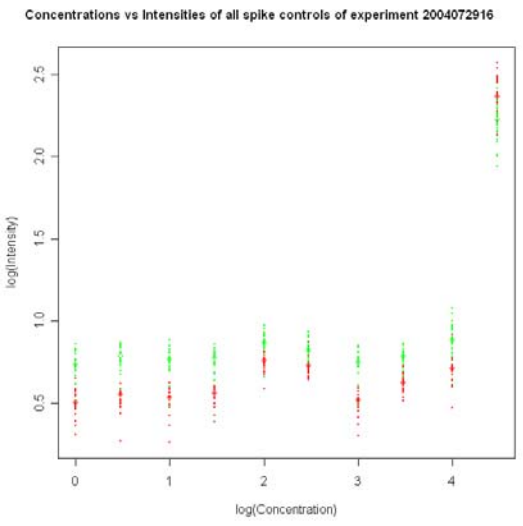


Figure 4.8: Plot of concentrations versus intensities of all spikes on experiment 2004072916. Red dots and green dots indicate spikes labelled with red and green dyes respectively.

4.3.2.2 Normalization

Quality assessment step illustrates that there are different sources of systematic variation in cDNA microarray experiments which affect the measured gene expression levels (e.g. differences in labelling efficiency between the two dyes). The normalization step aims at removing such variation. As mentioned in Section 2.2.2.5, many normalization approaches are proposed based on different assumptions of how the dye biases depend on the overall spot intensity and/or spatial location within the array. LOWESS [29] is the most popular one among those approaches.

We normalize the microarrays using the BioConductor [77] *limma* package [68]. In the *limma* package, the normalization method provided is loess (Locally Estimated Scatterplot Smoothing) normalization instead of lowess (LOcally Weighted Scatterplot Smoothing) normalization. The essential principles of these two methods are equivalent. These normalization methods consider the relation between M and A is in the form of M = c(A), rather than a linear relation. Then a local regression is applied to estimate the c(A) through a robust local linear fits by calculating a moving average along the A-axis. Both loess and lowess belong to this type of local regression methods, which are statistical methods for robustly fitting smooth curves given prior assumptions about the shape or form of the curve. When there is only one predictor variable and no prior weights, the two methods are in principle exactly equivalent. From implementation point of view, loess is most recent. It can be used with more than one predictor variables, accept prior weights, take more flexible models rather than only matrix as input and handles better large scale data. More detailed explanation about the difference between the two R functions can be found in the Stats package.

Three methods are provided by the *limma* package: loess for global normalization, print-tip loess that corrects each print-tip group separately, and robust-spline which is a compromise between print-tip and global loess normalization. By assessing the quality of the microarrays, we notice that the background signals are uniformly distributed (as shown in Figure 4.3), which indicates that there is not much spatial bias on the arrays. Therefore, loess global normalization is performed on all the experiments. Figure 4.9 shows the MA-plot before and after loess normalization.

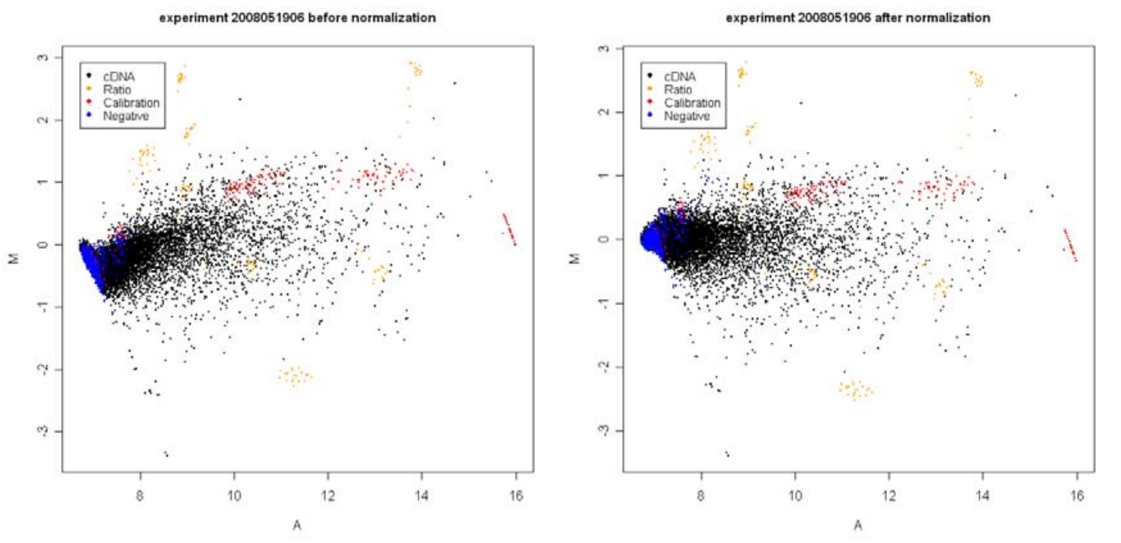


Figure 4.9: MA-plot of experiment 2008051906. Before and after normalization.

In addition to the normalization methods implemented in the *limma* package, other methods exist. For example, normalization methods in the *marray* package were tested in a preliminary comparison. As they showed a result comparable with that obtained via the loess global normalization mentioned above (data not shown), we used the latter method.

4.3.3 Postprocessing on the normalized data

4.3.3.1 Identification of differentially expressed genes

After the data have been normalized, further analysis is necessary to obtain biologically meaningful results. This study aims at exploring the mechanism of the AI-2 mediated quorum sensing. The most straightforward way is to detect the differentially expressed genes for each mutant experiment. These genes are most probably regulated by the mutant gene either directly or indirectly. Revealing the regulation mechanism will help us to discover or refine the function of the mutant gene and to understand the full metabolic pathway.

Many approaches exist to detect genes with differential expression between wild type and mutant experiments. For this study, we mainly used and tested the following two methods:

- Linear Model and Empirical Bayes Methods [68] in the *limma* package: Firstly, probe-wise linear models are built taking into account the design of a microarray experiment. Secondly, the expression data are used to estimate the coefficients and residuals in each linear model. Thirdly, given all the coefficient estimates and standard errors, moderate t-statistics, moderate F-statistics, and log-odds of differential expression are computed by empirical Bayes shrinkage of the probe-wise sample variance towards a pool estimate, which resulting in far more stable inference when the number of arrays is small. Finally, genes are ranked according to one of the chosen statistics. The differentially expressed genes list is exact from the rank by a user-defined maximum number of genes allowed in the list. Detailed derivation can be found in Smyth (2004).

- Significance Analysis of Microarrays (SAM) [78] in the *siggenes* package: SAM method uses a modified t-statistics to assign a score to each gene on the basis of change in gene expression relative to the standard deviation of repeated measurements. The modified t-statistics uses an offset standard deviation (*s(i)+s*$_0$) rather than the standard deviation in the conventional t-test. In this particular study, the modified t-statistics (named as d(i) for gene i), which represents the relative difference in gene expression, is defined as

$$d(i) = \frac{\bar{x}(i)}{s(i)+s_0} = \frac{\sum_j x_j(i) \big/ n}{\{\sum_j (x_j(i)-\bar{x}(i))^2/(n(n-1))\}^{1/2} + s_0}$$

where $\bar{x}(i)$ is defined as the average log ratios between the mutant sample and wild type sample for gene i. The gene specific scatter *s(i)* is the standard deviation of the repeated expression measurements. A small positive constant s_0 is added to the denominator to ensure the variance of *d(i)* is independent of gene expression. The coefficient of variation of *d(i)* was computed as a function of *s(i)* in moving windows across the data. The value for s_0 was chosen to minimize the coefficient of variation.

To find significant changes in gene expression, genes are ranked by magnitude of their *d(i)* values. The expected relative difference $d_E(i)$ is defined as the average over a certain number of permutations of the average log ratios. To identify potentially differentially expressed genes, a scatter plot of the observed relative difference *d(i)* vs the expected relative difference $d_E(i)$ is used (Figure 4.10). The genes that are represented by the points displaced from *d(i)* = $d_E(i)$ line by a distance greater than a threshold delta are the identified differentially expressed genes for this experiment set. The threshold delta is determined by designed false discovery rate (FDR). FDR is defined as follows:

$$FDR = E\left[\frac{\text{number of genes are actually differentially expressed}}{\text{number of genes detected as differentially expressed}}\right]$$

As delta decreased, the number of genes identified as differentially expressed by SAM increase but at the cost of an increasing FDR. In this study, FDR ≈ 0.1 is set for all the eight sets of experiments.

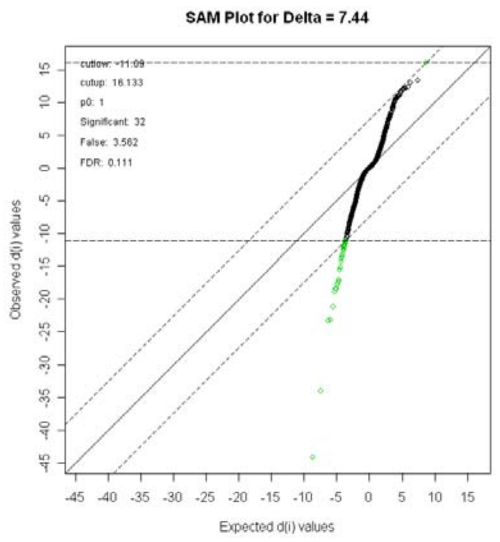


Figure 4.10: SAM plot of lsrG mutant experiments (experiment 2008051906 and experiment 2008051903) for FDR = 0.11 and delta =7.44. The green points on the plot indicate the differentially expression genes with respect to these thresholds.

The above mentioned methods are only two of many approaches to identify differentially expressed genes. Most of the methods to identify differentially expressed genes use a test statistic or a hypothesis test to rank the genes with respect to their differential expression between the two tested samples. Subsequently, an arbitrary threshold or rejection level is chosen to select the genes that warrant further analysis or validation. For instance, in *limma*, the threshold is the maximal number of genes in the differential expression list and in SAM, the threshold is the FDR value. In this study, the choice of these thresholds is totally empirical rather than analytic. To avoid the arbitrariness, a procedure to estimate the total number of genes that is actually differentially expressed starting from the p-values (or FDR corrected p-values) assigned by a certain hypothesis test to every gene [79] were applied in our study. This approach is independent of a certain rejection level defined in advance, thus avoiding the random choice of a threshold. When a rejection level is chosen to identify differentially expressed genes, the choice will result in two types of error in statistics:

- Genes whose expression is not affected by the difference between the two tested samples and therefore have no actual differential expression, can accidentally have a (modified) p-value that is lower than the rejection level and are therefore wrongly declared as differentially expressed genes. In statistics this is called **type I error**. This results in a number of **false positive** genes. Since the number of genes in a microarray, that is not actually differentially expressed, usually is high, the number of false positive genes at commonly used rejection levels (e.g. 5%), can be considerable.

- There are genes that are actually differentially expressed but have a (modified) p-value that is larger than the rejection level. In statistics this is called **type II error**. This results in a certain number of **false negative** genes. These potentially valid genes will be discarded in the further analysis.

Historically, much attention has been paid to the control of the number of false positives at the expense of false negatives remain uncontrolled and too large. However, the number of positives in studies of differential expression could be relatively large. In such case, in order to acquire reliable biological sound results, the control of both false positives and false negatives is equally important. As such, we chose the method which is proposed by De Smet *et al* (2004) [79] and based on Receiver Operating Characteristic (ROC) curves to obtain an optimal balance between type I error and type II error (e.g. reflected by the sum of the specificity and sensitivity).

The method starts with the estimation of the total number of genes that is actually differentially expressed introduced by Storey and Tibshirani (2003) [80]. Suppose we have used certain hypothesis tests to calculate the p-values of respective genes and ordered the genes according to the p-values, so that $p_1 < p_2 < \dots < p_i < \dots < p_n$. Now suppose that the number of genes that are not actually differentially expressed is n_0 and the number of genes that are actually differentially expressed is n_1. Since the distributions of the p-values of the genes without actual differential expression is assumed to be uniform, estimation of n_0 and n_1 are based on the following formula respectively:

$$n_0 = \lim_{p_i \to 1} \frac{n-i}{1-p_i} \tag{4.1}$$

$$n_1 = \lim_{p_i \to 1} \left(n - \frac{n-i}{1-p_i}\right) = \lim_{p_i \to 1} \frac{i - p_i n}{1-p_i} \tag{4.2}$$

In practice, we calculate V_i for every gene g_i as follows:

$$V_i = \frac{i - p_i n}{1-p_i} \tag{4.3}$$

and plot these values as Figure 4.11. If this plot reaches a constant level at a certain gene, this gives us n_1.

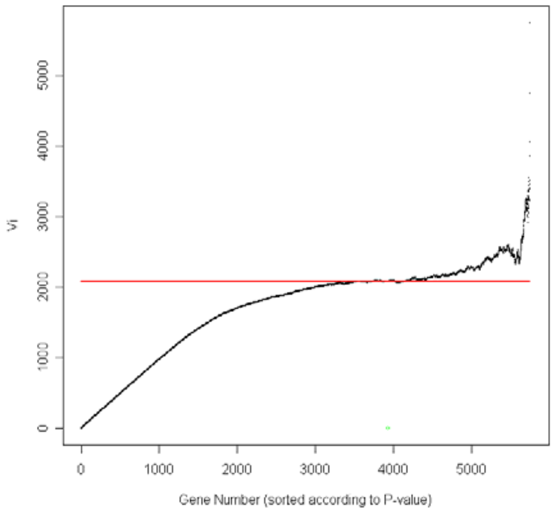


Figure 4.11: Plot V_i for every gene where gene number i on the x-axis and V_i on the y-axis for lsrFGE mutant experiments. The red line indicates the point where V_i reaches a constant level. This value is the estimated n_1. After reaching the constant level, the plot slightly fluctuate due to some approximation used to derive Equation 4.1.

Next, suppose that we identify the genes with a p-value smaller than or equal to a certain rejection level $\alpha = p_i$ as differentially expressed and the genes with a p-value larger than this rejection level as not differentially expressed. When the identified status of differential expressed is compared to the actual status of differential expression, four categories of genes, which are true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN), can be defined in Table 4.3.

Table 4.3: Definition of True and False Positive genes (TP_i and FP_i) and of True and False Negative genes (TN_i and FN_i) at a certain level of $\alpha = p_i$ (p-value of the ith gene after ranking them in ascending order by p-value). For each of them, the formula of the expected value is given.

		Actually differentially expressed?		
		YES	NO	
Identified differentiall y	YES ($p <= p_i$)	TP_i $\approx i - p_i n_0$	FP_i $\approx p_i n_0$ **Type I error**	Pos_i $= i$

	NO $(p > p_i)$	FN_i $\approx n_1 - I + p_i n_0$ **Type II error**	TN_i $\approx (1-p_i)n_0$	Neg_i $= n - i$
		n_1	n_0	

Using the values calculated in Table 4.3, the sensitivity ($SENS_i$) and the specificity ($SPEC_i$) at a certain rejection level $\alpha = p_i$ is defined as [81]:

$$SENS_i = \frac{TP_i}{TP_i + FN_i} = \frac{TP_i}{n_i}$$

$$SPEC_i = \frac{TN_i}{TN_i + FP_i} = \frac{TN_i}{n_0}$$

A ROC curve shows the trade-off or balance between specificity and sensitivity (and hence between the false positive genes and false negative genes) for every possible rejection level and therefore allows for the selection of a rejection level α^{opt} with an optimal balance between them. Optimal can be defined in several ways. In our study, we chose the optimal as the point on the ROC curve with a tangent line with slope 1 (smooth ROC curves is assumed), for which it can be proven that it maximizes the sum of the sensitivity and specificity.

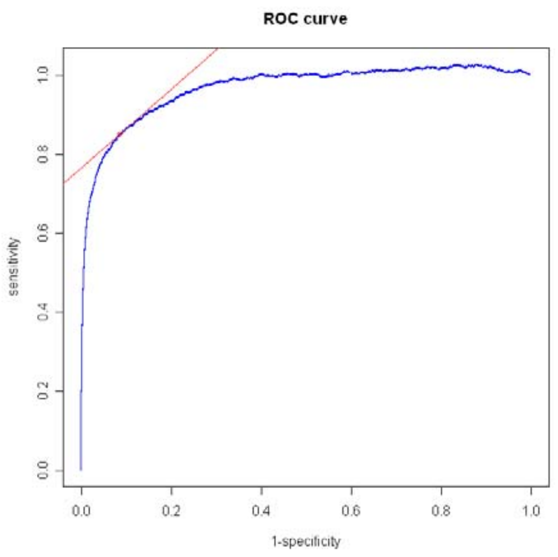


Figure 4.12: ROC curve for lsrFGE mutant experiment. The optimal rejection level α^{opt} which balances the false positive genes and false negative genes is shown as the point on the curve with a tangent line with slope 1 indicated as the red line.

In this section, we describe how we identify differentially expressed genes for each of the eight mutant experiments for studying the *Salmonella* AI-2 mediated quorum sensing. For each mutant experiment, we calculate for each gene the p-values of the modified t-statistics proposed by SAM method in the *siggenes* package. These p-values are used to estimate the actual number of genes with differential expression, which subsequently are used to define the optimal rejection level which balances the false positive genes and false negative genes. Hence, this rejection level helps to identify the list of differentially expressed gene, to avoid the choice of an arbitrary threshold in SAM and also to facilitate the integration and comparison of different mutant experiments. The whole procedure is applied to identify a list of differentially expressed genes for every set of mutant experiments. The eight lists are not shown here due to the space limitation. The union of the eight lists of differentially expressed genes is taken for further analysis.

4.3.3.2 Clustering

As described in the previous section, the first level of interest for microarray analysis is to identify genes whose expression level is significantly changed between mutant and wild type experiments. Such an analysis treats the genes individually rather than exploring their relation with each other. Besides their level of differentially expressed genes, the detailed information about the gene's expression profile as a whole over all the mutant experiments is also interesting. This information is obtained by clustering the genes into biologically meaningful groups according to their pattern of expression. Such groups of related genes (clusters) are more tractable for further studies including gene function, pathway analysis and regulation.

Based on the assumption that similarity in expression of the genes implies functional similarity (and vice versa), the challenge of finding genes that might be involved in the same biological process is thus transformed to the problem of clustering genes into groups based on their similarity in expression profiles. Genes that have similar expression profiles are said to be coexpressed. In our study, we tried two of many approaches available to look for these coexpressed genes.

Hierarchical Clustering

Agglomerative hierarchical clustering (Section 2.2.3) is applied on the union of the eight lists of differentially expressed genes. In this study, the complete linkage is chosen as distance measure between clusters in the hierarchical clustering analysis. The linkage function specifying the distance between two clusters is computed as the maximal object to object distance D(x, y) where object x belong to the first cluster and object y belong to the second cluster. In other words, the distance between two clusters is computed as the distance between the two farthest objects in the two clusters. This distance measure aims at finding compact cluster. As we are more interested in the specific functional groups than the general ones, the complete linkage is a better choice over the single linkage and the average linkage. The clustering result is as Figure 4.13. The final clusters are obtained by cutting the dendrogram into several groups by specifying the desired number of clusters. We tried different values for the number of clusters and chose the one which gives the maximal number of enriched pathways (Section 4.2.3.3) (threshold = 0.01) for all of the resulting clusters as the desired number of clusters (number of clusters = 34.).

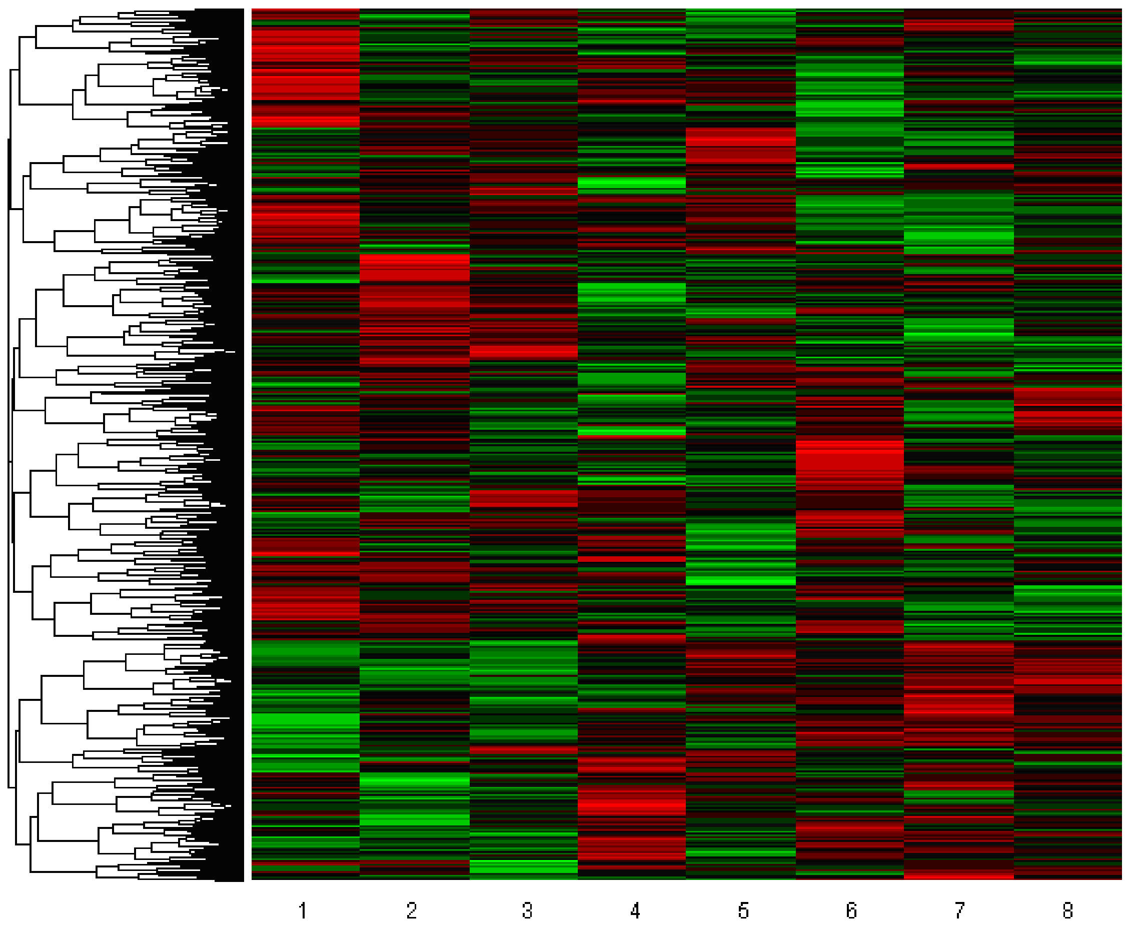


Figure 4.13: The result of hierarchical clustering of all eight sets of experiments for studying AI-s mediated quorum sensing in Salmonella. A heatmap presents the gene expression data, with a dendrogram to its side indicating the relationship between genes or experimental

conditions. The length of a branch in the dendrogram is proportional to the pair wise distance between the clusters. x-axis represents eight sets of mutant experiments (1-lsrG, 2-luxS, 3-luxS::Km, 4-lsrK, 5-lsrFGE, 6-lsrR4h, 7-lsrR6h, 8-lsrB).

AQBC

In order to avoid choose the number of clusters arbitrarily, an adaptive quality-based clustering method (AQBC) (Section 2.2.3) is also applied in this study. This guarantees that the clusters consist of less "noisy" genes and are more informative than the ones found by hierarchical clustering algorithm. Figure 4.14 shows the cluster results of our study by using AQBC.

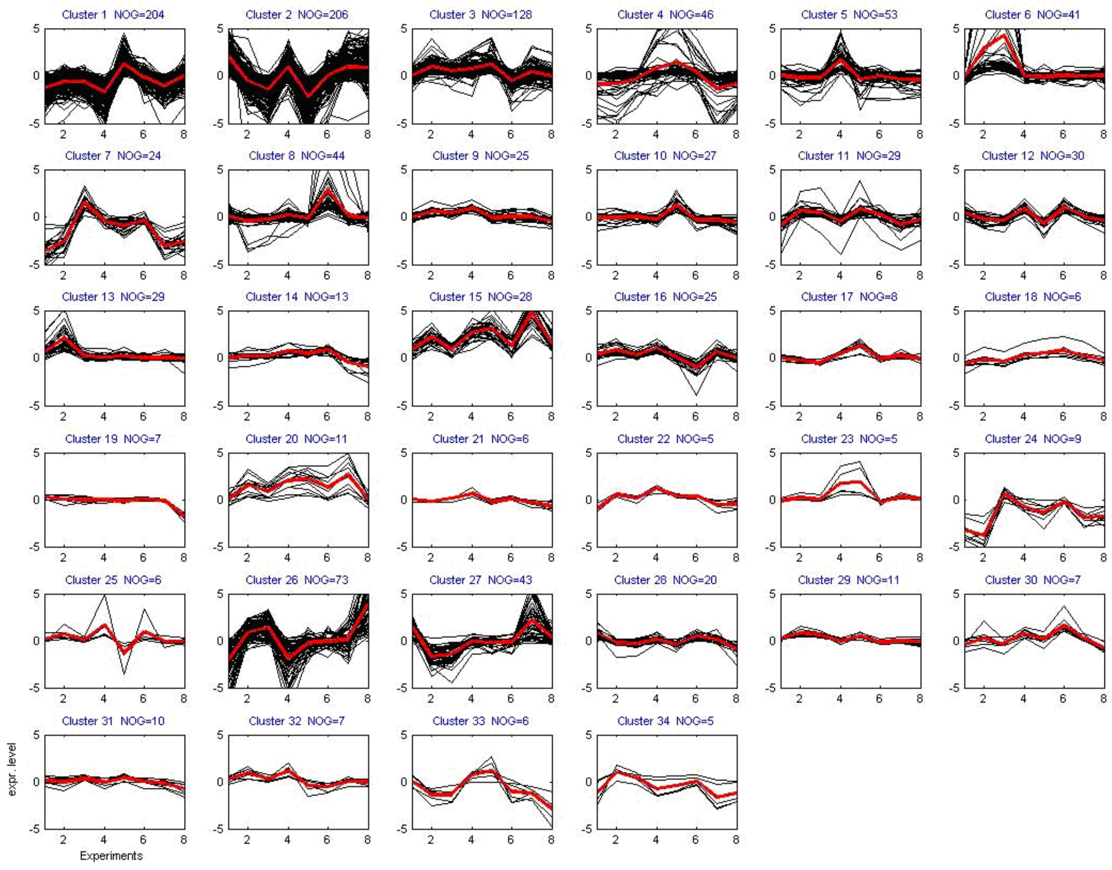


Figure 4.14: Cluster analysis with AQBC (MIN_NR_GENES = 5, S = 0.7) of the differentially expression genes of all the mutant experiments from Salmonella AI-2 mediated quorum sensing study. 34 clusters are found. NOG represents Number Of Genes assigned to each cluster. In the plot of each cluster, x-axis represents eight sets of mutant experiments (1-lsrG, 2-luxS, 3-luxS::Km, 4-lsrK, 5-lsrFGE, 6-lsrR4h, 7-lsrR6h, 8-lsrB). y-axis represents expression levels (in logarithm).

Now we want to have a close look at all of the clusters and exclude some clusters from our cluster list of interest. At this stage, we are not interested in the clusters with really flat mean expression profiles (like cluster 9 in Figure 4.14) because flat expression profile means no different expression in any of the mutant experiments. Therefore, these clusters are not chosen for the follow-up analysis. At this stage, we are interested in cluster 2, 4. For the rest of the clusters, we analyzed them one by one using the following step.

4.3.3.3 Cluster Analysis

Clustering algorithms discover genes that are coexpressed under all the conditions of interest. Whether or not these clusters are biologically sound is up to be verified. One of the most widely-used verification for clustering result is to search for functional enrichment or pathway enrichment in these genes using Gene Ontology or other annotation terms.

Several statistical tests are available for the enrichment calculation, such as the binomial, χ^2 and the hyper-geometric tests. The tests all aim to calculate how a query list is enriched in a previously computed reference list. When the query list has few or no intersections with the reference list, the binomial and χ^2 tests are more appropriate. While when the query list is compared with the reference, or when it comprises a subset of the reference list, the enrichment problem is best modelled by the hyper-geometric distribution.

In hyper-geometric test, term mapping counts in the query and reference list are used to form a 2 x 2 contingency table, from which the difference between observation and expectation for each category is measured. The hyper-geometric test uses the hyper-geometric distribution to calculate the probability to obtaining the contingency table as described above by chance. In our study, the query list is the list of genes belonging to a cluster and the reference list is the genes belong to a certain functional category in the case of functional enrichment calculation or the genes belong to a certain pathway in the case of pathway enrichment calculation. Therefore, the contingency table formed to illustrate our case is as Table 4.3.

Table 4.3: Contingency table for calculating functional enrichment of clustering results.

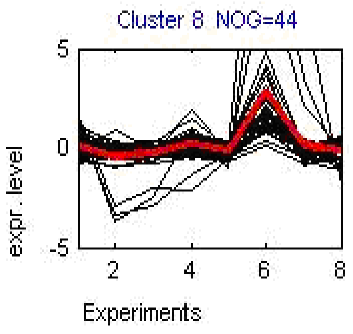
	Belong to a certain functional category	Not belong to a certain functional category	Total
In the cluster	*k*	*m-k*	*m*
Not in the cluster	*n-k*	*N+k-n-m*	*N-m*
Total	*n*	*N-n*	*N*

As illustrated by Table 4.3, when clustering a data set with *N* genes in total, one of the found clusters is with *m* genes. The hyper-geometric distribution describes the probability that exactly *k* genes in that cluster belong to a certain functional category with *n* genes. The probability is given by

$$P(X = k) = \frac{\binom{m}{k}\binom{N-m}{n-k}}{\binom{N}{n}}$$

where the choose function $\binom{m}{k}$ of *m* and *k* is interpreted as the number of *k*-element subsets of an *m*-element set, that is the number of ways that *k* things can be 'chosen' from a set of *m* things.

For this study, pathway categories of *Salmonella* Typhimurium *LT2* provided by the BioCyc Tier 3 Pathway/Genome Database (PGDB) (http://biocyc.org/STYP99287/organism-summary?object=STYP99287) is used as the pathway annotation term. The significant level is chosen as 0.05. FDR correction is applied on the calculated p-values.



Enriched Pathway		FDR corrected p-value
1.6.15.1.9.2	Sugar Acids Degradation	1.35e-03
1.6.15.1	Plant Secondary Metabolites Degradation	3.31e-03
1.3.9	Fatty Acids and Lipids Biosynthesis	3.01e-02
1.6.15	Secondary Metabolites Degradation	3.31e-03
1.6.15.1.9	Sugar Derivatives Degradation	3.31e-03
1.6.15.1.9.2.4	D-glucarate degradation	2.30e-04
1.3.12.2.2	Putrescine biosynthesis III	3.04e-02
1.3.9.8	Lipid A-core biosynthesis	6.83e-04
1.6.6.1.1	Formaldehyde assimilation I (serine pathway)	4.53e-02
1.3.7.7	GDP-mannose metabolism	1.53e-02
1.6.14.2.14	D-mannose degradation	1.53e-02
1.2.33	Superpathway of D-glucarate and D-galactarate degradation	2.30e-04
1.6.15.1.9.2.1	D-galactarate degradation	2.30e-04
1.6.7.6	Glycolate and glyoxylate degradation	3.04e-02

Figure 4.15: Cluster analysis for cluster 8. Expression profiles and enriched pathways for cluster 8.

Figure 4.15 shows the analysis results for cluster 8. Cluster 8 contains the *lsr* genes, of which it is known that they are regulated by the AI-2/LuxS mediated quorum sensing system (Figure 4.2). The enriched pathways of this cluster are listed in Figure 4.15. Some of these pathways are interesting related to the metabolic role of AI-2 (e.g. glycolate and glyoxylate degradation I; phosphoglycolic acid is a breakdown product of AI-2). Other pathways could be related to the link of the AI-2/LuxS mediated pathway with biofilm formation (e.g. LPS biosynthesis, polyamine biosynthesis). This is currently under further investigation.

In fact, most of the interesting clusters are enriched in metabolic genes, such as encoding for proteins involved in fatty acid synthesis (cluster 26), ethanolamine utilization (cluster 27), amino acid biosynthesis (cluster 6) and sulphate/sulphite redox reactions (cluster 4). This already points to a more metabolic role for the AI-2/LuxS mediated pathway. However, cluster 2 contains genes involved in flagella biosynthesis, motility, chemotaxis, propanediol utilization (also in cluster 12, with a similar profile as cluster 2) and invasion (SPI-1, *Salmonella* pathogenicity island-1, HilA regulated). Other experimental data (proteomics, ChIP-chip) pointed to a link between propanediol as a breakdown product of AI-2, flagella phase variation, and invasion. This cluster would support these findings. In addition to SPI-1, also SPI-2 came out as enriched in some clusters, i.e. cluster 7 and cluster 24 also show similar profiles and both contain genes encoding a type three secretion system and its effector proteins involved in survival of *Salmonella* in macrophages. The role of LuxS in survival of *Salmonella* in macrophages has been made before [82]. However, based on our results, the link between virulence and the AI-2/LuxS mediated pathway seems to be more metabolic. This will be further investigated. The enriched pathways for all these analyzed clusters are as Appendix A.

4.4 Conclusions

In this chapter, we proposed a microarray analysis flow which contributes in experimental design, data preprocessing, differential expression detection and clustering analysis. The *Salmonella* Typhimurium AI-2/LuxS mediated pathway was studied using this analysis flow. Seven sets of knocked out experiments were performed for this study. We explained how we selected the microarrays with good quality from all the experiments. After normalizing these arrays, differentially expressed genes are detected. Clustering analysis was performed on the union of differentially expressed genes for each knocked out experiment set. The cluster analysis results confirm the metabolic role of AI-2.

Chapter 5

*Pro*Bic Model

By exploiting the modularity of gene networks, the complexity of the network inference problem has been largely reduced: in contrast to assigning an individual program to each single gene as is done with direct network inference methods, module-based network inference procedures assign a regulatory program to pre-grouped gene sets (modules). This drastically lowers the number of interactions that needs to be evaluated during the inference process. Module inference is thus a crucial step in the inference problem. Module inference is solved by biclustering microarray datasets and searching for gene sets that are coexpressed together with the conditions under which they are coexpressed.

In this chapter, we describe a novel biclustering method, called ProBic, which was developed within the framework of Probabilistic Relational Models (PRMs). ProBic is an efficient algorithm that simultaneously identifies a set of potentially overlapping biclusters in a gene expression data set and which can be used in both a query-driven and a global setting. By using simulated data sets, we investigated the behaviour of the algorithm under different parameter settings. It is also shown that the algorithm handles missing values naturally and performs well if compared with other query-driven biclustering algorithms. Finally, when tested on a cross-platform E.coli compendium, ProBic was shown to detect biologically relevant biclusters under biologically from a set of seed genes.

5.1 Introduction

The reconstruction of the cellular signalling pathways is one of the foremost challenges in system biology [61]. While a large amount of high throughput omics data are available, the reconstruction of this signalling network still remains a highly underdetermined problem. Currently, insufficient data is available to uniquely identify all the interactions and their parameters in the model. However, transcriptional networks exhibit a modular organization [83], an aspect that is successfully exploited in a number of biclustering and module inference methods [84]. Module inference greatly reduces the number of required parameters, while preserving the essential characteristic of the network.

Biclustering was originally introduced by [85] in the context of gene expression data. Biclustering algorithms are well suited to identify regulatory modules. They perform simultaneous clustering in both the gene and condition dimensions and results in the simultaneous identification of gene sets that are coexpressed and the conditions under which they are coexpressed [57]. Such a gene set is called a module. The drawback of most existing biclustering algorithms is that they do not easily allow for the incorporation of additional biological knowledge. Molecular biologists might be interested in knowing whether a certain subset of genes is indeed coexpressed or in knowing which other genes are clustering together with a predefined set of seed genes.

Probabilistic Relational Models (PRMs) [86-88] were recently developed as an extension of Bayesian networks to the relational domain. PRMs have since been successfully applied to a variety of relational machine learning problems [89-94]. They provide an intuitive way of representing complex models as a composition of simple relations between its constituents. As we shown in Section 5.2, PRMs are also well suited for both query-driven and global biclustering of gene expression data.

We present the *Pro*Bic model, an alternative approach to identify overlapping biclusters in gene expression data using Probabilistic Relational Models, which is able to incorporate biological prior information in the form of a set of seed genes. *Pro*Bic is an efficient biclustering method that simultaneously identifies a number of potentially overlapping biclusters and allows biological prior knowledge (by means of a list of seed genes) to be incorporated.

5.2 Probabilistic Relational Models

A PRM specifies a probability distribution over a relational dataset and consists of two main parts: (1) a relational component that describes the relations in the domain of interest and (2) a probabilistic component that describes the probabilistic dependencies between the different entities in the domain. Given a particular dataset instance, a PRM specifies a joint probability distribution over this particular instance.

The JPD associates a probability to all possible instances that are a completion of the relational skeleton. One of the main questions to be answered is what the most likely model is that best explains the collected data. This question can be addressed using the ML or MAP principles that identify the model with the highest likelihood (or posterior) given the data.

5.3 Biclustering

The term biclustering was first introduced by Cheng and Church [85] in the context of gene expression data. Biclustering algorithms perform simultaneous clustering in both the gene and the condition dimension. The result is a set of biclusters that each contains a subset of genes and conditions. Starting from a m by n data matrix E, with a set of rows R = $r_1,\dots, r_m$ and a set of columns C = $c_1,\dots, c_n$, a bicluster (I, J) can be defined as a subset of rows I = $i_1,\dots, i_r$ and a subset of columns J = $j_1,\dots, j_s$ of E, such that the bicluster satisfies some specific measure of homogeneity [57].

Table 5.1 summarizes and compares the characteristics of *Pro*Bic with respect to the current state of the art algorithms. The *Pro*Bic biclustering model builds upon the expertise of previous work and extends the current generation of biclustering methods in the following areas:

- Many biclustering algorithms have no explicit model for the identification of multiple biclusters [85, 95-98]. Heuristic approaches such as masking and the use of well chosen initializations, are often applied to identify more than one bicluster. *Pro*Bic accommodates for this and uses a model-based approach for identifying multiple and potentially overlapping biclusters simultaneously.

- Current methods, that allow to model multiple biclusters simultaneously [93, 99, 100], all have an additive model for overlapping biclusters. Computational tractability is assumedly the main reason for using an additive overlap model. *Pro*Bic can use different overlap models that still retaining computational tractability.

- *Pro*Bic can be used both in a global mode as in a query-driven mode. Biclusters can thus both be identified by incorporating prior knowledge where applicable (in the form of a set of seed genes) and by using a global biclustering approach.

- *Pro*Bic performs the biclustering efficiently by well-chosen decompositions of the posterior distribution (see Section 5.4.4). No prior discretization of the gene expression data is required and ProBic is able to remove seed genes from the resulting bicluster if that gene has no matching expression profile with the bicluster. This feature is important in practice as biologists often have a list of potential seed genes rather than a list of definite seed genes.

Table 5.1: Table comparing different qualitative aspects of state of the art biclustering algorithms that have been successfully applied to gene expression data. Each of the columns highlights a different aspect: GBCL (global biclustering): indicates if the algorithm performs a global biclustering; MULT (multiple biclusters): indicates if the model can identify multiple biclusters simultaneously (without masking or other heuristics); OVL (overlap): indicates if overlapping biclusters are modeled with an explicit overlap model; PRB (probabilistic): indicates if the algorithm is based on a probabilistic framework such as Bayesian networks; NMR (noise and missing value robustness): is the algorithm robust w.r.t. noise and missing values; QD (query-driven): can the algorithm be used in a query-driven mode.

Name	**GBCL**	**MULT**	**OVL**	**PRB**	**NMR**	**QD**
BBC	+	+	-	+	+	-
ccc-biclustering	+	+	-	-	-	-
cMonkey	+	+/- [1]	-	+	+	-
DISTILLER	+	+	-	-	+/- [2]	-
GEMS	+	-	-	+	+	-
ISA	+	-	-	-	+	+
OPSM	+	+	-	-	-	-
Plaid models	+	+	+	-	+	-
*Pro*Bic	+	+	+	+	+	+
QDB	-	-	-	+	+	+
ReMoDiscovery	+	+	-	-	+/- [2]	-
SAMBA	+	+	-	-	+	-
SBK	+/- [3]	+	+	+	+	-

1. cMonkey also tends to repeatedly find the same biclusters from different seeds, they have however integrated a prior in their model that preferably includes each gene in two biclusters.

2. ReMoDiscovery is two-phased. In the first phase of the algorithm, seed modules are identified by means of a non-robust Apriori-based search step. In the second phase these modules are expanded using a noise robust expansion step.

3. The SBK algorithm is in between a gene clustering and a biclustering algorithm. It identifies gene clusters with an associated regulatory program that clusters all the conditions for that gene cluster.

5.4 Model and Algorithm

5.4.1 Model framework

An overview of the *Pro*Bic model is shown in Figure 5.1, the model contains three classes: Gene, Array and Expression. For each class, a set of specific gene, array and expression objects exist that are denoted by lowercase letters g, a and e respectively. The complete set of genes, arrays and expression objects are indicated by uppercase letters G, A and E. Detailed explanation of the model schema is explained by Figure 5.1.

The Joint Probability Distribution (JPD) for the *Pro*Bic model is shown in Equation 5.1. For notational convenience, the dependency on the model parameter is not explicitly written in the Conditional Probability Distributions (CPDs).

$$\begin{aligned} likelihood &= P(E.L, G.B, A.B, A.ID) \\ &= P(E.L \mid G.B, A.B, A.ID) \cdot P(A.B) \cdot P(G.B) \cdot P(A.ID) \\ &= \prod_{e \in E} P(e.L \mid e.gene.B, e.array.B, e.array.ID) \\ &\cdot \prod_{a \in A} \prod_{k \in K} P(a.B_k) \cdot \prod_{a \in A} p(a.ID) \cdot \prod_{g \in G} P(g.B) \end{aligned} \tag{5.1}$$

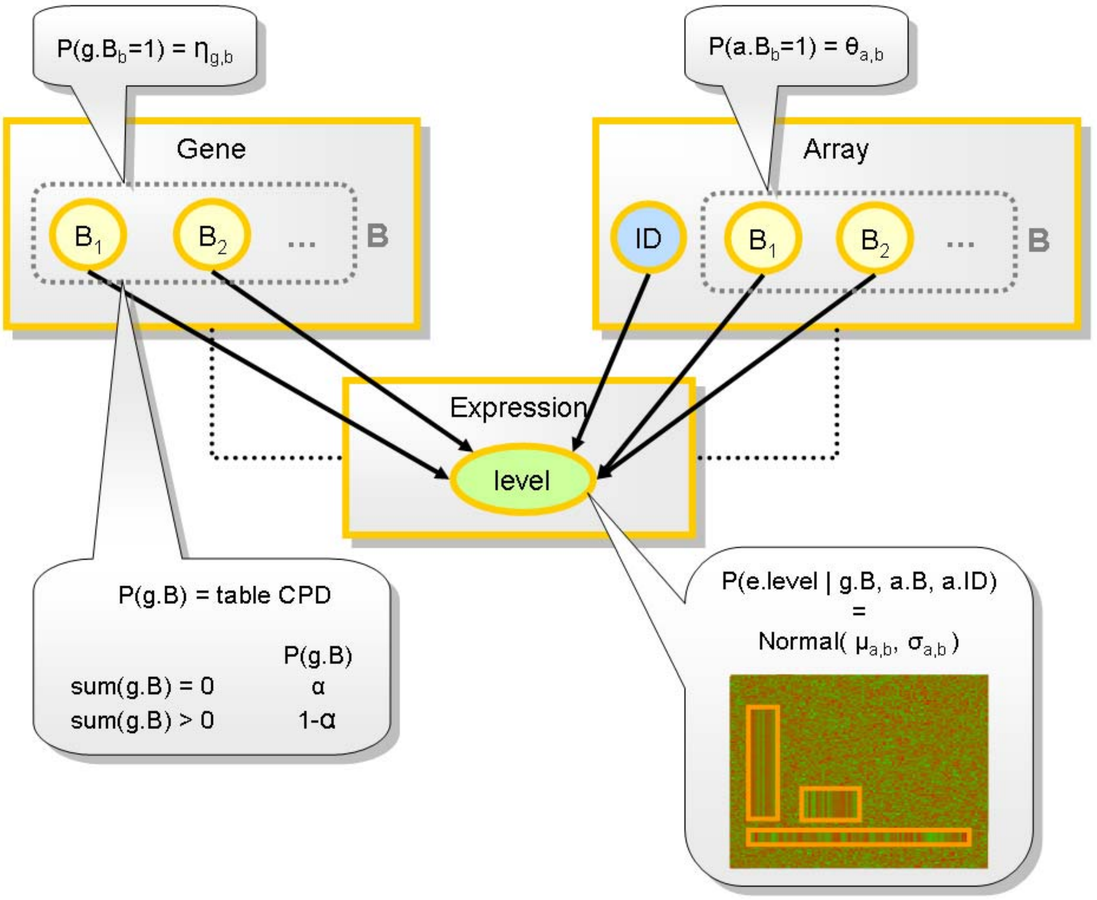


Figure 5.1: Schematic overview of the ProBic model and the conditional probability distributions of each of the attributes. The three classes of the PRMs are Gene, Array and Expression. Expression has one attribute level that contains the expression level and it contains two reference slots gene and array that indicate the gene and the array for which the expression level is measured. Gene and Array each have a number of boolean attributes B_b that represent the presence or absence of respectively a gene or array in a bicluster b. For notation convenience, these attributes are grouped in a vector $B = \{B_1, B_2, \dots, B_b\}$. The class Array has an additional attribute ID that uniquely identifies each array. The conditional probability distribution P(e.level|e.gene.B, e.array.B, e.array.ID) is modelled as a set of Normal distributions, one for each array-bicluster combination. A number of prior distribution $P(a.B_b)$, $P(g.B_b)$ and P(g.B) allow expert knowledge to be introduced in the model.

As shown in Equation 5.1, the JPD is composed by different parts, including the conditional probability distributions and the priors. In the following sections, a detailed overview of these different parts is given.

5.4.2 The conditional and prior probability distributions

5.4.2.1 Expression level conditional probability distribution

This is the main CPD for the biclustering model and it consists of two individual factors:

$$\begin{aligned}&P(e.level \mid e.gene.B, e.array.B, e.array.ID) = \\ &P_1(e.level \mid e.gene.B, e.array.B, e.array.ID) \cdot \\ &P_2(e.level \mid e.gene.B, e.array.B, e.array.ID)\end{aligned} \tag{5.2}$$

The first factor P_1 describes the main conditional probability of the expression level given the gene to bicluster and array to bicluster assignment attributes. Three separate cases are identified and a separate definition is given for each of these cases: one for the background distributions, one for bicluster regions without overlap and one for overlapping biclusters. After these three separate definitions, $P_1(\ldots)$ is re-defined with a single definition that covers all three cases.

Case 1: background distributions

Let us first evaluate how $P_1(\ldots)$ is modelled in the case where an expression level is part of no bicluster ($e.gene.B \cap e.array.B = \phi$), i.e. the expression level is part of the background distributions. In this case, the probability $P_1(\ldots)$ is modelled as a Normal distribution for each array a, each with a separate set of parameters $(\mu_a^{bgr}, \sigma_a^{bgr})$. The parameters of the background distributions are fixed and derived a priori from the dataset using a robust background distribution.

Case 2: bicluster without overlap

Since there is no overlap between different biclusters in this case, every expression level is part of exactly one bicluster. The probability $P_1(\ldots)$ is again modelled using Normal distributions with parameters (μ, σ) that indirectly depend on the gene to bicluster and array to bicluster assignments $(g, B, a.B)$ and on the unique array identifier *a.ID*. So a (μ, σ) value has to be defined for every possible combination of $g.B, a.B, a.ID$.

Since we consider here the case of an expression level that belongs to exactly one bicluster, the combination of $g.B, a.B, a.ID$ is uniquely determined by the array a and the bicluster b to which the expression level belongs. Then the probability for an expression level which is only part of a single bicluster is defined as:

$$\begin{aligned}&P_1(e.level \mid e.gene.B, e.array.B, e.array.ID)\\ &= P_1^*(e.level \mid e.array = a, e.bicluster = \{b\})\\ &= P_1^*(e.level \mid \mu_{a,b}, \sigma_{a,b})\end{aligned} \tag{5.3}$$

We introduced the probability $P_1^*(e.level \mid e.array = a, e.bicluster = \{b\})$ as the probability that an expression level belongs to a single bicluster. The attribute $e.biclusters$ does not formally exist in the model, but it is implicitly defined as the set of bicluster indices to which the expression level belongs, namely the intersection $e.array.B \cap e.gene.B$. In this case, the expression level belongs to exactly one bicluster b.

Case 3: overlapping biclusters

The probability $P_1(\ldots)$ in case of overlap can in principle be defined in many different ways. For example, the overlap probability could be modelled as having a high probability for expression values which are close to the sum, the average, the weighted sum, the minimum, the maximum, etc. of the individual means. It could be modelled with a distribution that has a separate set of parameters (μ, σ). We propose a model with some desired properties on both a biological and computational level. Similar to case 2, a probability $P_1(\ldots)$ has to be associated with every possible combination of $g.B, a.B, a.ID$.

Defining a separate set of parameters (μ, σ) for each of the overlap combinations would result in an over fitted model. For example, in case of two overlapping biclusters α and β with a single gene $g_{overlap}$ in the overlap region, a separate parameter set $(\mu_a^{\alpha,\beta}, \sigma_a^{\alpha,\beta})$ would be defined for each array in the overlap region. Such over-parameterization of the model leads to overfitting and needs to be avoided.

A better parameterization would be not to introduce new set of parameters (μ, σ) for the overlap, but rather to model the overlap region using the parameter sets that were already defined in case 2 (one per array to bicluster combination). We will define the expression level in the overlap region as the geometric mean of the separate bicluster probabilities. Due to the computation efficiency, this model takes one underlying assumption, which assumes the distribution of each individual bicluster is the same. Formally, $P_1(\ldots)$ is defined as:

$$\begin{aligned}
&P_1(e.level \mid e.gene.B, e.array.B, e.array.ID) \\
&= \prod_{b \in \{iset(e.gene.B) \cap iset(e.array.B)\}} P_1^*(e.level \mid a,b)^{1/\#\{iset(e.gene.B) \cap iset(e.array.B)\}} \\
&= \prod_{b \in iset(B_e^i)} P_1^*(e.level \mid \mu_{a,b}, \sigma_{a,b})^{1/\#iset(B_e^i)}
\end{aligned} \tag{5.4}$$

where the following notation was introduced: $iset(X)$, denoting the set of indices I for which the vector element Xi of binary vector X are 1. B_e^i is defined as the dot product of $e.gene.B$ and $e.arary.B$. Therefore, $iset(B_e^i)$ is the set of bicluster-indices in the intersection of $e.gene.B$ and $e.arary.B$, or formally: $iset(B_e^i) = iset(e.gene.B) \cap iset(e.array.B)$. Finally, $\# iset(B_e^i)$ is the number of elements in this set.

General case (covering case 1, 2 and 3)

For notational convenience, we will describe a general notation covering all case 1, 2 and 3:

$$\begin{aligned}
&P_1(e.level \mid e.gene.B, e.array.B, e.array.ID) \\
&= \prod_{b \in iset(B_e^i)} P_1^*(e.level \mid \mu_{a,b}, \sigma_{a,b})^{1/\#iset(B_e^i)}
\end{aligned} \tag{5.5}$$

Case 2 is implicitly covered in the notation of case 3 as it can be formulated as a special case of 'overlap' with only one bicluster. For including case 1 in this definition, we introduce a virtual bicluster with index -1 that describes the background as a special kind of bicluster. The parameters $(\mu_{a,b=-1}, \sigma_{a,b=-1})$ of this background bicluster are defined a priori based on the expression data and do not change during optimization. This background bicluster can by definition not overlap with any other biclusters.

The definition of the set $iset(B_e^i)$ also slightly changes as $\prod_{b \in iset(B_e^i)}$ now covers two cases:

B_e^i empty: background distribution, the product is over the set $b \in [-1]$ so $iset(B_e^i) = [-1]$.

B_e^i not empty: bicluster distribution, the product is over the set of biclusters in the intersection and never includes $b = -1$ by definition.

The second factor P$_2$ models a penalty factor that reduces the model complexity by decreasing the probability for adding an expression level to a bicluster compared to adding it to the background. The reduction of model complexity in Bayesian networks is usually done by including additional terms in the log-likelihood or log-posterior distributions such as the Bayesian information criterion (BIC) [101] or the Akaike information criterion (AIC) [102] and these criteria could equally be applied to PRMs. However, the application of these classical techniques for model complexity reduction to the *Pro*Bic model lead to computational intractability if an Expectation-Maximization algorithm is used to find MAP solution. The reason is that in the substeps of the Expectation Maximization (EM) algorithm, independent optimizations per gene or per array are not possible anymore if one of the above criteria is included in the model.

Therefore, an alternative strategy is used to reduce model complexity by introducing a 'penalty' factor $P_2(\ldots)$. Without such a penalty factor, the MAP solution for *Pro*Bic would include a very large number of biclusters since each additional bicluster also introduces additional degrees of freedom to the model, thus leading to a higher probability for models with many biclusters. The additional penalty factor $P_2(\ldots)$ is defined such that it only allows a set of expression values to be included in a bicluster if they are on average N times more likely to be in their respective bicluster distributions than in their background distributions. The factor $P_2(\ldots)$ decomposes similarly to $P_1(\ldots)$, leading to the following expression:

$$P_2(e.level \mid e.gene.B, e.array.B, e.array.ID) = \prod_{b \in iset_B(e)} P_2^*(e.level \mid b)^{\frac{1}{\#iset_B(e)}} \tag{5.6}$$

where $P_2(e.level \mid b) = \pi_{bgr}$ if the expression level is in the background ($b = -1$) and $P_2(e.level \mid b) = \pi_{bicl}$ if it is in a bicluster ($b \neq -1$).

This implies that a subset of expression values for a particular gene or array, will be assigned to a bicluster if Equation 5.7 holds:

$$\begin{gathered}\prod_{e \in E_a} P^*(e, bicl) \cdot \prod_{e \in E_s} \pi_{bicl} > \prod_{e \in E_s} P^*(e, bgr) \cdot \prod_{e \in E_s} \pi_{bgr} \\ \Leftrightarrow \\ \prod_{e \in E_s} \frac{\pi_{bicl}}{\pi_{bgr}} > \prod_{e \in E_s} \frac{P^*(e, bgr)}{P^*(e, bicl)}\end{gathered} \tag{5.7}$$

The user-defined ratio $\frac{\pi_{bicl}}{\pi_{bgr}}$ indicates how many times more likely it must be on average that an expression value is part of the bicluster distribution compared to being part of the background distribution before such a set of expression values Es is actually added to the bicluster.

5.4.2.2 Prior distributions

As shown by Equation 5.1, expect for the expression level CPD, a number of priors which allow introducing expert knowledge also contribute to the JPD of the model.

(1) Prior probability for gene to biclusters assignment P(g.B). This prior is also defined as a combination of two factors that each defines a separate aspect of the prior.

$$P(g.B) = P_1(g.B) \cdot \prod_{b \in B} P_2(g.B_b) \tag{5.8}$$

The first factor P_1(g.B) reflects general prior knowledge on gene to bicluster assignments. It is the prior probability that a gene is part of any bicluster. This prior indirectly has an effect on the average number of genes in a bicluster. The motivation for this prior is biological in nature: biologists are often interested in information concerning specific pathways. By penalizing the addition of genes to a bicluster, the average number of genes in a bicluster can be reduced, keeping only the best fitting genes in the bicluster profile. It is parameterized in the following way:

$$P_1(g.B) = \begin{cases} 1-\beta, & \text{if sum(g.B)} = 0 \\ \beta, & \text{if sum(g.B)} \neq 0 \end{cases} \tag{5.9}$$

The second factor $P_2(g.B_b)$ reflects prior knowledge on specific gene to bicluster assignments, namely, the probability for a particular gene g to belong to a particular bicluster b. This prior can be used to introduce expert knowledge in the model specifying which genes are highly likely to be in a specific bicluster. It is a user defined value and chosen according to the understanding of a particular gene. The prior is parameterized as:

$$P_2(g.B_b = 1) = \eta_{g.b} \tag{5.10}$$

(2) Prior probability for array to biclusters assignment $P(a.B_b)$. Similar to the prior on gene to bicluster assignments P(g.B), $P(a.B_b)$ is the prior probability for a specific array a to belong to a specific bicluster b:

$$P(a.B_b = 1) = \begin{cases} \pi_{bgr,} & \text{if a.B}_\text{b} = 0 \\ 1-\pi_{bgr,} & \text{otherwise} \end{cases} \tag{5.11}$$

While this prior has not been explicitly used in any of the presented results in the result section, it could be used in a similar way as the $P_2(g.B_b)$ prior, namely to introduce expert knowledge about which arrays are more likely associated with arrays.

(3) Prior for the array identifiers P(a.ID). Each array is given a unique identifier a.ID and in principle a prior distribution can be defined over these arrays. However, the use of such prior is very limited and for the *Pro*Bic model this prior distribution is therefore chosen uniform.

(4) Prior for the model parameters P(θ). The *Pro*Bic is parameterized by a set of parameters ($\hat{\mu}, \hat{\sigma}$), one set (μ_{ab}, σ_{ab}) per array a and per bicluster b. Biological and expert knowledge

can be introduced in the model through proper prior distributions. A straightforward decomposition of the prior is the product over all the arrays and over all the biclusters. Based on this composition, the individual distributions P($\mu_{a,b},\sigma_{a,b}$) can now be chosen such that they are conjugate to the expression level CPD.

$$\begin{aligned} P(\theta) &= P(\hat{\mu},\hat{\sigma}) \\ &= \prod_{a\in A}\prod_{b\in B} P(\mu_{a,b},\sigma_{a,b}) \end{aligned} \tag{5.12}$$

Any member of the exponential family is a conjugate prior distribution to Equation 5.12, for example, the Normal, scaled-inverse- χ^2 , Gamma, Inverse Gamma and Normal-Gamma distributions are all conjugate to the Normal distribution. In practice, the Normal-Inverse-χ^2 distribution is chosen as the prior. The Normal-Inverse- χ^2 distribution is very useful by slightly more complicated prior. It defines a prior distribution directly for the mean and standard deviation in the following way:

$$\sigma \sim Inv-\chi^2(\nu_0,\sigma_0^2) \tag{5.13}$$

$$\mu \mid \sigma \sim N(\mu_0, \sigma^2/\kappa_0) \tag{5.14}$$

This leads to the following formal distribution of the prior which is parameterized by $(\mu_0,\kappa_{0,}\nu_0,\sigma_0^2)$:

$$P(\mu_{a,b},\sigma_{a,b}) = \sigma^{-1}(\sigma^2)^{-\nu_0/2+1}\exp(-\frac{1}{2\sigma^2}[\nu_0\sigma_0^2+\kappa_0(\mu_0-\mu)^2]) \tag{5.15}$$

The posterior distribution $P(\mu_{a,b},\sigma_{a,b})\cdot P(X)$, where X represents the expression data, is again a Normal-Inverse- χ^2 distribution characterized by:

$$\mu_p = \frac{\kappa_0}{\kappa_0+n_X}\mu_0 + \frac{\kappa_0}{\kappa_0+n_X}\frac{s_X}{n_X} \tag{5.16}$$

$$\kappa_p = \kappa_0 + n_X \tag{5.17}$$

$$\nu_p = \nu_0 + n_X \tag{5.18}$$

$$\nu_p\sigma_p^2 = \nu_0\sigma_0^2 + (n_X-1)ssq + \frac{\kappa_0 n_X}{\kappa_0+n_X}(\frac{s_X}{n_X}-\mu_0)^2 \tag{5.19}$$

5.4.2.3 Posterior distributions

The following log(posterior) distribution is obtained by combining the prior (Equation 5.12) and the likelihood function (Equation 5.1):

$$\log(posterior) = \sum_{a\in A}\sum_{b\in B}\log P(\mu_{a,b},\sigma_{a,b}) + \sum_{e\in E}\sum_{b\in B_e^i}\frac{\log P^*(e.level \mid \mu_{a,b},\sigma_{a,b})}{\# iset(B_e^i)} + \sum_{b\in B}\sum_{a\in A}\log P(a.B_b) + \sum_{a\in A}\log P(a.ID) + \sum_{b\in B}\sum_{g\in G}\log P_2(g.B_b) + \sum_{g\in G}\log P_1(g.B) \tag{5.20}$$

The hidden variables in Equation 5.20 are the gene to bicluster assignments g.B$_b$ and the array to bicluster assignments a.B$_b$. The model parameters are the Gaussian distribution parameters $(\hat{\mu},\hat{\sigma})$, with one set of $(\mu_{a,b},\sigma_{a,b})$ values per array a and bicluster b.

The calculation of the full posterior distribution over all possible assignments of hidden variables is exponential in both the number of genes and arrays and thus computationally intractable. However, the computational cost can be drastically reduced by only deriving the MAP solution of this function w.r.t. the model parameters and the hidden variables. For the ProBic model, we will outline an Expectation-Maximization strategy in Section 5.4.4.

5.4.3 Query-driven biclustering in *Pro*Bic

The *Pro*Bic model can be used to perform query-driven biclustering by using the prior distribution explained in Section 5.2.2.2. Starting from a set of seed genes which are of interest to a biologist, the model searches for a bicluster with a similar expression profile as the seed genes in a subset of arrays.

The Normal-Inverse-χ^2 distribution is used for all results presented in this chapter. By choosing the prior mean $\mu^0_{a,b}$ as the sample average μ_a^{query} for the expression values defined by the set of seed genes for each array a, the algorithm will identify a biclusters b around the expression profile of these seed genes. The prior standard deviation $\sigma^0_{a,b}$ is default chosen to be smaller than the background standard deviation σ_a^{bgr} by a fraction f_{bcl} in order to identify tight bicluster profiles. The user can however define priors that are more stringently or more loosely tied around the expression profile of the seed genes by varying $\sigma^0_{a,b}$. The other two parameters ν_0 and κ_0 in the distribution also reflects how much flexibility the bicluster is allowed around the expression profile of these seed genes.

From a biological perspective, this definition of query-driven biclustering together with the prior parameter setting has an interesting property: since the user can define the tightness of the bicluster around the expression profile around the seed genes, it is possible that some seed genes can be removed from the biclusters if they do not fit the biclusters profile well while the bicluster is still around the expression profile of other seed genes. This is very useful in situations where a biologist is interested in identifying all the genes that are involved in a specific pathway and where a known set of seed genes is available that re expected to be part of that pathway, meaning that most of these seed genes are indeed in the pathway but some are outliers that do not belong to the pathway. The application on the *E.coli* compendium will show examples like this in Section 5.6.2.

5.4.4 Learning the model

5.4.4.1 Expectation-Maximization (EM) algorithm

An attractive approach that is specifically tailored to maximize the posterior in case of missing values is Expectation-Maximization (EM). It deals with missing or hidden values by replacing them by an estimated value or by an estimated distribution over all values given the current model parameters and then it recalculates the model parameters given these estimated values. This process is repeated until the values converge. The EM procedure is guaranteed to converge to a local optimum under fairly general conditions [103, 104].

We will apply a hard-assignment EM approach. Hard assignment means that a single value is assigned to the hidden variables during the expectation step rather than a distribution over all possible values as in the case of soft-assignment EM.

The EM iterates the following steps until convergence:

- Maximization step: maximize posterior w.r.t the model parameters, keeping the hidden variable (i.e. the gene- and array-bicluster assignment G.B and A.B) fixed.
- Expectation step: find the expected values for the hidden variables G.B and A.B, keeping the current model parameters fixed. This step can be split into two substeps. In these substeps the gene-bicluster assignment G.B and the array-bicluster assignment A.B are alternatively fixed and the posterior is maximized w.r.t the other attributes (A.B and G.B respectively). This approach leads to a further decomposition of the log-posterior in each substeps into a number of terms that can independently optimized:

  Expectation step 1: maximize posterior w.r.t gene-bicluster attributes G.B, keeping model parameters $\hat{\mu}, \hat{\sigma}$ and A.B fixed.

Expectation step 2: maximize posterior w.r.t array-bicluster attributes A.B, keeping model parameters $\hat{\mu}, \hat{\sigma}$ and G.B fixed.

5.4.4.2 EM initialization

Any EM algorithm starts the optimization procedure from an initial assignment of the hidden variables. For *Pro*Bic model, these are the gene to bicluster and array to bicluster assignment attributes that require an initialization. Once these attributes are known, the algorithm starts with a Maximization step in which the parameter matrices $\hat{\mu}$ and $\hat{\sigma}$ are estimated. Several strategies can be applied to initialize the bicluster assignment attributes G.B and A.B:

- Random initializations.
- Initialization around known gene clusters, know biclusters or around a set of seed genes.
- Initialization with the complete set of genes and arrays.

When *Pro*Bic model used to perform query-driven biclustering, initialization around a set of seed genes is chosen to be the initialization procedure.

5.4.5 Time complexity of the EM algorithm

For discussing the time complexity of *Pro*Bic, the following notation is used: G is the number of genes, A the number of arrays, B the number of biclusters and I the number of iterations.

For the maximization step (Section 5.4.4.1), model parameters $\mu_{a,b}$ and $\sigma_{a,b}$ are calculated for every array-bicluster combination. Each calculation is a summation over the number of genes in that bicluster, so an overestimation of the time complexity of this step is $O(A \cdot G \cdot B)$.

Expectation step 1 optimizes the posterior for all A.B combinations and consists of two parts: the first part is to precalculate the log likelihood $\log P(e.level \mid \mu_{a,b}, \sigma_{a,b})$ given G.B is fixed and the second part is the calculation of the actual score for each array involves an Apriori-like algorithm that generates the candidate a.B vectors. Therefore, a total time complexity $O(A \cdot G \cdot B \cdot \log(B) + A \cdot B^{N+M})$ is obtained for Expectation step 1. Analogously, the time complexity for Expectation step 2 is of order $O(A \cdot G \cdot B \cdot \log(B) + A \cdot B^{N+M})$. Assuming that there are I iterations until convergence and that each of the Expectation steps is followed by a Maximization step, the time complexity of the complete algorithm is $O(I \cdot (A \cdot G \cdot B \cdot \log(B) + (A+G) \cdot B^{N+M}))$.

We observe that the worst case time complexity is linear in the number of genes and arrays and polynomial in the number of biclusters. However, for determining the average time complexity, we notice that only a small fraction of the biclusters actually overlap in practice. A conservative estimate is that the number of overlaps is of the same order of magnitude as the number of biclusters, so the number of possible gene-bicluster vector g.B is *O(B)*, leading to an average time complexity that is quadratic in the number of biclusters: $O(I \cdot (A \cdot G \cdot B \cdot \log(B) + (A+G) \cdot B^2))$

The identification of a single bicluster in a synthetic data set of 500 genes and 200 arrays, take about 1 minute on a dual Opteron 250 server with 2GB RAM (using only one core) using default settings for the convergence speed.

5.5 Methods

5.5.1 In house simulated data set

The simulated data used in this section is modelled according to the *Pro*Bic model assumptions. Background values were sampled from a Normal distribution $N(\mu_{bgr} = 0, \sigma_{bgr} = 1)$ for all arrays. The expression values $e_{g,a}$ that are part of a single bicluster b were sampled from the distributions $N(\mu_{a,b}, \sigma_{a,b})$ (meaning a separate distribution per array a and per bicluster b). The values for $\mu_{a,b}$ were chosen from a Normal distribution $N(0, \sigma_{\mu})$ where $\sigma_{\mu} = 1$ unless specified otherwise. In the following sections, we use the format n x m to indicate a simulated dataset with n genes and m arrays.

5.5.2 Prelic data set

In Dhollander *et al.* (2007) [97], a systematic comparison of query-driven biclustering algorithms was performed using various simulated gene expression datasets defined by Prelic *et al.* [105]. The tested algorithms were ISA [96], Gene Recommender [106] and QDB [96, 97].

Prelic et al. have investigated various experimental settings. Datasets were generated containing (1) biclusters with constant expression values, called scenario 1 and (2) datasets containing biclusters with constant columns and an additive bicluster model, called scenario 2. The datasets are 100 genes by 50 arrays (scenario 1) and 100 genes by (100…110) arrays (scenario 2) respectively. For each of these scenarios, two settings were tested. In setting A, noise is added to the gene expression values and a model without overlap between the biclusters is used. The standard deviation σ of this distribution represents the noise level. In setting B an increasing amount of overlap between the biclusters is tested, varying between 0 (overlap) and 10 (50% overlap) for both genes and arrays of the biclusters. In each dataset 10 biclusters are implanted of fixed size 10x10 for setting A and of increasing size (10+i)x(10+i) in setting B where i represents the amount of overlap. More details on the experimental setup can be found in Prelic et al. We used a gene match score (a detailed explanation of the match scores can be found in Prelic et al.) to assess the performance of the algorithms.

5.5.3 Biological data set

For the application of *Pro*Bic on biological data sets, a cross-platform compendium for *Escherichia coli* dataset was used. The compendium is constructed from *E.coli* microarray datasets which are publicly available from three major databases: Stanford Microarray Database (SMD) [4], Gene Expression Omnibus (GEO) [3] and ArrayExpress [5]. The compendium contains a collection of 870 publicly available microarrays for diverse experimental conditions. Both cDNA microarray and Affymetrix datasets are normalized and combined into a single compendium. Several experiments were run to evaluate the performance of the algorithm.

5.5.4 Calculation of enrichment scores

For the genes in a bicluster, both functional enrichment and enrichment in targets of the known regulators were calculated by the hypergeometric distribution (Section 4.2.3.3) (0.05 significance level, FDR correction). In order to evaluate the condition content of the biclusters, we constructed a functional category for the conditions (Appendix B). Consequently, the quality of the condition selection was assessed using this functional category and the hypergeometric distribution (0.05 significance level, FDR correction).

5.6 Results

An evaluation of the *Pro*Bic algorithm was performed on simulated gene expression data sets to investigate the behavior of the algorithm under various parameter settings and input data. We tested the automatic derivation of the optimal settings of some parameters for a specific data set and the algorithm’s ability to handle missing values. A comparison between *Pro*Bic and a number of state-of-art biclustering algorithms was performed using previously published artificial data [105] to avoid any bias in the results. Finally, *Pro*Bic has been applied on an *E.coli* expression data compendium using the query-driven setting.

5.6.1 Artificial data set

5.6.1.1 Identification of the number of biclusters

The *Pro*Bic model in principle requires the number of biclusters to be defined upfront. However, the actual number of biclusters in reality is difficult or even impossible to estimate.

Firstly, the optimal number of biclusters depends indirectly on the amount of noise in the data. As a model contains more biclusters, also more spurious biclusters will be added that contain a random set of accidently correlated genes. The ’optimal’ number of biclusters is the set of biclusters that contains as few spurious biclusters as possible while avoiding to miss any real biclusters. Depending on the research goals, the relative importance of these two numbers (which are in fact the number of false positive and false negative biclusters) will be different and this will be reflected in the parameter settings of the model. Secondly, interesting biclusters exist at different resolution level [97]. A researcher can both be interested in a small, tightly correlated bicluster or a large, more loosely correlated bicluster that is a superset of the smaller bicluster, depending on his/her research goals. The optimal number of biclusters will therefore necessarily depend on the parameter settings that the researcher applies in a particular study.

This problem recurs in many clustering and biclustering settings [93, 100] and often the number of (bi)clusters is estimated upfront either by the author, e.g. using expert knowledge [93], or by applying other methods that perform an estimation of this number. The *Pro*Bic model can use any of these techniques to estimate the number of biclusters in the data set. However it can also automatically select the optimal number of biclusters in the dataset for a given parameter setting of the model. The model is initialized with N_{max} (empty) biclusters where N_{max} is chosen larger than the number of actual biclusters in the dataset.

An analysis was performed for simulated 500x200 datasets with varying degrees of noise that were implanted with 7 biclusters each containing 50 genes and 50 arrays. The model score was calculated for models with N biclusters where N varied between 1 and 10. Figure 5.2 shows the model score in function of the number of biclusters in the model for an artificial dataset containing seven biclusters. In an iterative procedure, a model is constructed with 1, 2,…optimized biclusters respectively where the remaining biclusters are empty. We observe that the model score increases up until seven identified biclusters, after which the model score does not increase anymore: no more score improvements are achieved since the score would decrease by adding a non-empty bicluster to the model. As soon as the real number of biclusters in the data is obtained, the model will therefore only add empty biclusters to the model.

As discussed, the optimal number of biclusters will vary in function of the different parameter settings. For example, if no penalty is introduced for adding expression values to a bicluster ($\frac{\pi_{bicl}}{\pi_{bgr}} = 1$), the model will select as many biclusters as possible since adding a bicluster increases its degrees of freedom to fit the data.

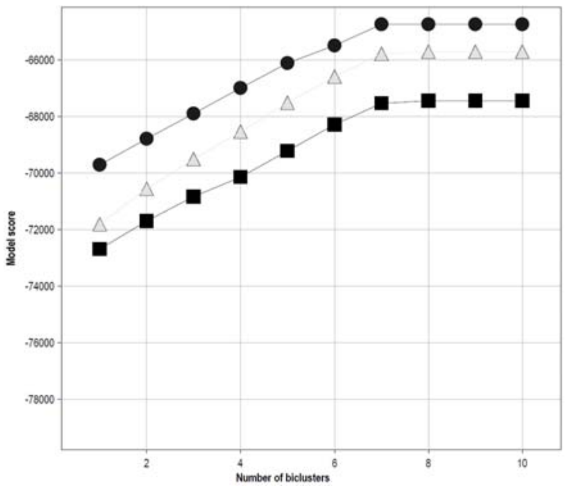


Figure 5.2: Identification of the number of biclusters in a simulated dataset for varying degrees of noise. For simulated 500x200 datasets with seven 50x50 biclusters and for varying noise levels, the model score is shown (Y-axis) for a model with N biclusters (X-axis). Legend noise level: circle: 0.4; triangle: 0.6; square: 0.8.

5.6.1.2 *Optimal model parameter settings*

The *Pro*Bic biclustering model has several parameters that potentially affect the biclustering results. While it is important for real life application that parameters can be fine-tuned by the researcher, sensible default values need to be defined that are broadly applicable on a wide range for each of the parameters of data sets.

Parameter $\frac{\pi_{bicl}}{\pi_{bgr}}$ ratio (Section 5.4.2.2):

The optimal ratio between π_{bicl} (the probability that an expression level is in a bicluster) and π_{bgr} (the probability if the same expression level is in the background, see supplementary material S1 for the detailed explanation for these two parameters) indicates how many times more likely it is for an expression value to be part of a bicluster compared to the background distribution before it is actually added to the bicluster. In order to determine this optimal ratio, we assume there exists one or more known clusters of genes in the data set that are co-expressed in a subset of conditions. This assumption is not too stringent as in most practical biological situations, such known sets of genes exist (e.g. a set of operon genes). If such a set would not be available, standard clustering techniques can also be used to indentify one or more these clusters.

Figure 5.3 illustrates how to determine the optimal ratio in two different circumstances: if the conditions are known in which the genes are coexpressed (i.e. we have a set of known biclusters), for every array a, the difference in score (defined as δ) is calculated between the situations where either that array is part of the bicluster or where it is part of the background. We then solve a classification problem where the optimal threshold is determined for the δ's that minimizes the global error rate (= the product of the false positive rate and the false negative rate) for classifying the arrays in either the background or in the known bicluster. On the other hand, if the conditions are unknown (i.e. in an *E.coli* compendium), after calculating the δ's for all the arrays, a plot of sorted δ's is made in which a clear cutoff point between arrays wit high and low δ's can often be determined visually, as shown in Figure 5.3 (right).

Parameter β:

The overall probability that a gene is in any bicluster (defined as β) indirectly affects the size (in number of genes) of the biclusters. Figure 5.4 illustrates the effect of varying β on the bicluster size for a 500x200 data set with one 100x100 bicluster which is composed of three parts:

- Expression values for 20 genes, sampled from $N(\mu_a, 0.2)$.
- Expression values for 30 genes, sampled from $N(\mu_a, 0.5)$.
- Expression values for 50 genes, samples from $N(\mu_a, 1.0)$.

We expect that lower β values will lead to a smaller bicluster (keeping only the most correlated genes) and this is confirmed in Figure 5.3. A single bicluster is identified with the *Pro*Bic model and the number of genes in the bicluster decreases successively from 100 to 50 to 20 for decreasing β as predicted. We confirmed that the identified bicluster indeed corresponded to the original 100x100, 50x100 and 20x100 biclusters respectively.

The Normal-Inverse-χ^2 priors parameters:

The Normal-Inverse-χ^2 priors have four parameters: μ, σ, κ, ν per array and per bicluster. For the background normal distribution, the parameters should be informative in settings where there is no prior knowledge about the background distributions. Therefore, the four values are set as 0, 1, 0, 1 respectively. For the bicluster distribution, in the query-driven biclustering setting, it is desirable to create strong prior distribution with mean and standard deviation around the mean and standard deviation of the seed genes, leading μ and σ are set as described in Section 5.2.2.2. κ is set as a relatively big value, for example 100000 to guarantee the bicluster staying around the seed genes. Finally, the parameter ν is the relative weight of the prior compared to the data. We chose 1000 as the value for it to keep the balance of the prior and the data itself.

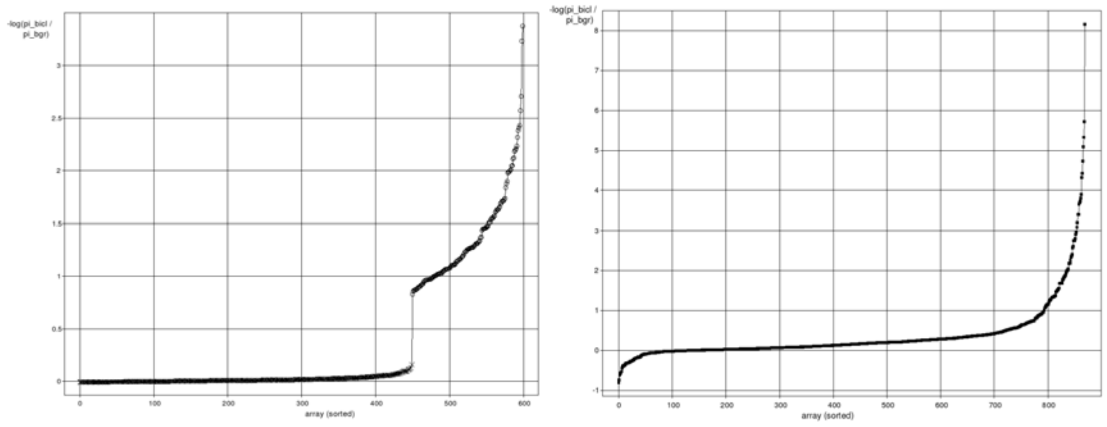


Figure 5.3: Illustration of determining the optimal ratio for $\frac{\pi_{bicl}}{\pi_{bgr}}$*. (Left) For a simulated 500x200 data set with three 50x50 biclusters (noise level 0.2). The plot shows the* δ *'s over all arrays (multiple with the number of biclusters) for a set of genes that are known to be co-expressed in a number of arrays. Large* δ *'s indicate that the genes are likely to be in the bicluster distribution for that condition rather than in the background distribution. The* δ *threshold that classifies these two sets of arrays best according to the ratio is 0.5, leading to an optimal ratio of* $\log\frac{\pi_{bicl}}{\pi_{bgr}} = -0.5$ *. (Right) In the E.coli compendium for the set of genes that are known to be regulated by FNR, the plot shows sorted* δ *'s over all arrays. Based on this plot, good values for* $\log\frac{\pi_{bicl}}{\pi_{bgr}}$ *are between -0.5 and -0.8.*

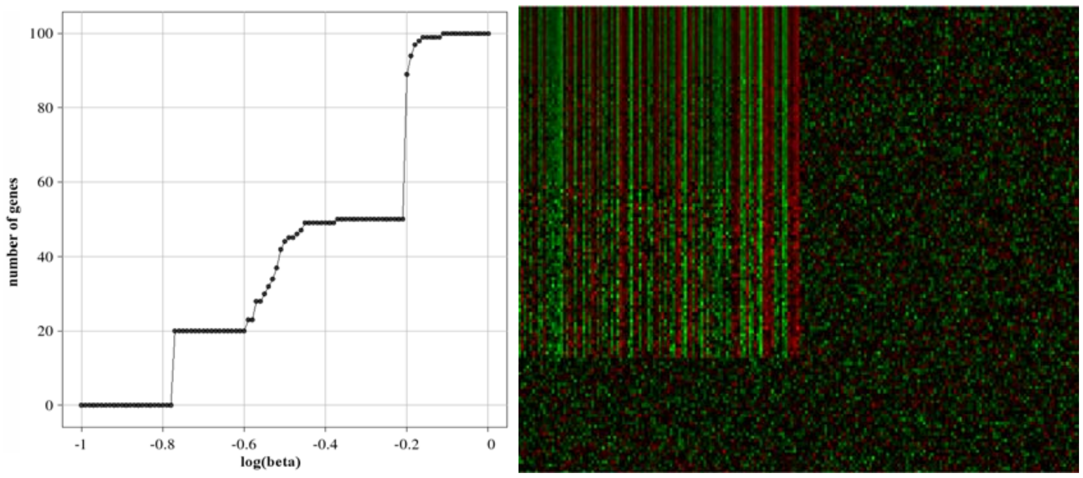
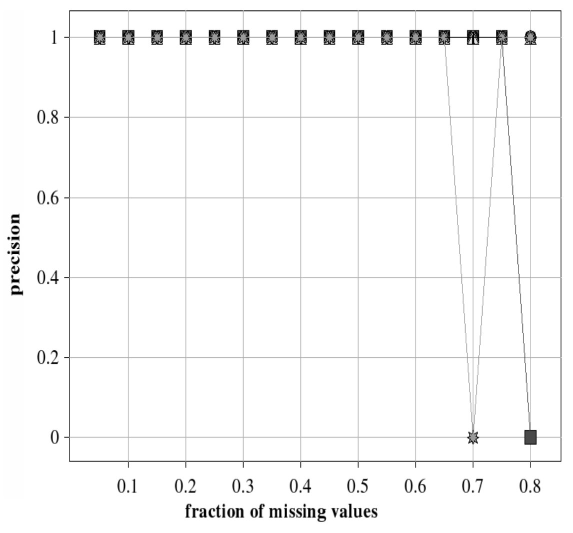


Figure 5.4: Illustration of the effect of the β parameter on the number of inferred genes in a bicluster. A 500x200 simulated data set contains one 100x100 bicluster that is composed out of three parts. (Left) Effect of β parameter on the number of genes in the inferred bicluster. (Right) Heatmap of a part the simulated data set containing the 100x100 bicluster with varying degrees of noise.

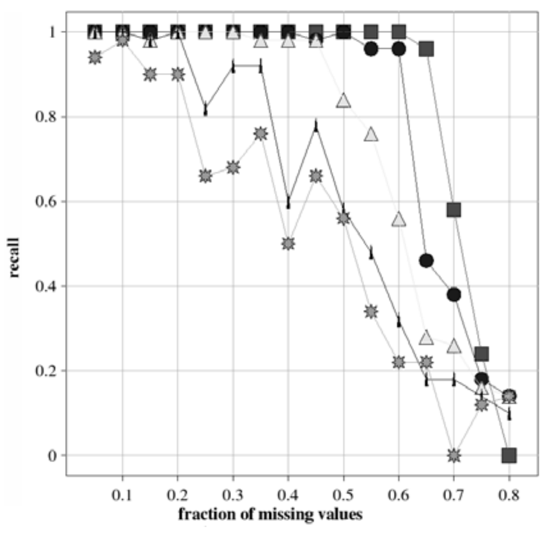
5.6.1.3 Noise and Missing values robustness

To assess the effects of noise and of missing values on the bicluster identification, a 500x200 simulated data set was constructed as previously described with three 50x50 non-overlapping biclusters. The percentage of missing values in the data set varied between 0% and 100% and the bicluster noise level was varied between 0 (no bicluster noise) and 1 (bicluster noise = background noise). The resulting effect on the precision and recall for both the genes and the arrays is shown in Figure 5.5.

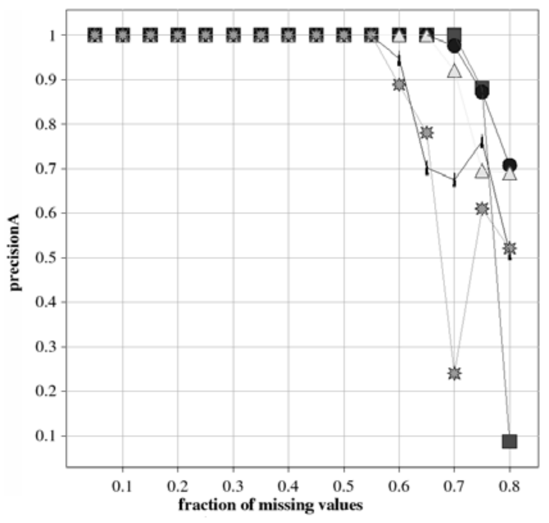
Figure 5.5a and 5.5c show that for lower bicluster noise levels (< 0.5), the presence of up to 60% of missing values in the data set does not interference with the deletion of the true bicluster genes and arrays (a precision and recall of about 100% is obtained for the genes and arrays). As can expected the model sensitivity to missing values increases with the bicluster noise level, but even in the presence of high noise level (1.0), the presence of up to 20% missing values does not considerably deteriorate the algorithms recall and precision (recall levels of 0.9 and 0.88 for the genes and arrays respectively).



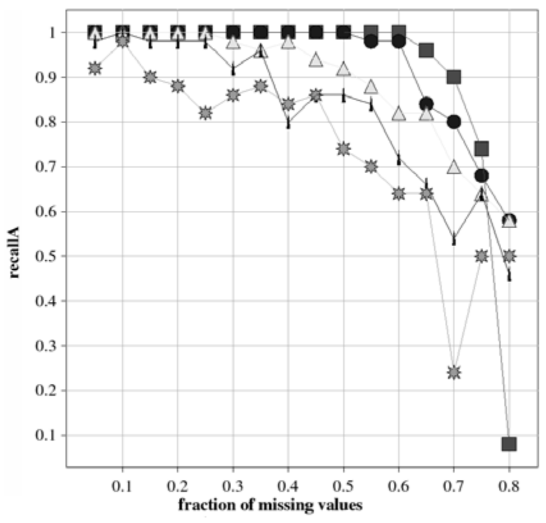
(a) Precision genes



(b) Recall genes



(c) Precision arrays



(d) Recall arrays

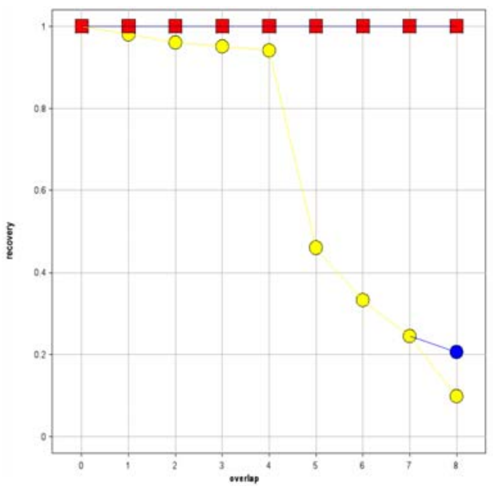
Figure 5.5: Noise and missing value robustness of ProBic. ProBic performance on a 500x200 simulated data set (containing three 50x50 biclusters) with varying degrees of noise and missing values. The x-axes show the percentage of missing values (between 0% and 100%) in the data set and the plots indicate the average precision and recall for the genes and arrays respectively. Noise level legend: square: 0.2; circle: 0.4; triangle: 0.6; small triangle: 0.8; star: 1.0.

5.6.1.4 *Comparison of ProBic with state-of-art query-driven biclustering algorithms*

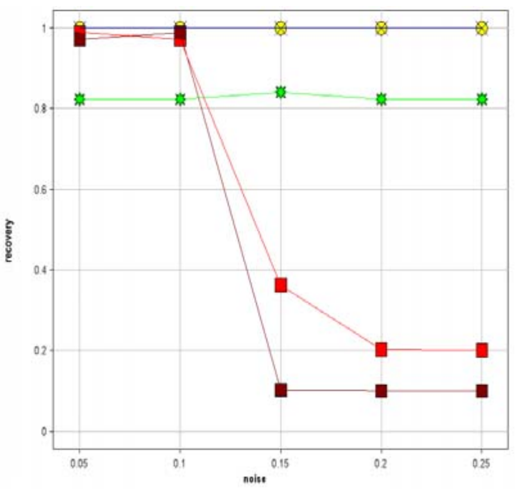
Since many biclustering algorithms are based on different statistical measures of 'good' biclusters, a fair comparison between the algorithms is difficult and often biased. Therefore, we used previously published artificial data to avoid any bias in the results. In Dhollander *et al.* [97], a systematic comparison of query-driven biclustering algorithms was performed using various simulated gene expression data sets defined by Prelic *et al*. . The tested algorithms were ISA [96], Gene Recommender [106] and QDB [97].

Prelic et al. have investigated various experimental settings. Data sets were generated containing (1) biclusters with constant expression values, called scenario 1 and (2) data sets containing biclusters with constant columns and an additive bicluster model, called scenario 2. The data sets are 100 genes by 50 arrays (scenario 1) and 100 genes by (100…110) arrays (scenario 2) respectively. For each of these scenarios, two settings were tested. In setting A, noise is added to the gene expression values and a model without overlap between the biclusters is used. The standard deviation σ of this distribution represents the noise level. In setting B an increasing amount of overlap between the biclusters is tested, varying between 0 (no overlap) and 10 (50% overlap) for both genes and arrays of the biclusters. In each data set 10 biclusters are implanted of fixed size 10x5 for setting A and of increasing size (10+i)x(10+i) in setting B where i represents the amount of overlap. More details on the experimental setup can be found in [105]. We use gene match score (detailed explanation of the match scores can be found in [105]) to assess the performance of the algorithms.

Figure 5.6 shows the results of this analysis. For the noise experiments (setting A), we observe that *Pro*Bic performs well for low noise levels but more sensitively for higher noise levels compared to the other biclustering algorithm. Further investigation suggests that the some biclusters are not recovered is not due to any difference in the initialization and a resulting local optimum of the EM algorithm: experiments for query-driven biclustering with different types of initialization (complete data set and random initializations) lead to the same optima. There are several possible reasons for poor performance on noisy data set. One is due to the parameter setting of Normal-Inverse-χ^2 prior (Section 5.4.2.2) regarding to this data set. Analyses have shown that in this example, up to 20% variation for the bicluster relevance and recovery scores were seen in setting A for varying the σ_0 parameter which controls the tightness of a bicluster. Secondly, for these data sets, there are no real background genes. This leads to a disadvantage for biclustering methods with an explicit background model such as QDB and *Pro*Bic since the high percentage of biclusters confounds robust background estimation. For overlap experiments (setting B), *Pro*Bic outperforms other biclustering algorithms since it models the overlap explicitly and it is more robust for variations in the parameter of the prior distributions in this setting.

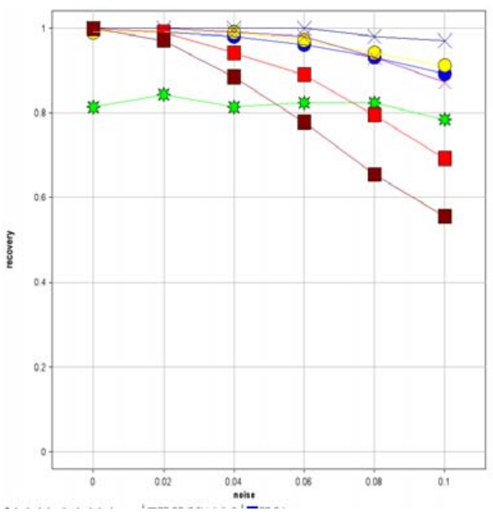


(a) Scenario 1, setting A

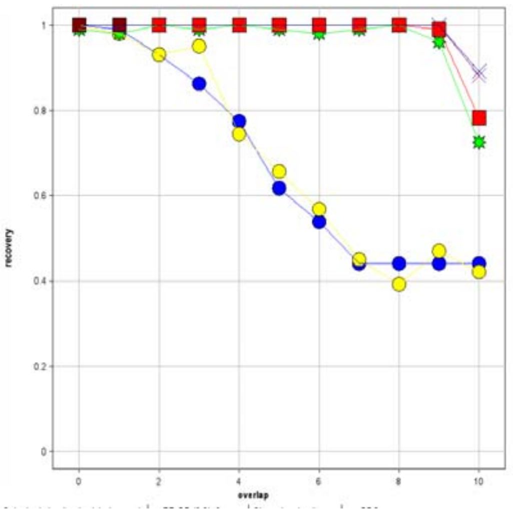


(b) Scenario 1, setting B

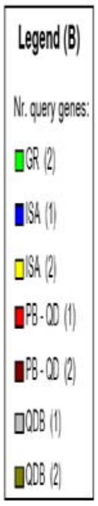
Legend (A)
Algorithm:
GR
ISA
PB - QD
PB global
QDB

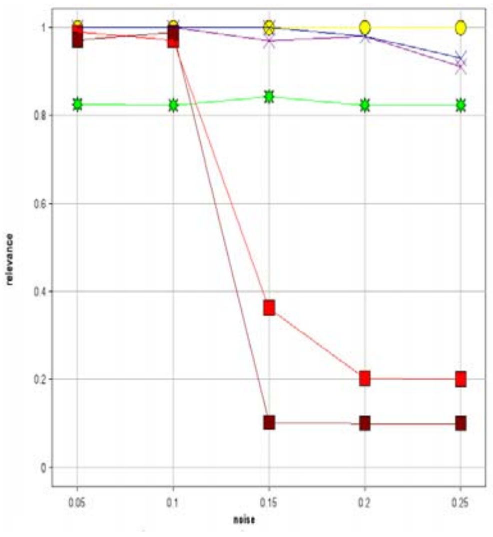


(c) Scenario 2, setting A

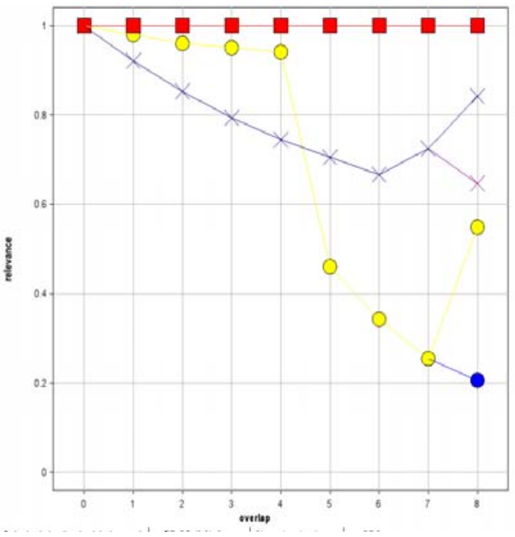


(d) Scenario 2, setting B

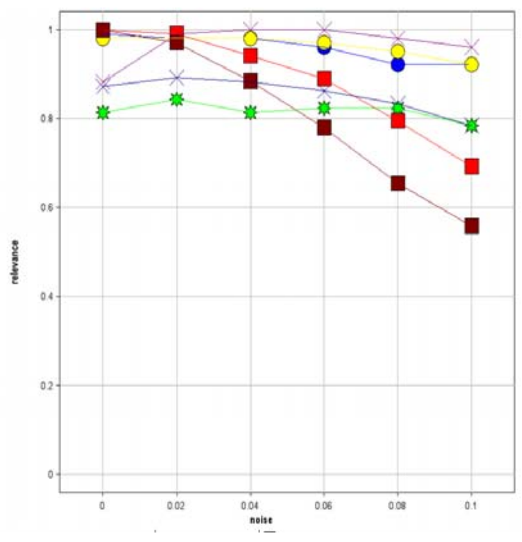




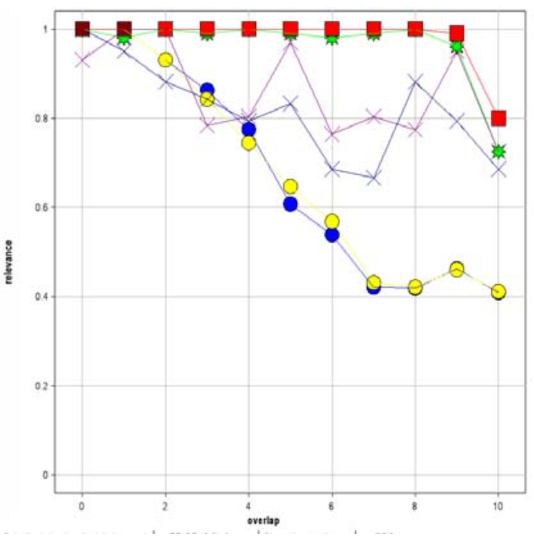
(f) Scenario 1, setting A



(g) Scenario 1, setting B



(h) Scenario 2, setting A



(i) Scenario 2, setting B

Figure 5.6: Performance comparison of ProBic and other query-driven biclustering algorithms for simulated data sets described in Prelic et al. ProBic is compared to Gene Recommander (GR), ISA and QDB. The bicluster recovery scores are shown in (a)-(d) and the bicluster relevance scores in (f)-(i). In panels (a), (c), (f) and (h) these scores are plotted in function of amount of noise and in (b), (d), (g) and (h) they are plotted in function of the overlap degree.

5.6.2 *E.coli* expression data compendium

Although synthetic data provide insight in the working and specific shortcomings of the implemented method, benchmarking on real experimental data is of essential importance to obtain a depiction of the biological relevance of the results. Therefore, we further evaluated the performance of *Pro*Bic on an *E. coli* compendium.

5.6.2.1 Single gene queries

A number of experiments were conducted to identify biclusters in the *E.coli* compendium based on single gene queries. Table 5.2 shows a selection of six known target genes of different regulators together with the identified biclusters, enriched regulators and gene ontology information. For verifying that the algorithm converges to similar biclusters for different target genes of known regulators, two different target genes were chosen as seeds for both the regulators LexA and CysB. The identified biclusters are shown in Figure 5.7. All of these biclusters show a very tight coexpression. It indicates that the algorithm performs well when taking single gene as seed.

Table 5.2: Overview of using single gene queries for query-driven biclustering. For each seed gene, the known regulator is shown together with the number of genes and conditions in the identified bicluster. The most enriched regulator and the most enriched GO term for the identified bicluster are shown associated with their p-values.

Known Reg.	**Seed gene**	**Nr. of genes in bicluster**	**Nr. of conditions in bicluster**	**Enriched Reg.**	**Enriched GO term**
LexA	*uvrB*	11	32	LexA (1.0394e-14)	DNA repair (1.26e-11)
LexA	*dinI*	8	20	LexA (9.8312e-06)	DNA repair (1.01e-10)
Fur	*fhuE*	20	75	Fur (1.3682e-23)	enterochelin (enterobactin) (1.43e-12)
CysB	*cysK*	12	97	CysB (2.3838e-20)	Sulfur metabolism (5.36e-12)
CysB	*cysD*	10	110	CysB (1.0744e-18)	sulfur metabolism (2.17e-15)
NtrC	*ddpX*	7	28	NtrC (7.0782e-10)	nitrogen metabolism (4.81e-04)

5.6.2.2 Outlier removal for query-driven biclustering

Biologists are often interested in the genes that are part of a specific pathway for which only a subset of the genes is known. In some cases, the set of seed genes contains one or more outliers, genes that actually do not belong to the pathway. In such cases, the query-driven biclustering should remove these outlier genes from the query while retaining the other genes in the final biclusters that is associated with the pathway of interest.

We simulated this situation for the *E.coli* compendium, by selecting two co-expressed seed genes that belong to the cyoABCDE operon *cyoA* and *cyoB*. The resulting bicluster is shown for this query in Figure 5.8. Two settings with outlier genes were investigated with respectively one and two additional randomly chosen genes, namely (*rfbB*) and (*rfbB*, *yniB*). The resulting biclusters are almost identical in both genes and conditions, as shown in Figure 5.8. The results show that even when 40% noisy seed genes, the correct bicluster is still identified and the noisy genes are removed from the resulting bicluster.

All resulting modules were most enriched for the Fur regulator. Fur is an inhibitor which has been implicated in the regulation of a large number of operons that encode enzymes involved in iron transport and is a known regulator for the cyoABCDE operon. Fur p-values ranged between 1e-29 and 1e-16, except for query (*cyoA*, *cyoB*, *rfbB*) where both Fur (p=1e-16) and ArcA (p=4e-17) were most enriched.

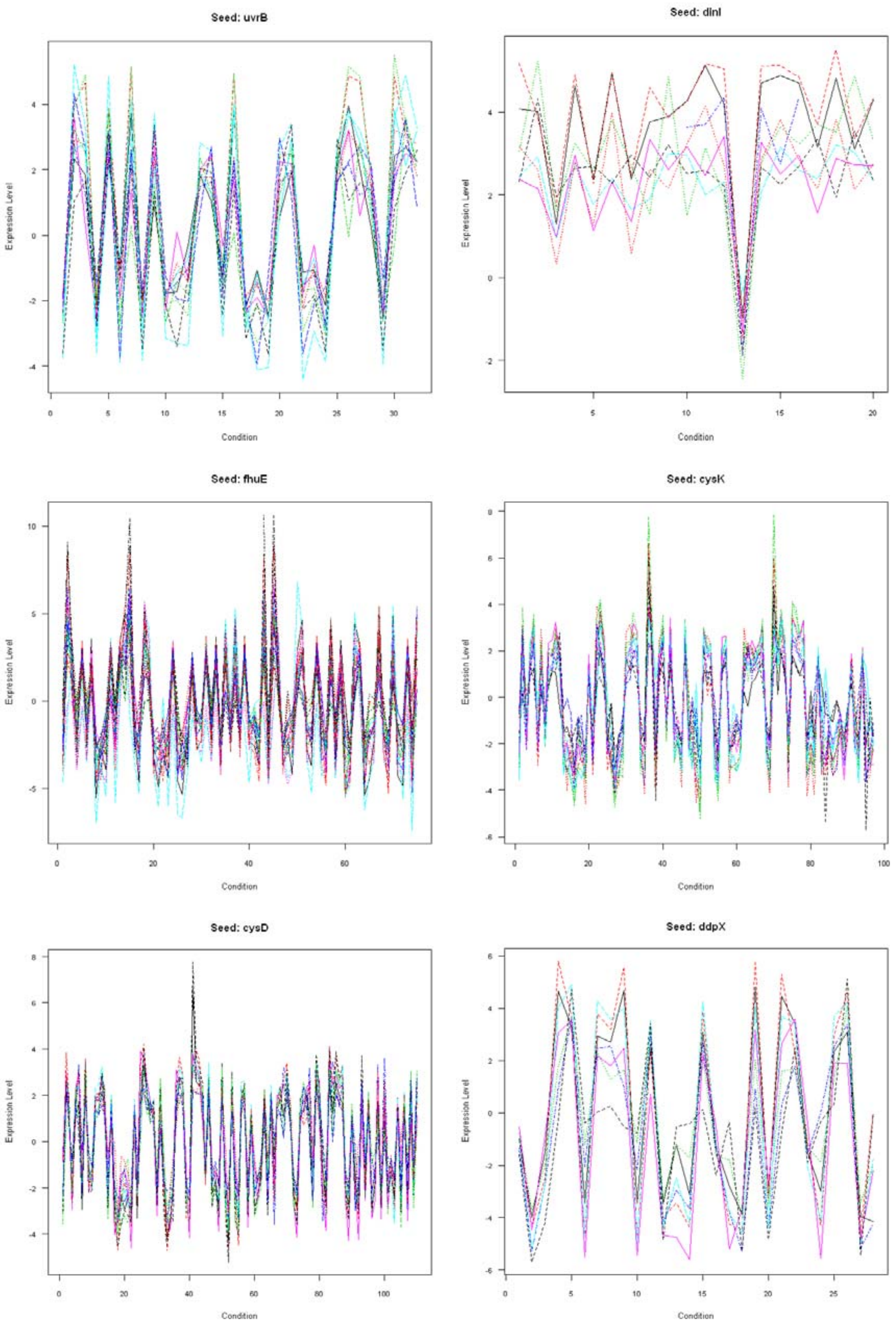


Figure 5.7: Identified biclusters expression profiles for different single gene queries. In each figure, the expression profiles of different genes are represented as different coloured lines. All bicluster results show high coexpression.

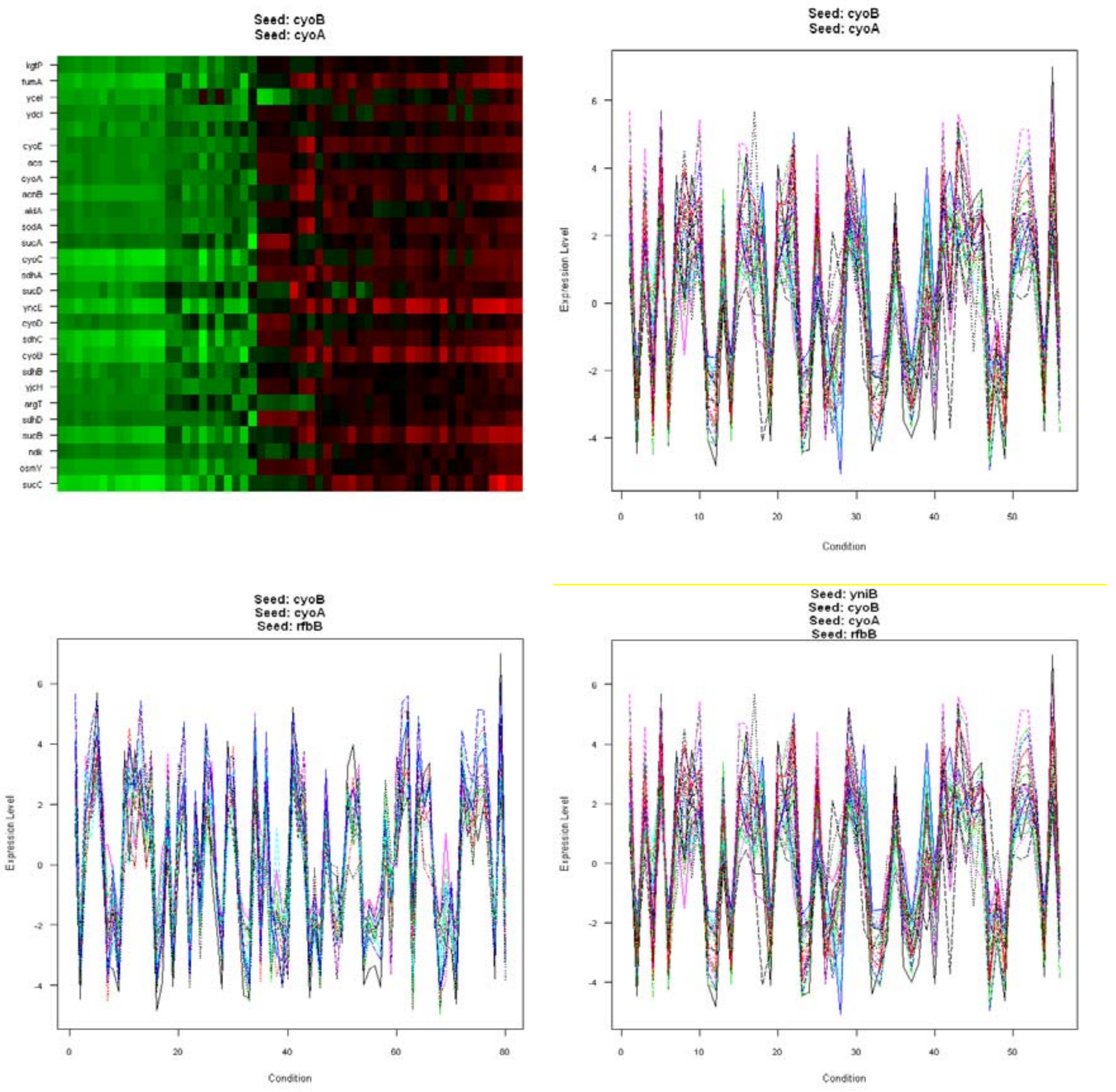


Figure 5.8: Illustration of the effect of a 'noisy query' using the E.coli compendium. The upper panels: heatmap and expression profile of the identified bicluster using two co-expressed seed genes cyoA and cyoB that belong to cyoABCDE operon. The lower panels: The resulting bicluster profiles for noisy queries with one (rfbB) and two (rfbB and yniB) additional randomly selected genes respectively to the original query cyoA and cyoB.

5.6.2.3 Validation of bicluster relevance

Next, we want to verify whether *Pro*Bic can successfully reproduce known regulatory interactions in RegulonDB.

Based on information in RegulonDB [107], two data sets of seed genes were constructed for the query-driven biclustering setting. We listed all the regulators and the combinations of regulators. For each regulator or combination of regulators, the genes regulated by these regulators were taken as seeds. In RegulonDB, the evidences associated to all regulatory interactions are classified as ‘strong’ or ‘weak’, based on the confidence level of the experiment or prediction that supports these regulatory interactions. A ‘strong’ evidence is assigned to a regulatory interaction when the experimental data provide high certainty of its existence; otherwise, it is a ‘weak’ evidence. For a regulatory interaction, DNA binding of purified transcription factor is a strong evidence. On the other hand, gene expression analyses and computational predictions are considered as week evidence [108]. For this study, we only consider the regulatory interactions with strong evidences and the first genes of the operon. A first data set contains non-overlapping seed genes. That is say so, if a gene is annotated to be regulated in EcoCyc by the combination of regulators R1 and R2 (the sets of genes regulated by them are called set R1 and set R2 respectively), namely the gene contain motif R1 and R2, the gene will not belong to either set R1 or set R2 since it is assumed that regulation of this gene is different under all conditions from those of the more complex motif. However, since it might well be that under certain conditions the genes in the different sets in the hypothetical example postulated above are coexpressed and consequently coregulated under certain conditions, we also constructed a second set of seed genes, in which the seed for different motifs can overlap. Both sets of seed genes are used for query-driven biclustering.

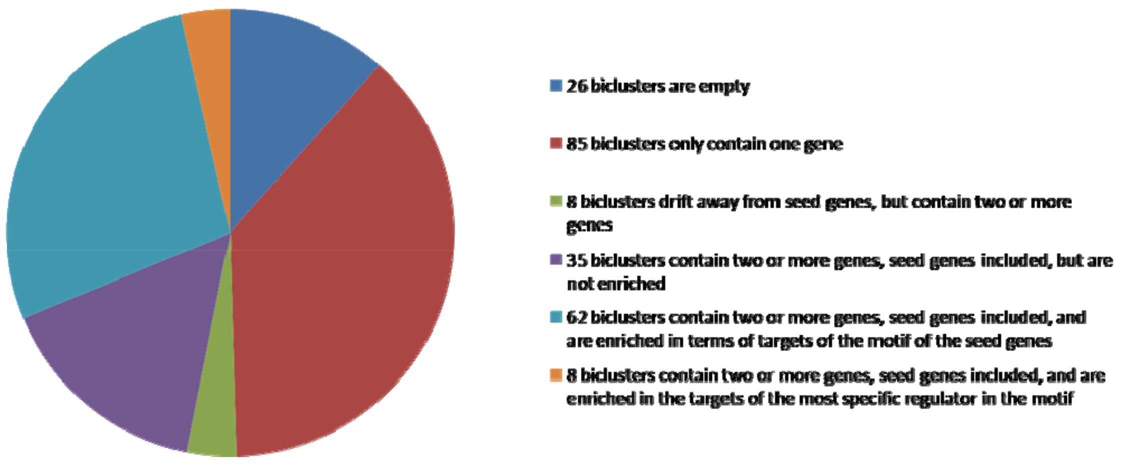


Figure 5.9: Summary of the resulting biclusters from the E.coli compendium using data set 1 as the set of seed genes.

Data set 1

The first dataset contains 224 different sets of seeds ranging in size from 1 gene to 32 genes. Figure 5.9 summarizes the resulting biclusters using this data set. For those seeds 113 biclusters were obtained which contained at least 2 genes, the rest was either empty or contained only 1 gene. Of those 113 biclusters there are 105 for which at least one of the seed genes is still contained within the bicluster, and hence are considered as possibly biologically relevant. These biclusters were further evaluated in terms of the genes and conditions within the bicluster.

For the 111 outputs of the algorithm which we do not further assess (i.e., those who lead to our definition of biologically irrelevant biclusters), we investigated whether the failure to find a meaningful bicluster is due to the quality of the seed genes, since the seed largely determines the outcome of the algorithm. Two criteria were considered concerning the quality of a set of seed genes: seed correlation and seed variance. Seed correlation represents the homogeneity of a set of seed genes. It is defined as the mean of Pearson correlation of each pair of genes in the set across all the conditions. Seed variance indicates how distinguishable a set of seed genes is from the background. It is defined as the variance of the mean profile of all the seed genes across all the conditions. Figure 5.10 shows the difference in seed correlation and seed variance between the good results and the bad results. As seen from Figure 5.9, good quality seed genes are the ones with less heterogeneity and with larger difference from the background.

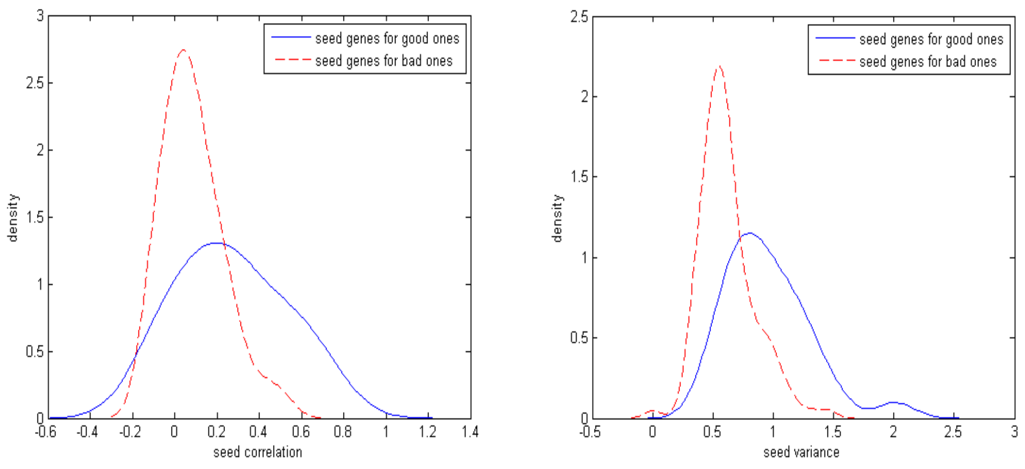


Figure 5.10: The difference of seed correlation and seed variance between the good results and the bad results. (Left) The correlations of the seed genes which lead to the bad results are much lower (around zero) than the ones leading to the good results. (Right) The variances of the seed genes which lead to the bad results are mostly low and closer to the background, while the seed genes leading to good results are more distinguishable from the background.

For the 105 remaining biclustering results which gave a sensible output in terms of the number of genes and the number of seeds included in the bicluster, we further evaluated the biclusters in terms of genes and conditions.

Particularly, each bicluster is associated with a set of regulators which are known to regulate the seed genes of the bicluster (called the motif). Consequently we assess whether the bicluster results are significantly enriched for targets of the regulators of the seed genes. For 62 of the 105 'good' biclusters the genes selected seem indeed to be significantly enriched in terms of targets of the motif of the seed genes. Whereas some biclusters are not significantly enriched in targets of the regulators making up the motif, they are significantly enriched in targets of the most specific regulator in the motif (i.e. the regulator regulating the least number of genes). This is indeed the case for 8 additional biclusters.

Remark that most of the biclusters which are not significantly enriched in the targets of the regulator making up the motif are larger in size in terms of the number of genes selected. This might indicate that the specific conditions which trigger the regulators making up the motif do not exist in the compendium and that hence the bicluster diverges into a more global one, still containing some of the seed genes but no longer enriched in targets for the regulators.

In order to evaluate the condition content of the bicluster we calculated the condition enrichment based on the functional category listed in Appendix B. All of the 70 biclusters are with biologically meaningful enriched condition categories.

Moreover, we assessed that if the seed genes for two biclusters are regulated by the overlapping motifs, the resulting biclusters also show overlap in the gene dimension of the biclusters as illustrated by Figure 5.11. It indicates that the algorithm can outcome meaningful gene selection and condition selection.

Data set 2:

The same analysis was applied on the second data set. Figure 5.12 summarizes the resulting biclusters. The second dataset contains 224 different sets of seeds ranging in size from 1 gene to 98 genes. For those seeds 122 biclusters were obtained which contained at least 2 genes, the rest was either empty or contained only 1 gene. Of those 122 biclusters there are 121 for which at least one of the seed genes is still contained within the bicluster, and hence are considered biologically relevant. For 70 of the 121 'good' biclusters the genes selected are significantly enriched in terms of targets of the regulatory program of the seed genes. 9 additional biclusters are significantly enriched in targets of the most specific regulator in the motif. All of the 79 biclusters are enriched biological meaningful conditions.

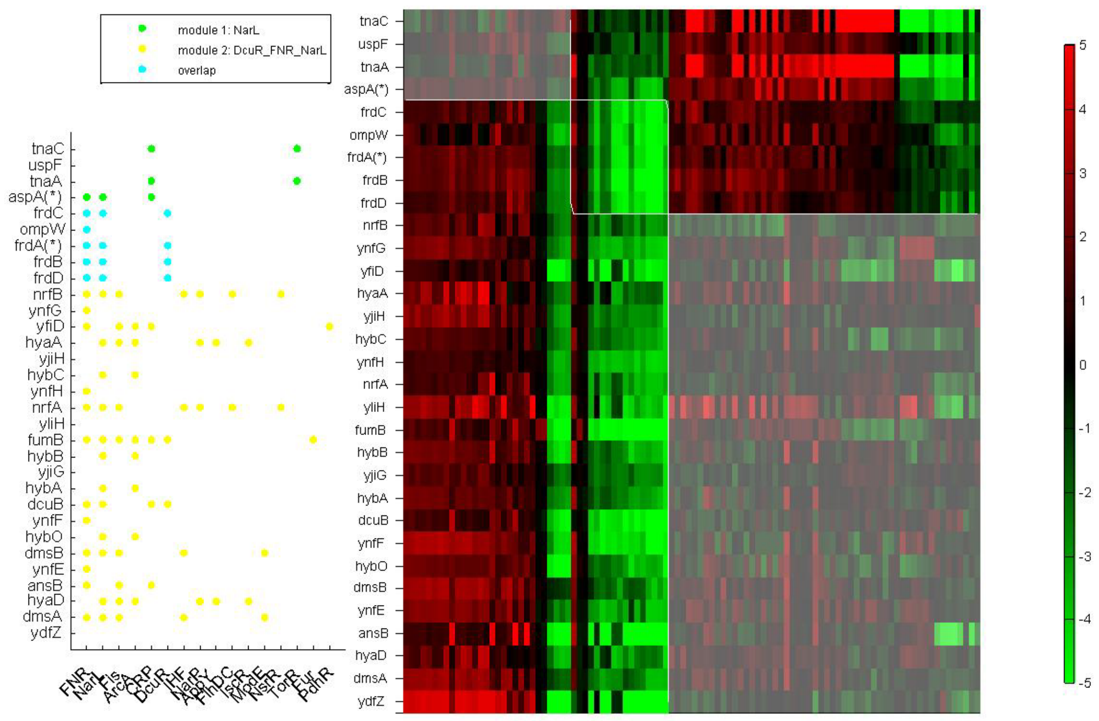


Figure 5.11: Illustration of Overlap biclusters. Bicluster 1 (upper right) takes the genes regulated by NarL as seeds. Bicluster 2 (lower left) takes the genes regulated by DcuR_FNR_NarL as seeds. The two resulting biclusters are overlap with gene frdC, ompW, frdA, frdB and frdD.

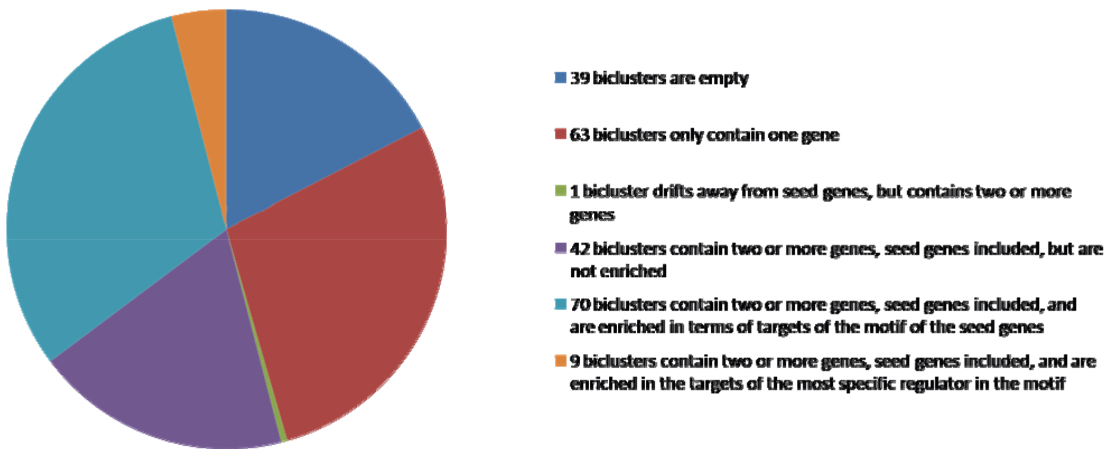


Figure 5.12: Summary of the resulting biclusters from the E.coli compendium using data set 2 as the set of seed genes.

5.7 Conclusions

*Pro*Bic is a PRM based model that identifies overlapping biclusters in a gene expression data set. It is an efficient biclustering algorithm in which expert knowledge can be introduced in the form of seed genes through a number of prior distribution parameters. Sound biclusters can be found around the seed genes. Experiments on the simulated data sets show the missing value robustness of the algorithm and explain the choice of certain parameters for the model. Prelic data set is used to compare *Pro*Bic with other biclustering algorithms. The results show that *Pro*Bic performs well for the low noise level and it outperforms other biclustering algorithms in the overlap settings. *Pro*Bic is also applied on a cross-platform *E.coli* compendium. In query-driven settings where the set of seed genes contain one or more outliers, the algorithm also performs well and is able to remove such outlier genes from the identified bicluster. This feature is particularly useful to biologist in practice. It is also verified that *Pro*Bic can successfully reproduce known regulatory interactions in RegulonDB.

Chapter 6

Normalization Influence

Inferring transcriptional networks from high-throughput expression data is becoming increasingly feasible as microarray experiments have gained common ground in many laboratories. Combining microarray expression obtained from different laboratories or from different platforms into one comprehensive microarray compendium requires proper normalization techniques to make all data mutually comparable.

In this chapter we discuss the influence of applying different ways to normalize cross platform microarray compendium on the results of different transcriptional network inference algorithms.

6.1 Introduction

A system biology approach aims at understanding the mechanisms of signal transduction and molecular interactions that give rise to the observed behavior, for example, understanding the underlying transcriptional networks. The introduction of microarray technology has made it possible to measure the gene expression levels of thousands of genes simultaneously. This introduced a new impulse to transcriptional network inference, namely inferring large-scale transcriptional networks based on measured expression data. Thus far, a multitude of algorithms have been proposed for inferring transcriptional network from a microarray compendium. Nevertheless, as discussed in Section 2.3.2, different normalization methods can be applied to generate a microarray compendium. Therefore, a question that arises is whether the results of algorithms are different when applying them on the compendia generated by different normalization methods. We choose certain representative network inference algorithms to study this question.

6.2 Methodology

6.2.1 Data sets

6.2.1.1 Microarray compendium

A cross platform expression compendium of *Escherichia coli* is used for this study (the same one as we used in Chapter 5).The compendium is constructed from *E.coli* microarray data sets which are publicly available in three major databases: Stanford Microarray Database (SMD) [4], Gene Expression Omnibus (GEO) [3] and ArrayExpress [5]. The compendium contains a collection of 870 publicly available microarrays for diverse experimental conditions. These microarrays are either from the same laboratories or with the same reference. In the compendium, a block of microarrays is defined as follows: a block can be all microarrays from the same laboratories (called an experiment block) or microarrays with the same reference (called a reference block). Within-array normalization procedure (Section 2.3.2) is applied on each of these microarrays individually.

6.2.1.2 Normalization strategy

To guarantee comparability of microarray experiments performed in different laboratories and on different platforms, a careful preprocessing of the compendium is needed. As mentioned in Section 2.3.2, preprocessing of a compendium includes within-array normalization step and between-array normalization step. Within-array normalization has been performed on each microarray. In this chapter, we aim to study the influence of different between-array normalization procedures on the result of different network inference algorithms. We simplify the term between-array normalization as normalization in this chapter.

Firstly, the expression data in the compendium are presented as expression log ratios of measurements in a test condition versus a reference condition, which can be seen as ‘a condition contrast’. A condition contrast is the set of all genes' expression log ratios, each of which is represents the log ratio of a gene’s expression values obtained in one particular experiment for which a test is compared to a reference sample. Within-array normalization of cDNA microarrays naturally results in log ratios. Affymetrix data have to be converted into log-ratios in the following way. For each set of experiment, one array is chosen as the reference condition. Then other arrays in that set can be seen as test conditions. The log-ratios are calculated by dividing the test conditions by the reference condition and then taking the logarithm. Thus, the expression data in the compendium are all in the format of log-ratios. The use of log-ratios makes data generated from different platforms comparable.

Secondly, normalization (Section 2.2.2.5) steps, such as mean (median) centring and variance rescaling should be applied on the compendium to remove variations across each individual array. Setting the mean of the individually normalized microarray data to the same value is equivalent to centring the distribution of these data to a certain predefined mean value (sample mean). Therefore, we call this approach mean centring. However sometimes the sample mean can provide an inaccurate estimate of the true centre of a distribution due to the presence of outliers in the sample data. In such cases, the median of the sample data can serve as a more robust estimate of the true distribution centre. When using the median the centring approach is called median centring. Different ways exist to perform mean (median) centring on the compendium. One way is to calculate the sample mean (median) across all the microarrays and then centre them to this sample mean (median). Alternatively, the sample means (medians) can be calculated per block and then each block of microarrays is centred with its corresponding mean (median). After normalizing the compendium by mean (median) centring, we may wish to bring the normalized data into a common scale in order to prevent any single microarray dominating when subsequent analyses are preformed on the compendium. We call this approach variance rescaling. Analogous as mean (median) centring, variance rescaling can also be performed differently, such as across all the microarrays, per reference group or per experiment group.

6.2.1.3 Data sets

For this study, we generate five differently normalized versions of the *E.coli* compendium as shown in Figure 6.1, to study the influence of different normalization methods on the results of network inference. We choose mean centring in this study.

We start from M-values and use different mean centring and variance rescaling strategies to generate the following five different versions of the compendium:

- CnRn: no mean centring and no variance rescaling, namely M-values.
- CnRt: no mean centring and variance rescaling with the variance calculated across all arrays.
- CrRn: mean centring with the means calculated per reference block and no variance rescaling.
- CrRr: mean centring with the means calculated per reference block and variance rescaling with the variance calculated per reference block.

- CrRt: mean centring with the means calculated per reference block and variance rescaling with the variance calculated across all arrays.

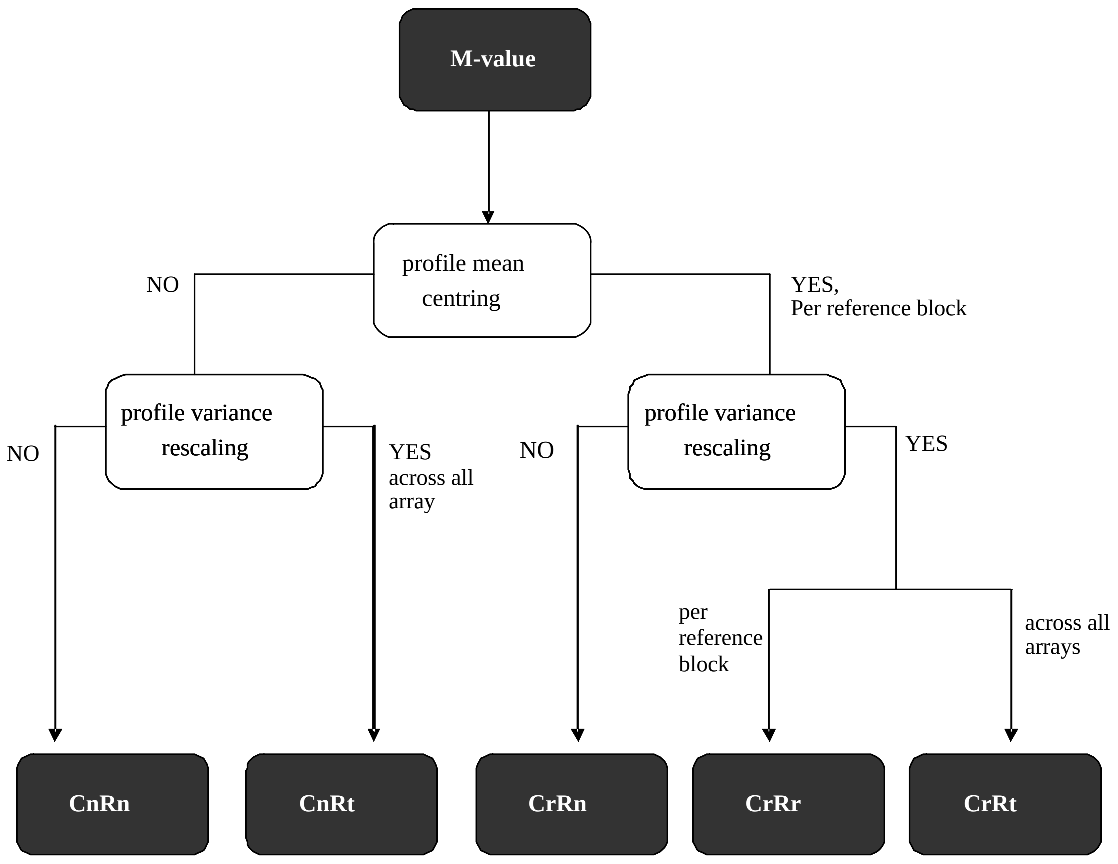


Figure 6.1: Five differently normalized versions of the E.coli compendium.

Apart from these five data sets, five random data sets are generated and used for result comparison. We first randomize the M-values, and then apply the corresponding mean centring and variance rescaling strategies on the randomized M-values, as shown in Figure 6.2.

The algorithms described in the next section are applied on each of these five versions of the *E.coli* compendium and as a reference also on the randomized data sets.

6.2.2 Network inference algorithms

Learning transcriptional networks from a microarray compendium can either focus on the detection of modules (module inference) or on the detection of a regulatory program (regulatory program inference). Module inference algorithms search for sets of coexpressed genes that are assumed to be under influence of the same regulatory mechanism but without explicitly inferring this common regulatory program. Algorithms for regulatory program inference on the other hand try to assign regulators or sets of regulators to their corresponding target genes. They thus focus on the interactions in the network.

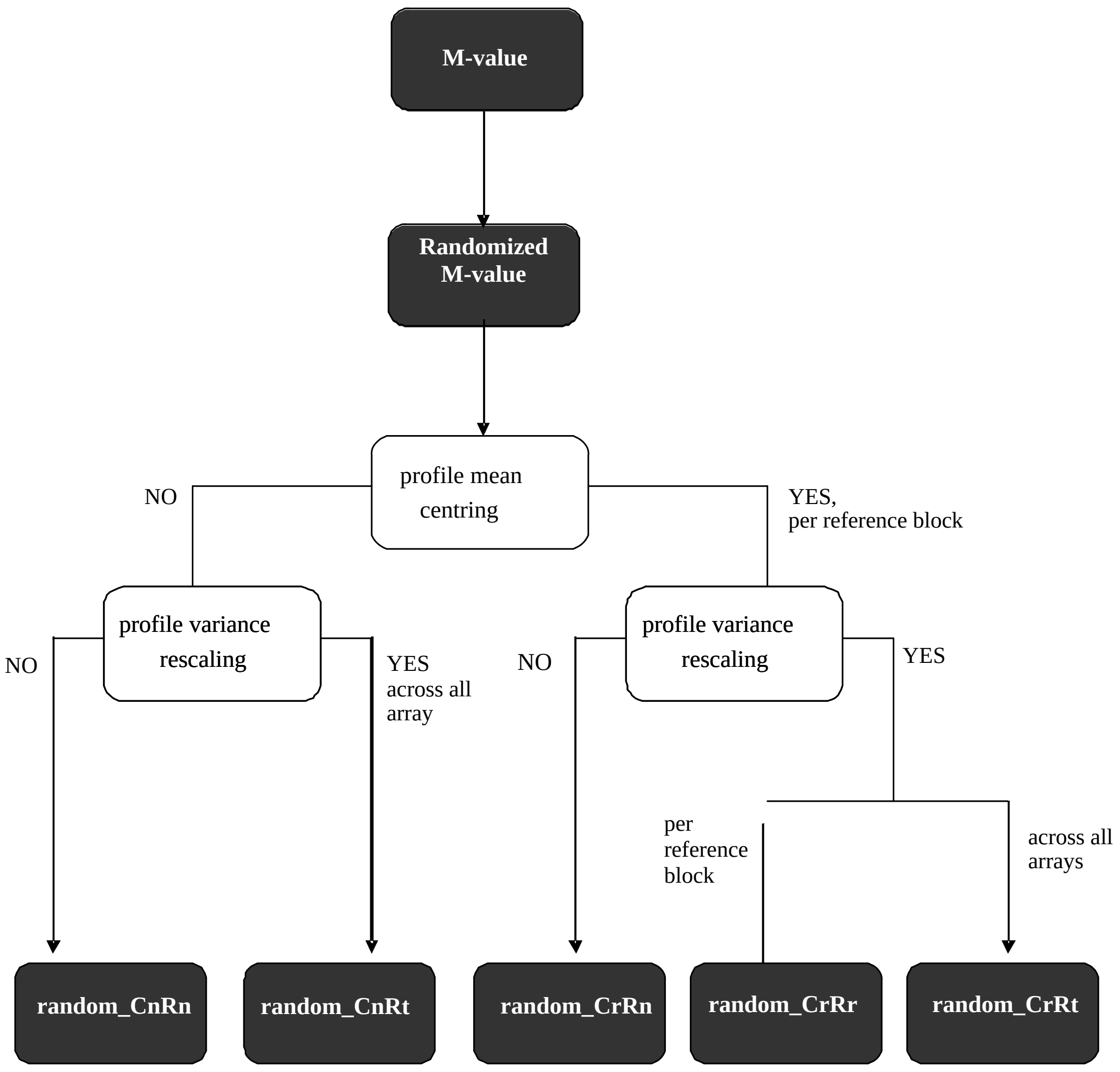
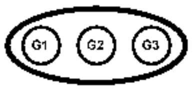
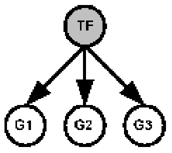
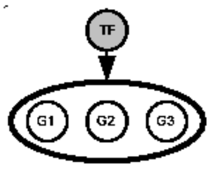


Figure 6.2: Five different randomized data sets.

An important property of the transcriptional network is its modularity: the network consists of overlapping modules of functionally related genes that all act in concert under certain conditions. Module inference algorithms are useful on their own as they tell us which genes are coexpressed. Clustering or biclustering algorithms can be used to learn about these modules. Even though clustering approaches were among the pioneers to mine microarray data, clustering algorithms general ignore the condition dependency of regulatory programs and thus the fact that target genes will only be coexpressed under the conditions in which the regulatory program is active. Whereas, biclustering algorithms deals with this shortcoming of clustering algorithms by combining a search for coexpressed genes with a condition selection step to identify the conditions under which the genes are coexpressed. Among biclustering algorithms, two different ways of biclustering can be used to solve different biological questions: global biclustering and query-driven biclustering. Global biclustering algorithms identify the more dominant patterns in the data set and result in large modules of coexpressed genes. Such modules correspond to large pathways, involving lots of genes, responsible for general responses. While query-driven biclustering search for smaller subtle patterns of coexpression, involving only a few genes or conditions which are coexpressed with a set of seed genes and might be of a particular interest to a certain scientist. Therefore, we consider one global biclustering algorithm and one query-driven biclustering algorithm as representatives of module inference algorithms in this chapter.

Algorithms that infer regulatory programs can be further subdivided into two categories: those that infer the program for each gene individually (‘direct’ network inference methods) and those that perform a module inference step prior or together with the assignment of the regulatory program. In the latter case one program is assigned to a complete module at once (‘module’ - based network inference). We choose one algorithm for each category as representatives in this Chapter. Table 6.1 shows different categories of network inference algorithms and the representative algorithms of each category used in this chapter.

Table 6.1: Four different categories of network inference algorithms.

Algorithm category			**Figure Illustration**	**Representative algorithm**
Network inference algorithms from a microarray compendium	Module inference algorithm (biclustering)	Global biclustering		ISA
		Query-driven biclustering		*Pro*Bic
	Regulatory program inference algorithm	Direct network inference algorithm		CLR
		Module based network inference algorithm		Genomica

Next, we will introduce the representative algorithm of each category into detail:

6.2.2.1 ISA (Iterative Signature Algorithm) [96]

ISA is one of the global biclustering algorithms. The algorithm aims to find sets of coregulated genes together with the relevant experimental conditions that induce their coregulation. Such a combined set is defined as a transcription module (TM) [96]. The algorithm searches for the modules encoded in the data by iteratively refining sets of genes and conditions until they match the definition of a transcription module. Note that the degree of similarity of genes or conditions in a TM is defined by a pair of threshold parameters. For each condition *c* in the TM the average expression level of the genes in the TM is above a certain threshold T_C (condition threshold). Conversely, for each gene g in the TM the average expression level over the conditions of the TM is also above some threshold T_G (gene threshold).

ISA takes a set of random genes that are called seed genes as initialization. Although the set of possible input seed genes is large, usually there exist only a rather limited number of fixed points for a given set of thresholds. A fixed point gene vector $g^{(*)}$ satisfies $\frac{|g^{(*)} - g^{(n)}|}{|g^{(*)} + g^{(n)}|} < \varepsilon$, for all n above a certain number of iterations. Together with the associated fixed point condition vector $c^{(*)}$ it defines a fixed point.

To sum up, ISA is applied as follows: 1) generate a sufficient large number of random seed genes. 2) find the fixed points corresponding to each seed through iterations. 3) collect the distinct fixed points in order to decompose the expression data into modules. The structure of this decomposition depends on the choice of gene threshold and condition threshold. The output of ISA is a list of modules (biclusters).

6.2.2.2 ProBic (Chapter 5)

*Pro*Bic can be used as a query-driven biclustering algorithm. Overlapping biclusters (modules) are identified in gene expression data using Probabilistic Relational Models, which is able to incorporate biological prior information in the form of a set of seed genes. Given a set of seed genes, an initial bicluster distribution for each condition is calculated and the background distribution for each condition is estimated. For a certain gene/condition, the likelihood scores of this gene/condition belonging to a bicluster or several biclusters versus its likelihood to belong the background are calculated. Depending on these likelihood scores the gene/condition is assigned to either background or a bicluster. Gene assignment and condition assignment change iteratively until convergence. *Pro*Bic recruits genes which have a similar expression profile as the seed genes under the selected conditions. Details of the algorithm are described in Chapter 6.

6.2.2.3 CLR (context likelihood of relatedness) [60]

CLR is one of the algorithms which detect the regulatory program directly for each individual gene. The algorithm takes a gene expression data set and a list of regulators as input and calculates the mutual information of each gene-regulator pair. CLR uses mutual information as a metric of similarity between the expression profiles of each gene and regulator pair (Faith et al., 2007). The mutual information for two discrete random variables X and Y can be defined as:

$$MI(X;Y) = \sum_{i,j} P(x_i, y_j) \log \frac{p(x_i, y_j)}{p(x_i)p(y_j)} \tag{6.1}$$

Then CLR algorithm calculates the statistical likelihood (z-score) of each mutual information value. The z-score of each regulatory interaction (z_{ij}) depends on the distribution of MI scores for all the possible regulators of the target gene (z_i) and on the distribution of MI scores for all possible target genes of the regulator (z_j) (Faith et al., 2007). Different dependencies between z_{ij} and z_i, z_j are available in their open source code. The following formula describing the dependence is used in the study:

$$Z_{ij} = sqrt(Z_i^2 + Z_j^2) \tag{6.2}$$

Using this concept of z-score guarantees that the most probable interactions correspond to those whose mutual information scores stand significantly above the background distribution of all mutual information scores.

The output of CLR is a list of individual interactions. In order to make the results of the four algorithms comparable, the results of CLR are also converted into modules by grouping the genes regulated by the same regulator from the list of predicted interactions. However, the length of this list depends on the choice of the threshold in the on the z-scores (Section 6.2.2.3). (A gene (i) and a regulator (j) are predicted to interact if the z_{ij} is above a certain threshold). Faith et al. suggests determining the threshold from the precision recall curve calculated using RegulonDB. In our study we used a similar approach: the threshold is chosen at a point with about 60% precision as suggested in the original article. Figure 6.3 shows the precision recall curves of the results from the five differently normalized data sets and indicates the points where the thresholds are chosen.

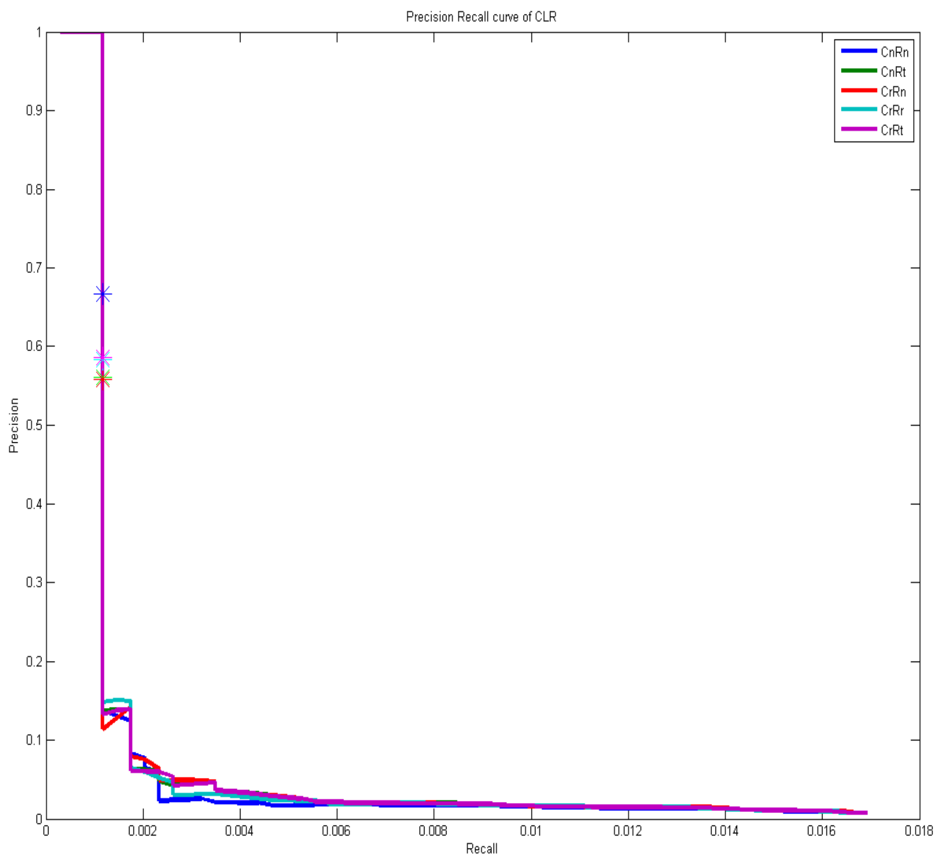


Figure 6.3: Precision recall curve for the CLR algorithm results using the five differently normalized data sets. On each curve, the asteroid indicates the precision and recall values given the threshold where 200 predicted interactions found.

6.2.2.4 Genomica [109]

Genomica is a module-based network inference algorithm. It identifies modules of coexpressed genes, their regulators and the conditions under which their regulation occurs [109]. The algorithm takes as input a gene expression data set and a large set of candidate regulators. Given these inputs, the algorithm searches simultaneously for a partition of genes into modules and for a regulatory program for each module that explains the expression behaviour of the genes in the module. To initialize the search, Genomica uses the results of a clustering algorithm on the whole data set. The clustering algorithm can be chosen from several correlation based agglomerative clustering algorithms (Section 2.2.3). Genomica takes a score-based approach to learn module networks. The score function (detail see Segal et al., 2003 technical report) is similar to the Bayesian score. Learning the module network is then applied as two search steps: a search for the modules and a search for the regulatory program for each module. Each regulatory program is as a tree structure. In each step, a score is calculated using the score function. The algorithm tries to find the high scoring module and regulatory program in the two steps respectively. Genomica outputs the list of modules and their corresponding regulatory program.

6.2.3 Analysis flow

The following steps are applied on each algorithm:

1. Prepare each data set according to the algorithm's required input format, for example a list of regulators for CLR and Genomica and the list of seed genes for *Pro*Bic. Missing values are handled according the algorithms instructions (Table 6.2).

2. Run each algorithm with all ten data sets as input. However, ISA applies an own normalization procedure [96], which results in that each condition and each gene following a normal distribution with mean zero and variance one. As this normalization would overrule the effect of our ten different normalization strategies, we skipped their own default normalization step and used all ten data sets directly as input. Except for the CnRn data set where we only applied their default normalization approach (M-values, see Section 6.2.1.3). We refer to this normalized data set as the default data set in this chapter. Result from the default data set will be compared with the results from the other ten data sets.

3. When running each algorithm, the default (recommended) parameter settings are used (Table 6.2).

4. Output results. ISA, *Pro*Bic and Genomica output a list of modules (biclusters) as results. The original output of CLR that is a list of individual interactions is converted into a list of modules using the way described in Section 6.2.2.3. Thus, the results of four algorithms are comparable.

5. For each obtained module (cluster) functional and regulator enrichment are calculated (Section 4.2.3.3) using the hyper-geometric distribution (details see Section 4.2.3.3). In this study, the enrichment of a function (regulator) is called significant when its p-value is smaller than 0.01.

Details for each analysis step for each algorithm are explained as Table 6.2.

Table 6.2: Detailed analysis flow for each algorithm

	Data Preparation			Algorithm Run
	Data requirement	Input format	Missing value handling	Parameters used
ISA	Gene expression data set	Matlab structure	Set all NaN values to zero	The recommended values in the original article: gene threshold[1]: 1 to 4 with step 0.1 Condition threshold: 2
*Pro*Bic	Gene expression data set A list of seed genes	.txt file .txt file	Remove all the conditions with all zero values.	Array to bicluster assignment prior: $\pi_{bicl} / \pi_{bgr} = -0.5$ Background and bicluster variance contrast: $\sigma_{bicl} = 1 * \sigma_{bgr}$ General gene to bicluster assignment prior: $\beta = -0.1$

CLR	Gene expression data set
	A list of regulators
		Matlab structure
		Matlab structure
			Filter out all the genes and conditions with more than 50% of NaN values.
			Replace the rest of NaN values to random values in the data set range.
				The parameters described in the article:
				Number of bins[3] = 7;
				spline degree[3] = 3;
				Method[4] = plos.

Genomica	Gene expression data set A list of regulators	.txt file .txt file	Filter out all the genes and conditions with more than 50% of NaN values. Replace the rest of NaN values to random values in the data set range.	The default parameters: Max nr of modules[6] = 50; Max tree depth[7] = 10; Min experiments per context[8] = 5; Regulator split constraints[9]: below -0.5 or above 0.5

(Continued)

	Result Output	
	Threshold	Output format
ISA	The one which gives the most modules.	A list of modules, each of which consists of a list of genes and a list of conditions.
ProBic	Not applicable	For each set of seed genes, one bicluster (module) is found: a list of genes and a list of conditions.
CLR	The one which results in 60% precision5.	A list of modules (clusters), each of which is a list of genes and the regulator regulates this list of genes.
Genomica	Choose the modules correspond to the split of their top regulators10	A list of modules (clusters), each of which is a list of genes and the top regulator assigned for this module.

1. There is no specific recommended gene threshold in the article. The suggested values are within the range of 2 to 4, so we take the whole range from 1 to 4 with step 0.1, and then choose the one giving the most modules as the final gene threshold. The modules generated given this threshold are considered as the final results.

2. For The reason of why these parameter settings are used we refer to Chapter 5 (Section 5.4.1).

3. CLR uses mutual information (Equation 6.1) as the distance measure to calculate the similarity of two expression profiles. Since the two expression profiles should be considered as discrete values, B-spline smoothing and discretization method is used to compute mutual information. In the original article, it is given that all the mutual information values are computed using 10 bins and third order B-spline.

4. Parameter 'method' indicated which method to use to calculate the z-score (Section 6.2.2.3). We use the same approach as described in the article (Section 6.2.2.3).

5. The matrix with z-score of each gene and regulator pair.

6. The maximum number of clusters got from the initial clustering algorithm (Section 6.2.2.4).

7. A regulatory program is as a tree structure. The tree depth is then the depth of a regulatory program.

8. The minimal number of experiments in one module.

9. The regulator within the regulator program (which is a tree structure) split at level -0.5 if it down-regulates its target genes or 0.5 if it up-regulates its target genes.

10. The top regulator is the root of the regulator program.

6.2.4 Criteria used to evaluate the results

1. The proportion of the modules that have at least one function enriched (proportionFunc) and the proportion of the modules that are enriched with at least one regulator

(proportionReg). These two values are used to evaluate how many of the resulting modules are biologically meaningful.

2. Proportion of the modules of which the calculated enriched regulators match with the regulators assigned by the algorithm (proportionRegMatch). This criterion is only applicable to the methods that explicitly assign a regulatory program i.e. *Pro*Bic, CLR and Genomica (but not for ISA). The extent to which for each of the tested algorithms the results obtained from differently normalized data sets overlap (overlap). For *Pro*Bic, CLR and Genomica, two modules considered to sufficiently ‘match’ when their regulatory programs were the same. For ISA, overlap between two lists of modules (e.g. *A* and *B*) is calculated as follows: start from the first module in list *A*, namely *A1*, and look for the module in list B that shows the largest overlap in the number of genes with *A1*. Remove the selected modules from both module lists and continue the search with the remaining modules in both lists until one of the two lists is empty.

3. Comparison of all enriched functions (matFunc) and all enriched regulators (matReg) for each of the differently normalized data sets. Given that *Pro*Bic works in a query-driven mode, it does not give any global view of a data set. These two comparisons are not applied on *Pro*Bic,

6.3 Results and Discussions

In this section, we will present the results of our analysis. At first we evaluated whether the results obtained with the respective algorithms on each of the data sets are biologically meaningful using the criteria as described in Section 6.2.4. Table 6.3 shows for each data set and algorithm combination 1) the proportion of the modules that have at least one enriched function, 2) the proportion of the modules that are enriched with at least one regulator, 3) the proportion of the modules whose enriched regulators match with their assigned regulators. Irrespective of the used normalization procedure the values of all calculated evaluation criteria for the real data sets are higher than those obtained from randomized data sets, indicating that the \results are biologically meaningful. However note that the absolute values are quite low indicating that few known biology could be recapitulated.

Table 6.3: Analysis results of all four algorithms. For each algorithm and each used data sets values of three criteria are listed in the table: 1) the proportion of the modules that have at least one function enriched (proportionFunc), the proportion of the modules that being enriched

with at least one regulator (proportionReg) and 3) the proportion of modules of which the enriched regulators match with the regulators assigned by the algorithm (proportionRegMatch).

	ISA		***Pro*Bic**			**CLR**			**Genomica**		
	Prop. Func	**Prop. Reg**	**Prop. Func**	**Prop. Reg**	**Prop. RegMatch**	**Prop. Func**	**Prop. Reg**	**Prop. RegMatch**	**Prop. Func**	**Prop. Reg**	**Prop. RegMatch**
CnRn	0.4571	0.4571	1	1	0.8276	0.4857	0.2857	0.1052	0.7	0.4	0.02
CnRt	0.5625	0.3625	1	1	0.7931	0.4712	0.2788	0.1062	0.7143	0.3673	0.0816
CrRn	0.4182	0.3273	1	0.9655	0.7931	0.4571	0.2857	0.1052	0.7347	0.4083	0.0408
CrRr	0.4750	0.3500	1	0.9655	0.8276	0.4571	0.2857	0.1052	0.7755	0.4490	0.0612
CrRt	0.5625	0.3625	1	1	0.8276	0.4808	0.2885	0.1062	0.6600	0.36	0.06
Random_CnRn	0.0578	0	0.0793	0	0	0.0828	0	0	0.1	0	0
Random_CnRt	0.0535	0	0.1483	0	0	0.0765	0	0	0.04	0	0
Random_CrRn	0.0600	0	0.0793	0	0	0.0702	0	0	0.06	0	0
Random_CrRr	0.0554	0	0.1483	0.0345	0.0345	0.0769	0	0	0.04	0	0
Random_CrRt	0.0584	0	0.1483	0	0	0.0581	0	0	0.08	0	0

To have an idea on how the results compare between different algorithms (testing whether they confirm each other in the results). (Given that *Pro*Bic works in a different mode (query driven) than ISA, CLR and Genomica (global) we did not include it in the comparison.) Figure 6.4 shows that the overlap of the functions which are enriched in the modules obtained by of the algorithms is relatively small. This indicates that results obtained from different algorithms are rather complementary rather than similar. One should however realize that comparing results between these algorithms is not trivial as they all might work on at a different precision/recall trade-off. For CLR we can define the trade-off of the used working regime: as we can calculate the precision and recall for a whole range of thresholds on the z scores. However, for ISA and Genomica, the algorithms result in a fixed list of modules. We can not change threshold setting that alters the recall precision trade-off. As a result we do have no idea on the used precision/recall trade-off. The fact that all three algorithms work at a different trade-off level might also explain the low overlap in results. In the remainder of the analysis we therefore only compare between results obtained from the same algorithm but with different normalized data sets.

From Table 6.3, we can conclude that, irrespective of the used normalization procedure all results are biologically sound. To have an idea about the similarity in results obtained with each of the different data sets, we calculated their overlap as described in Section 6.2.4. Results are indicated in Figure 6.5. For *Pro*Bic, CLR and Genomica, the results did not differ much between the different normalized data sets. For ISA the dependency of the results on the used dada set was slightly more pronounced. In the subsequent section we will discuss the algorithmic reasons that can explain the observed differences.

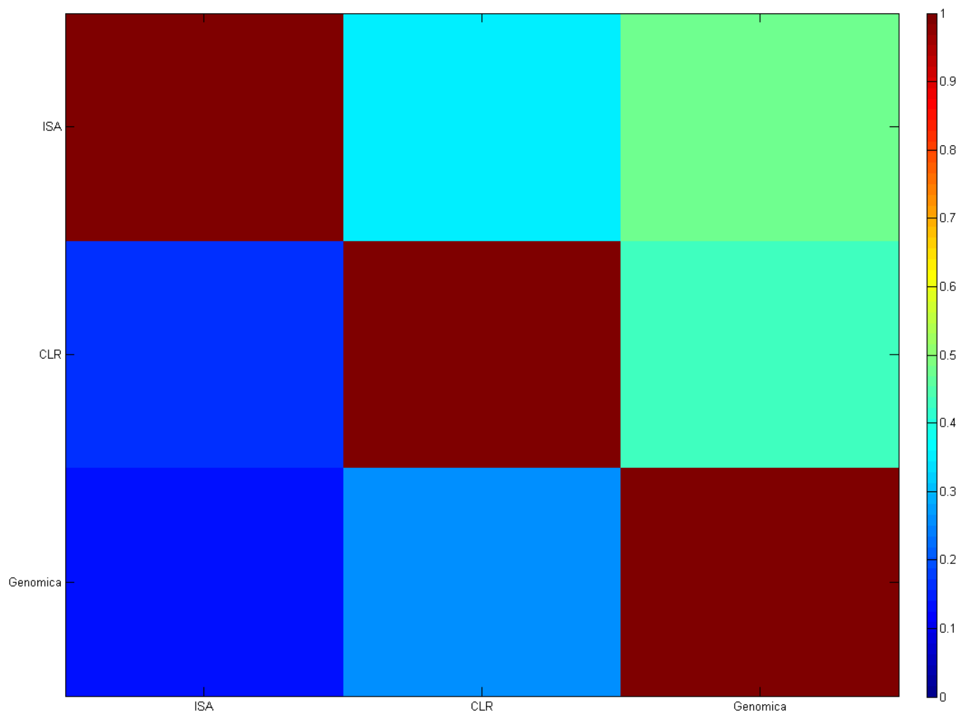


Figure 6.4: The overlap of the functions which are enriched in the modules obtained by ISA, CLR and Genomica. For every normalized data set, the pair wise overlap in results between two algorithms A and B is calculated as the number of the functions found enriched in the modules of that were in common between respectively algorithm A and B by both of the two algorithms divided by the number of enriched functions found by algorithm A. Overlap between B and A is the same number of the common functions divided by the number of enriched functions by algorithm B. The degree of overlap indicated by the color gradient is the average pair wise overlap for all five normalized data sets.

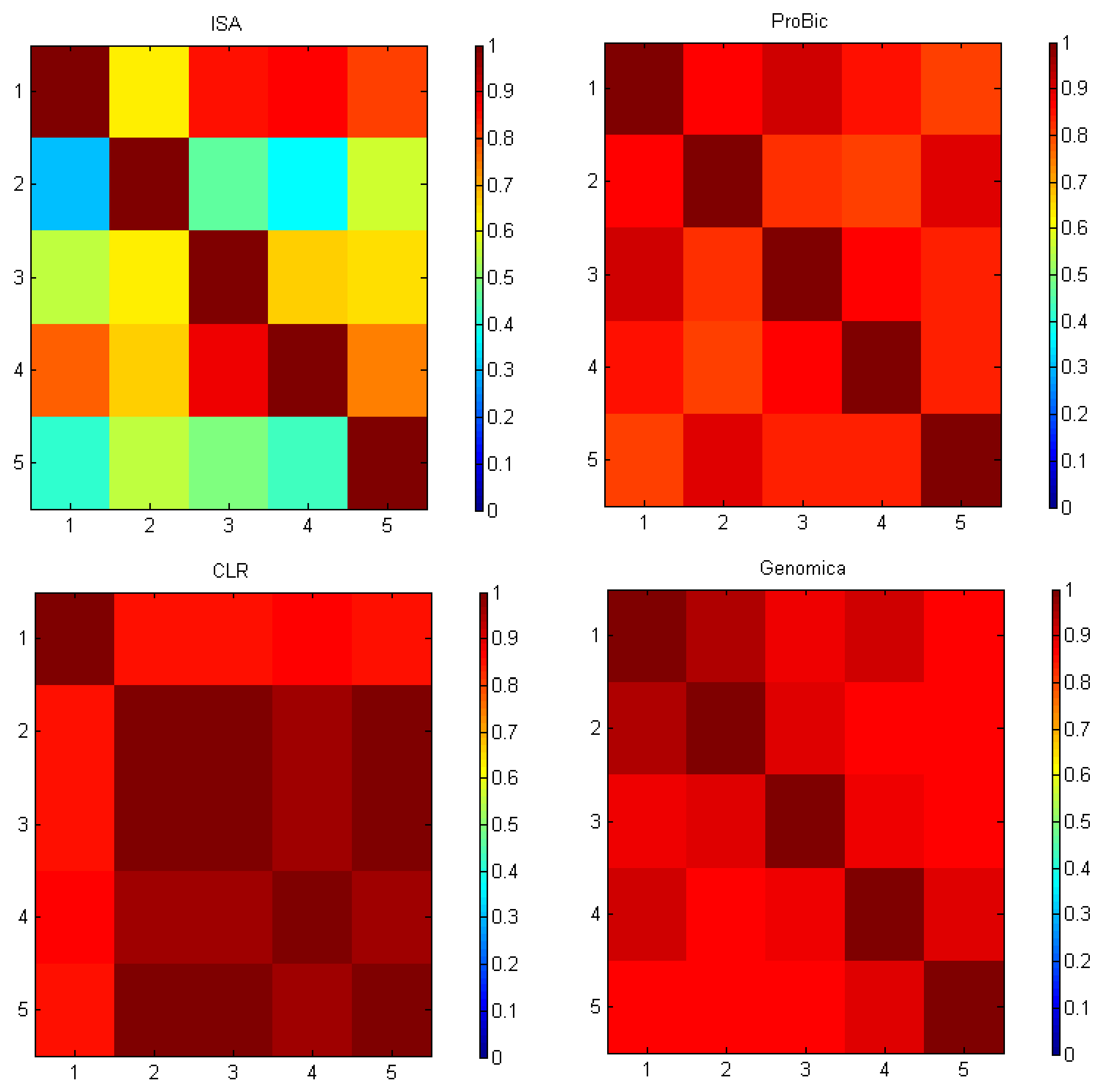
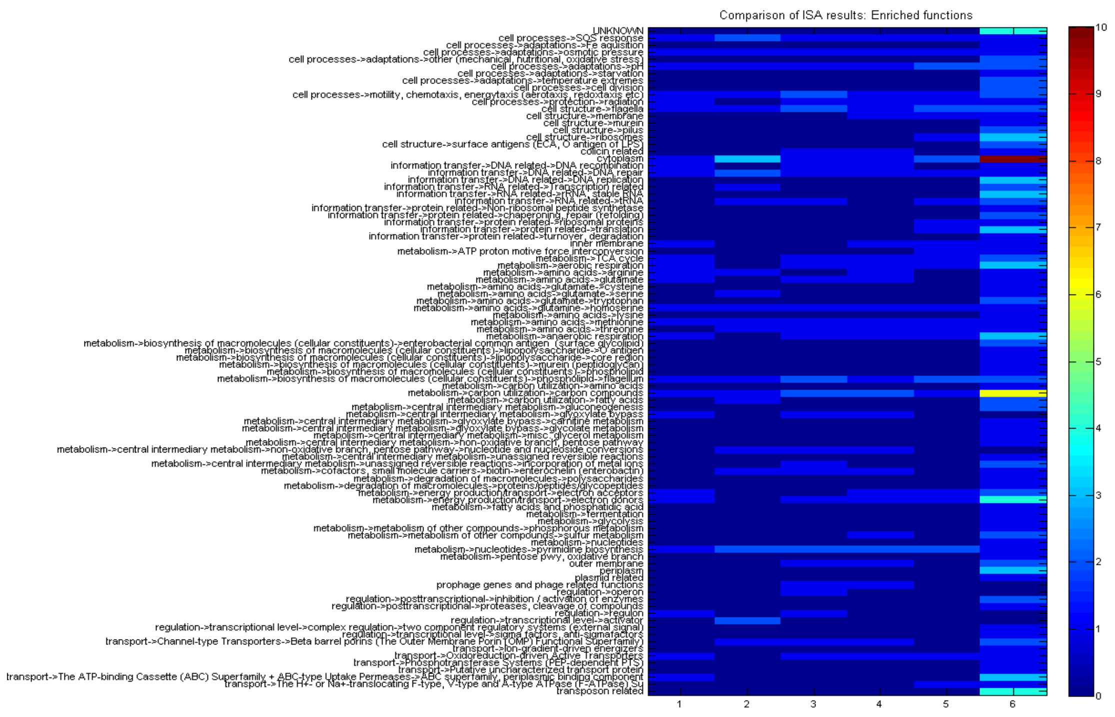


Figure 6.5: Overlap between the results obtained from five differently normalized data sets for each algorithm. The numbers in both axes represent different normalized data sets: 1- CnRn; 2 - CnRt; 3 - CrRn; 4- CrRr; 5 - CrRt. The color gradient indicates the degree of overlap between two results (with a higher value showing a bigger overlap).

ISA

ISA is a global biclustering algorithm that outputs a list of modules (biclusters) (Section 6.2.2.1). In order to calculate the overlap between two module lists, we have to find the best match between the two lists as described in Section 6.2.4. However, the match between the two lists can be quite complex and the relation between two matching modules is not necessarily a one to one relation. For example, one module in list A can be split into two modules in list B. This complexity explains the asymmetry in the similarity matrix shown in Figure 6.5 and makes the judgment on the degree of overlap between different results less straightforward as for the other algorithms. Therefore, we use the additional criterion that is the comparison of the enriched functions (matFunc) and the enriched regulators (matReg) for each of the differently normalized data sets (Section 6.2.4) to have a more detailed view on the results. The comparison is shown in Figure 6.6. The first five columns in Figure 6.6(left) and Figure 6.6(right) point out, that although it is difficult to find a match between the modules of different data sets, the functions and the regulators enriched in all the modules are highly similar. From this, we can conclude that also for ISA there is no significant difference between the results obtained with each of the five differently normalized data sets.

If we zoom in on the results, it becomes clear that the performance of ISA on these five data sets is rather bad. Small values of the proportion of the modules that have at least one enriched function and of the modules that are enriched with at least one regulator shown in Table 6.3 confirm this low performance. To test whether ISA performs better when applying its own default normalization approach, we evaluate its performance on the default data set (see Section 6.2.3). The values of the proportion of modules which have at least one enriched function and of modules which are regulated by at least one enriched regulator are 0.6316 and 0.3947 respectively, which is not significantly better than the values obtained with the other five data sets but as can be seen in Figure 6.6 that the functions and regulators enriched in the modules obtained from the default data set are quite different from the ones obtained with the five own normalized data sets. Conclusively using its default normalization procedure gives rise to different but equally bad results than when using our own five normalized data sets.



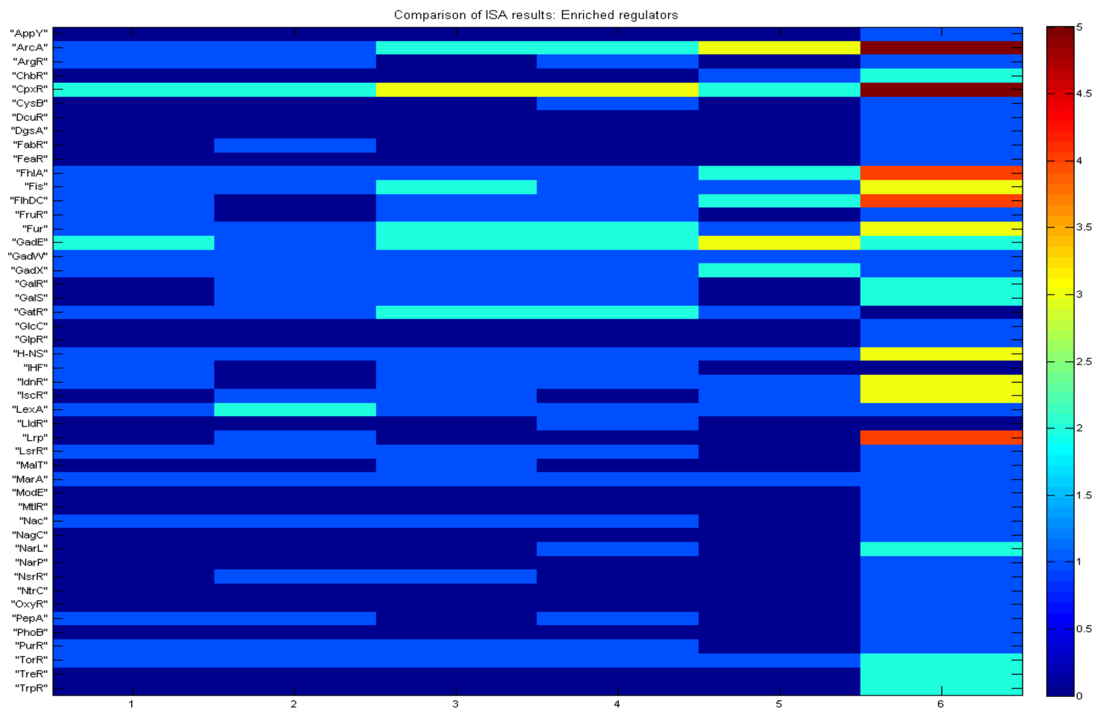


Figure 6.6: Comparison of the ISA results for the differently normalized data sets: the upper panel displays enriched functions and the lower panel displays enriched regulators. The numbers in x-axis of both panels represent differently normalized data sets: 1- CnRn; 2 - CnRt; 3 - CrRn; 4 - CrRr; 5 - CrRt; 6 - the default data set. The y-axis represents the functions (upper) and the regulators (lower) which were found to be enriched in any of the modules found detected by all of the data sets. The color gradient indicates how many modules found from a

certain data set are enriched for a particular enriched function (or regulated by a particular regulator). The color gradient indicates how many modules found from a certain data set are enriched for a particular enriched function (or regulated by a particular regulator).

The explanation of the relatively low performance and difference in results obtained from the different data sets can be explained by the specificities of the ISA algorithm. ISA starts from a large number of input seed genes and then finds the fixed points corresponding to each seed through iterations. The final modules are collected from these fixed points. As described in Bergman et al, 2003, the iterative scheme reads as follows:

$$\hat{c}^{(n)} = \frac{E \cdot \hat{g}^{(n-1)}}{|E \cdot \hat{g}^{(n-1)}|} \tag{6.3}$$

$$\hat{g}^{(n)} = \frac{E^T \cdot \hat{c}^{(n)}}{|E^T \cdot \hat{c}^{(n)}|} \tag{6.4}$$

The fixed points of the above equations correspond to the pairs of vectors $(\hat{g}_m, \hat{c}_m)$, where $\hat{g}_m = g_m / |g_m|$ and $\hat{c}_m = c_m / |c_m|$ are the normalized eigenvectors of respectively $E \cdot E^T$ and $E^T \cdot E$. Both eigenvectors are associated with the common eigenvalue $\mu_m^2 = \left|E \cdot \hat{g}_m\right|^2 = \left|E^T \cdot \hat{c}_m\right|^2$ (Bergman et al, 2003). The way the fixed point is defined in their article implies that the gene and the condition vector should follow the same distribution. This is the reason why ISA requires each gene and each condition in the expression matrix to follow normal distribution with mean zero and variance one. If this is not the case, the performance of the algorithm will be low because the data do not meet the prerequisites of the iteration. This explains why using the five own normalized data sets result in a low performance. However, this prerequisite itself is questionable. Forcing all the genes and conditions into one unique distribution will result in information loss which might explain why ISA also performs badly using the default data set.

*Pro*Bic

*Pro*Bic is used as a query-driven biclustering algorithm. Given a set of seed genes, *Pro*Bic recruits genes that have similar expression profile with the seed genes under certain conditions as the resulting modules (biclusters). For a certain gene/condition, the likelihood scores of this gene/condition belonging to a bicluster or several biclusters versus its likelihood to belong the background are calculated. Depending on these likelihood scores the gene/condition is assigned to either background or a bicluster. This basic idea implies that the decision on whether a gene belongs to a bicluster is made based on a relative rather than an absolute comparison of the measurements. Since different normalization approaches will only change the absolute expression level of a gene but not their relative behaviour, the genes with significantly similar expression profiles as the seed gene sets will always be recruited into the module (bicluster). This explains the low impact of the used normalization procedures on the observed results.

CLR

Table 6.3 and Figure 6.3 show that CLR provides very similar results irrespective of the type of normalization but with the chosen threshold, the algorithm performs quite badly. Less than half of the modules have at least one function enriched and less than one third of the modules are enriched with minimally one regulator. Only about 10% of the modules' enriched regulators match with the regulators assigned by the algorithm. CLR assigns a regulatory program to a gene based on the similarity between the genes expression profiles and that of its matching regulator pair. As CLR s a direct method it does not select conditions, the assignment of a regulator is thus based on the similarity in the gene regulator expression profiles being assessed over all conditions. As the used compendium is quite heterogeneous in the conditions it tests (840 conditions in this study), it can be expected that a gene is not being active under all conditions. The similarity in expression will thus also be restricted to a subset of the conditions. As a result calculating an overall similarity measure that considers all conditions might lower the quality of the results. The high similarity in results obtained with the different data sets, is explained by the basic principle of CLR. The algorithm uses mutual information as a metric of similarity between the expression profiles of each gene and regulator pair [60]. The mutual information for two discrete random variables X and Y can be defined as Equation 6.1. In this equation the denominator is the normalization factor $P(x_i)P(y_j)$. The explicit involvement of a normalization factor overrules the normalization step.

Genomica

Genomica identifies modules of coregulated genes, their regulators and the conditions under which regulation occurs [109]. Table 6.3 shows that about 70% of the modules have at least one enriched function and about 30% of the modules are enriched with at least one regulator. Also here, the match between the regulators enriched in the modules and the regulator assigned to the module is quite low. Even though Genomica uses a score-based approach to learn modules and regulatory programs, it takes the results of a correlation-based clustering as initialization. Correlation coefficient between two random variables X and Y is defined as:

$$\rho_{X,Y} = \frac{\mathrm{cov}(X,Y)}{\sigma_x \sigma_Y} = \frac{E[(X-\mu_x)(Y-\mu_Y)]}{\sigma_X \sigma_Y} \tag{6.5}$$

where μ_x and μ_Y are the means and σ_x and σ_Y are standard deviations. Like the equation for mutual information, there are also normalization factors σ_x and σ_Y in the definition of correlation coefficient. When initializing Genomica with the correlation-based clustering algorithm, an inherent normalization will be applied. Irrespective of the used input, the algorithm will initialize all data sets in the similar way. Genomica learns the modules and the regulatory programs iteratively by a score-based approach. The basic principle of this approach is similar to the iterative approach used by *Pro*Bic (Chapter 5). Therefore, the same reasoning holds to explain why no significant difference is found with differently normalized data sets.

6.4 Conclusions

Different normalization methods can be applied to generate microarray compendia. We assumed the final results of network inference algorithms applied on an expression compendium would be affected by the type of normalization that was used to generate the compendium. In this chapter, we studied the effect of 5 different normalization procedures on the results obtained with the following four inference algorithms: ISA, *Pro*Bic, CLR and Genomica. These algorithms were chosen as representatives of four different categories of network inference algorithms. For neither of the algorithms we found significant difference between the results obtained with compendia that were normalized differently. The explanation for this observation is that each algorithm has its own internal normalization procedure which overrules the effect of normalization procedures applied on the compendia. Conclusively, whenever the signal in the data set is sufficiently strong it will be detected irrespective of the way the compendia were normalized.

Chapter 7

Conclusions and Perspectives

The research described in this dissertation covers a number of topics related to preprocessing and integration of high-throughput data. In this chapter, the results and achievements that presented from this work are summarized followed by a description of some directions for future work.

7.1 Summary and achievements

Chapter 3 presents a BioConductor package CALIB designed for estimating absolute expression levels from two-color microarray data. The normalization method provided in the package is based on a physically motivated model, explicitly modelling the hybridization of transcript targets to their corresponding DNA probes and the relation between the measured fluorescence and the amount of hybridized, labelled target. The method allows for normalizing the data without making any assumptions on the distribution of gene expression, as opposed to procedures relying on the Global Normalization Assumption. The method links target concentration to measured intensity, estimating absolute expression levels of transcript targets in the hybridization solution rather than the log-ratios between two samples becomes possible. CALIB is a user-friendly BioConductor package designed to increase this method’s usability and accessibility.

Chapter 4 illustrates a systematic microarray analysis flow, from assessing the quality of the microarrays to selecting appropriate normalization method, from identifying differentially expressed genes to clustering analysis of the differentially expression lists. A range of biological results was obtained as the application of this analysis flow.

Chapter 5 describes the development of a model-based biclustering algorithm *Pro*Bic that is developed within the framework of probabilistic relational models. Both query-driven and global biclustering are combined in a single model and the model simultaneously identifies multiple potentially overlapping biclusters containing genes with correlated expression profiles. The biclustering model naturally deals with missing values due to its probabilistic nature. This leads to the robust identification of biclusters under various settings of noise and missing values. Results on simulated data from Prelic et al show that *Pro*Bic competes with state-of-the-art biclustering algorithms and even outperforms ISA and QDB for settings with high overlap between biclusters. In query-driven setting, the algorithm can remove seed genes that have no matching expression profile with the bicluster. This is an important feature in practice to biologists. It is also verified that *Pro*Bic can successfully reproduce known regulatory interactions in RegulonDB.

Chapter 6 discusses the influence of using compendia that were normalization differently on the following four chosen inference algorithms: ISA, *Pro*Bic, CLR and Genomica, which were chosen as the representatives of four different categories of network inference algorithms. For the chosen algorithms, no significant difference is found between the results obtained with data sets that were normalized differently. Each algorithm has its own internal normalization procedure which overrules the effect of normalizations that were or were not applied on the compendia. This means that whenever the signal in the dataset is sufficiently strong it will be detected and for most algorithms the way the compendia are normalized is of little influence.

7.2 Future work

CALIB

Currently, two steps are performed in CALIB package to estimate the absolute expression levels. After a physically motivated calibration model is built, the first step is parameter estimation followed by normalization as the second step. The parameters of this model are estimated from external spike measurements array by array in the set of experiment. Once the parameters of the calibration model have been estimated for each array, they can be used to normalize the data. Absolute expression levels for each combination of a gene and condition in the experiment design are estimated, regardless of the number of replicates. However, some future research can be done to find computational inexpensive ways that adequately quantify the error distribution on the estimated target levels. With proper test statistics, confidence intervals based on these distributions could be used to identify genes with significantly changing expression, or genes that were measured with a high inconsistency. Also, having a measure of reliability on the estimated expression levels could prove a welcome addition for the subsequential analysis of microarray data (Section 2.2.3).

Microarray analysis flow

Currently, we included only a subset of methods in the analysis flow as it was adapted to the analysis of small scale microarray experiments (20 microarrays at the most), Extending this flow with additional analysis methods will enlarge the number of possible applications: towards the future the inclusion of preprocessing methods that analyze *de novo* sequencing data will be of crucial importance.

***Pro*Bic**

In order to convert *Pro*Bic into a user-friendly software package that is of direct use to a biologist, a number of tasks need to be completed. First, the current version of *Pro*Bic is command line based. While this has been proven very useful for running high-throughput experiments, a graphical user interface needs to be developed in order to be of practical use to end users. Second, the query-driven mode of *Pro*Bic is quite sensitive to different choices of priors and their parameters. While a bioinformatician who is 'skilled-in-the-art' can explore this parameters space and define sensible parameter values for a particular data set, research remainto be done to fully automate this prior and parameter selection. Finally, the biological results shown in Chapter 6 are for *Escherichia coli*. More extensive analysis of the biclustering algorithm on the different biological data sets would be useful.

7.3 Perspectives

Linking genomic variation to phenotypic information of individual genes will become one of the largest future bioinformatics challenges. Many genomic aberrations result in alterations in the signal transduction network that ultimately affect gene expression. Molecular profiles (expression or protein profiling) of individuals or patients can thus also be considered as phenotypes or traits [105]. Similarly to linking them to physical phenotypes, genomic markers can be linked to expression profiles (eQTL) as well.

State of the art methods try to link genomic variations to the expression profiles of individual genes. Because often datasets are limited in size (the number of assessed expression traits in a population is generally several orders of magnitude lower than regular phenotypes) classical association statistics often fail (limited power). Biclustering algorithms have their advantages handling this challenge because they can significantly enhance the statistical power and the confidence in the predictions of the association. Therefore, the biclustering algorithm *Pro*Bic we developed can serve as basis for this challenge.

Moreover, with the advent of recent next generation sequencing techniques and other high-throughput technologies (e.g. ChIP-chip data, sequence data), magnificent amounts of genome-wide data are available and supportive to reveal this challenge. Large amount of heterogeneous data sources require the integrative models. Integration of heterogeneous data sources aims at combining selected data sources so that they form a unified view to illustrate the interaction of the biological system. On one hand, this view revealed by data integration is unprecedented, large-scale and focuses on global picture of the biological system. On the other hand, data integration will make good use of the complementarily of different heterogeneous data sources. Different data source only provide partial information for the biological system. For example, if we want to study transcriptional networks, expression data, ChIP-chip data and motif data are all available. Expression data can provide information for co-expression. Co-expression genes are potentially co-regulated but can also have different regulatory program. ChIP-chip data is used to investigate binding between a regulatory and its target genes. However, binding does not necessarily imply that the regulatory regulates these genes since binding could be not functional. Sequence data can identify genes with the same motif. These genes all have the binding site of a certain regulatory, but the regulatory could bind to some of these genes but not to others. That is to say, each individual data source can only provide limited information for the transcriptional networks reconstruction. Therefore, data integration would help overcome their shortcomings of and obtain a comprehensive view of the whole network. At the same time, given the high noise level and incompleteness in all these data sources, integrative models could significantly enhance noise and missing value robustness. *Pro*Bic can be extended and sever as this kind of integrative models to solve this challenge.

The *Pro*Bic model can be extended to integrate with other data sources and to explain the expression variation based on PRMs. The integration of different other data sources can be done both in the gene and the condition dimension. In the gene dimension, detection additional information on regulatory motifs, ChIP-Seq or ChIP-chip, pathway information can steer the module detection process. In the condition dimension, experiment information or phenotypic traits with the individuals (strains) of which the expression was profiles can steer the condition selection (as shown in Figure 7.1). Steering the module detection in the gene and the condition direction could be done by using the concept of prior information or by introducing new classes to the PRMs. This would finally result in a generic model that can be tuned to solve different problems in a flexible way.

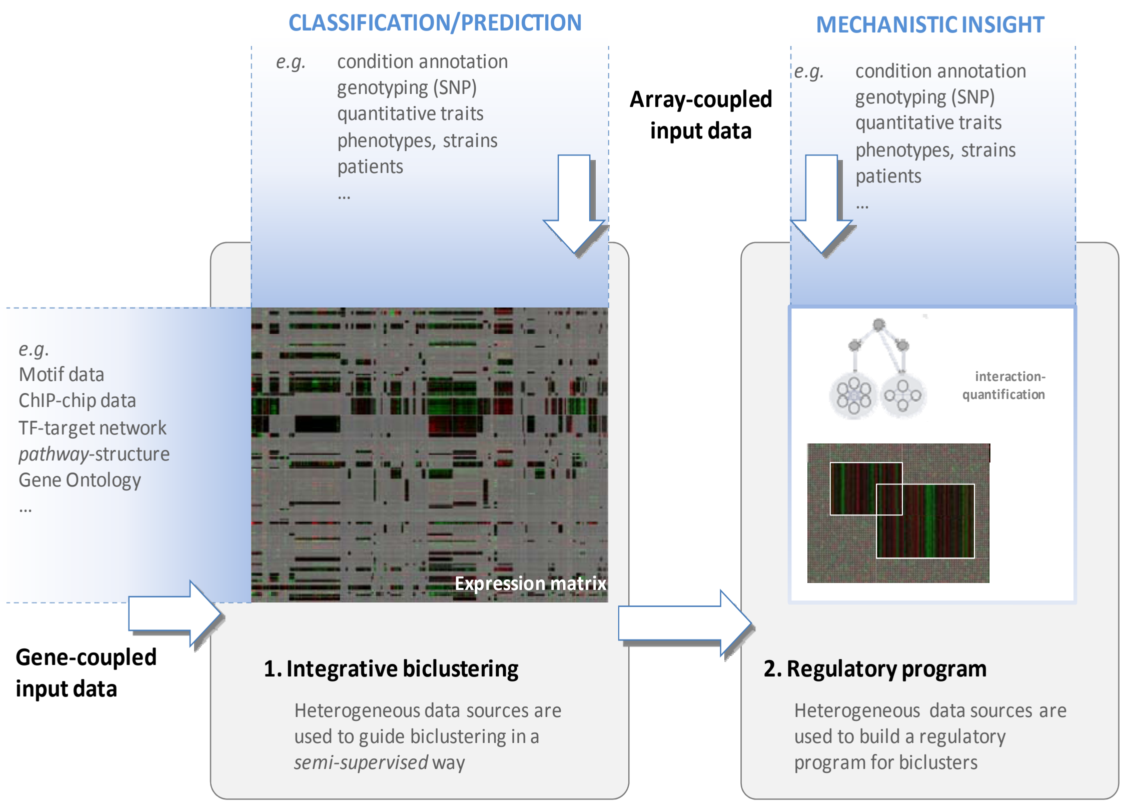


Figure 7.1: A flexible and generic data integration framework for the explanation of expression data. First (left part of the figure), ProBic will be extended towards heterogeneous data sources, coupled to genes and/or conditions in a gene expression dataset, to guide the identification of complex gene expression patterns (i.e., biclusters). Therefore the biclusters will not only be inferred on the base of expression data, but through the combination of the data sources. Next (right part of the figure), the expression variation in the inferred biclusters will be modeled, also on the base of external data sources to couple a regulatory program to each module.

Appendix

A) Enriched pathways for the clusters for the study of AI-2 mediated quorum sensing in *Salmonella* Typhimurium

A-1. Cluster 2

Enriched pathway	p-value
1.6.1.2 Glycerol Degradation	1.26e-03
1.6.1 Alcohols Degradation	9.75e-03
1.3.8.9.6.1 Adenosylcobalamin Biosynthesis	1.26e-03
1.3.8.9.6 Cobalamin Biosynthesis	1.26e-03
1.2.14 adenosylcobalamin biosynthesis II (late cobalt incorporation)	1.26e-03
1.3.8.9.6.1.2 adenosylcobalamin biosynthesis II (late cobalt incorporation)	1.26e-03
1.6.1.2.1 glycerol degradation III	9.75e-03

A-2. Cluster 4

Enriched pathway	p-value
1.6.11.3.1 Sulfate Reduction	3.18e-03
1.6.11.3.3 Sulfite Oxidation	3.18e-03
1.5.9.2.2 sulfate reduction IV (dissimilatory)	3.18e-03
1.6.11.3.1.1 sulfate reduction IV (dissimilatory)	3.18e-03
1.6.11.3.3.1 sulfite oxidation III	3.18e-02
1.6.11.3.12 sulfate activation for sulfonation	3.18e-03

A-3. Cluster 6

1.3.6 Carbohydrates Biosynthesis	1.36e-03
1.6 Degradation/Utilization/Assimilation	7.47e-03
1.3.8.1 Coenzyme A Biosynthesis	4.23e-02
1.6.6.3 CO2 Fixation	2.18e-04
1.3 Biosynthesis	7.70e-04
1.3.14.1.7 Glutamate Biosynthesis	4.23e-02
1.6.4.8 Glutamine Degradation	2.84e-02
1.3.8.9.4 Pantothenate Biosynthesis	4.23e-02
1.6.4.14 Leucine Degradation	2.84e-02
1.6.6.1 Formaldehyde Assimilation	7.47e-05
1.3.14 Amino acids Biosynthesis	8.20e-03
1.6.12 Nucleosides and Nucleotides Degradation and Recycling	4.00e-03
1.3.14.1.1 Alanine Biosynthesis	2.84e-02
1.6.6 C1 Compounds Utilization and Assimilation	7.51e-04
1.2 Superpathways	1.08e-02
1.3.14.1.22 Valine Biosynthesis	4.23e-02

1.5.4 Pentose Phosphate Pathways	7.47e-05
1.5 Generation of precursor metabolites and energy	4.29e-03
1.3.6.2 Sugars Biosynthesis	1.09e-03
1.3.14.1.12 Leucine Biosynthesis	4.23e-02
1.6.12.4 superpathway of ribose and deoxyribose phosphate degradation	4.00e-03
1.2.25 superpathway of ribose and deoxyribose phosphate degradation	4.00e-03
1.3.10.2.4 de novo biosynthesis of pyrimidine deoxyribonucleotides	2.84e-02
1.2.29 pentose phosphate pathway	9.50e-05
1.5.4.4 pentose phosphate pathway	9.50e-05
1.3.14.1.1.1 alanine biosynthesis I	2.84e-02
1.3.14.1.7.3 glutamate biosynthesis I	2.84e-02
1.6.6.1.2 formaldehyde assimilation II (RuMP Cycle)	3.24e-05
1.3.14.1.22.1 valine biosynthesis	4.23e-02
1.6.4.14.1 leucine degradation III	2.84e-02
1.3.8.9.4.2 pantothenate biosynthesis II	4.23e-02
1.6.4.23.2 valine degradation I	4.23e-02
1.6.4.23.1 valine degradation II	2.84e-02
1.6.4.13.1 isoleucine degradation I	4.23e-02
1.5.4.3 pentose phosphate pathway (partial)	4.29e-06
1.3.14.3 superpathway of leucine, valine, alanine, and isoleucine biosynthesis	4.13e-04
1.2.2 superpathway of leucine, valine, alanine, and isoleucine biosynthesis	4.13e-04
1.3.8.9.4.1 pantothenate biosynthesis I	4.23e-02
1.6.6.3.2 Calvin cycle	4.37e-05
1.3.6.2.4 Calvin cycle	4.37e-05
1.5.5.1 Calvin cycle	4.37e-05
1.6.4.13.2 isoleucine degradation II	2.84e-02
1.6.4.8.2 glutamine degradation II	2.84e-02
1.3.8.1.2 pantothenate and coenzymeA biosynthesis II	4.23e-02
1.2.10 pantothenate and coenzymeA biosynthesis II	4.23e-02
1.6.12.2 (deoxy)ribose phosphate degradation	4.00e-03
1.3.14.1.12.1 leucine biosynthesis	4.23e-02
1.5.4.1 pentose phosphate pathway (non-oxidative branch)	7.08e-06
1.3.8.9.8 flavin biosynthesis	4.23e-02
1.3.14.4 superpathway of leucine, valine, and isoleucine biosynthesis	3.03e-04
1.2.3 superpathway of leucine, valine, and isoleucine biosynthesis	3.03e-04

A-4. Cluster 8

Enriched pathway	p-value
1.6.15.1.9.2 Sugar Acids Degradation	1.35e-03
1.6.15.1 Plant Secondary Metabolites Degradation	3.31e-03
1.3.9 Fatty Acids and Lipids Biosynthesis	3.01e-02
1.6.15 Secondary Metabolites Degradation	3.31e-03
1.6.15.1.9 Sugar Derivatives Degradation	3.31e-03
1.6.15.1.9.2.4 D-glucarate degradation	2.30e-04
1.3.12.2.2 putrescine biosynthesis III	3.04e-02
1.3.9.8 Lipid A-core biosynthesis	6.83e-04
1.6.6.1.1 formaldehyde assimilation I (serine pathway)	4.53e-02
1.3.7.7 GDP-mannose metabolism	1.53e-02
1.6.14.2.14 D-mannose degradation	1.53e-02
1.2.33 superpathway of D-glucarate and D-galactarate degradation	2.30e-04
1.6.15.1.9.2.1 D-galactarate degradation	2.30e-04
1.6.7.6 glycolate and glyoxylate degradation I	3.04e-02

A-5. Cluster 26

Enriched pathway	p-value
1.6 Degradation/Utilization/Assimilation	3.66e-03
1.5.6 TCA cycle	3.18e-04
1.3.9.4.1.2 Cyclopropane Fatty Acids Biosynthesis	2.54e-02
1.6.4.7 Glutamate Degradation	3.03e-02
1.6.10 Fatty Acid and Lipids Degradation	7.21e-05
1.6.10.2 Fatty Acids Degradation	1.89e-06
1.3.9.4.1 Unusual Fatty Acid Biosynthesis	2.54e-02
1.6.4.3 Arginine Degradation	1.52e-04
1.6.4 Amino Acids Degradation	1.99e-04
1.5.6.1 TCA cycle variation I	1.19e-03
1.5.9.2.3 respiration (anaerobic)	3.03e-02
1.3.9.4.1.2.1 cyclopropane fatty acid (CFA) biosynthesis	2.54e-02
1.5.6.2 TCA cycle	2.25e-03
1.6.4.3.4 arginine degradation II	1.58e-05
1.6.10.2.1 fatty acid &beta;-oxidation I	1.89e-06

A-6. Cluster 27

Enriched pathway	p-value
1.3.8.8.2 menaquinone biosynthesis	4.43e-02

B) Functional category for the conditions in the *E.coli* compendium

1	carbon_source
2	aerobic__anaerobic
3	DNA
4	Fe
5	Biofilm
6	pH
7	diauxic_shift
8	amino_acids
9	heat_cold_stress
10	antibiotic__toxic_agent
11	Cu
12	nitrosating_agent
13	sigma_factor
14	growth
15	unknown

Bibliography

1. Ideker T, Galitski T, Hood L: **A new approach to decoding life: systems biology**. *Annu Rev Genomics Hum Genet* 2001, **2:**343-372.

2. Brown PO, Botstein D: **Exploring the new world of the genome with DNA microarrays**. *Nat Genet* 1999, **21:**33-37.

3. Barrett T, Suzek TO, Troup DB, Wilhite SE, Ngau WC, Ledoux P, Rudnev D, Lash AE, Fujibuchi W, Edgar R: **NCBI GEO: mining millions of expression profiles--database and tools**. *Nucleic Acids Res* 2005, **33:**D562-D566.

4. Demeter J, Beauheim C, Gollub J, Hernandez-Boussard T, Jin H, Maier D, Matese JC, Nitzberg M, Wymore F, Zachariah ZK, Brown PO, Sherlock G, Ball CA: **The Stanford Microarray Database: implementation of new analysis tools and open source release of software**. *Nucleic Acids Res* 2007, **35:**D766-D770.

5. Parkinson H, Kapushesky M, Shojatalab M, Abeygunawardena N, Coulson R, Farne A, Holloway E, Kolesnykov N, Lilja P, Lukk M, Mani R, Rayner T, Sharma A, William E, Sarkans U, Brazma A: **ArrayExpress--a public database of microarray experiments and gene expression profiles**. *Nucleic Acids Res* 2007, **35:**D747-D750.

6. D'haeseleer P, Wen X, Fuhrman S, Somogyi R: **Linear modeling of mRNA expression levels during CNS development and injury**. *Pac Symp Biocomput* 1999**:**41-52.

7. van Someren EP, Wessels LF, Backer E, Reinders MJ: **Genetic network modeling**. *Pharmacogenomics* 2002, **3:**507-525.

8. de JH: **Modeling and simulation of genetic regulatory systems: a literature review**. *J Comput Biol* 2002, **9:**67-103.

9. Hartwell LH, Hopfield JJ, Leibler S, Murray AW: **From molecular to modular cell biology**. *Nature* 1999, **402:**C47-C52.

10. Ihmels J, Bergmann S, Barkai N: **Defining transcription modules using large-scale gene expression data**. *Bioinformatics* 2004, **20:**1993-2003.

11. Rapley R, Harbron S: *Molecular Analysis and Genome Discovery*: John Wiley & Sons. Inc.; 2005.

12. Ferrantini A, Carlon E: **On the relationship between perfect matches and mismatches in Affymetrix Genechips**. *Gene* 2008, **422:**1-6.

13. Schuchhardt J, Beule D, Malik A, Wolski E, Eickhoff H, Lehrach H, Herzel H: **Normalization strategies for cDNA microarrays**. *Nucleic Acids Res* 2000, **28:**E47.

14. Churchill GA: **Fundamentals of experimental design for cDNA microarrays**. *Nat Genet* 2002, **32 Suppl:**490-495.

15. Kerr MK, Churchill GA: **Statistical design and the analysis of gene expression microarray data**. *Genet Res* 2001, **77:**123-128.

16. Leung YF, Cavalieri D: **Fundamentals of cDNA microarray data analysis**. *Trends Genet* 2003, **19:**649-659.

17. Quackenbush J: **Microarray data normalization and transformation**. *Nat Genet* 2002, **32 Suppl:**496-501.

18. Ritchie ME, Silver J, Oshlack A, Holmes M, Diyagama D, Holloway A, Smyth GK: **A comparison of background correction methods for two-colour microarrays**. *Bioinformatics* 2007, **23:**2700-2707.

19. Yang YH, Buckley MJ, Speed TP: **Analysis of cDNA microarray images**. *Brief Bioinform* 2001, **2:**341-349.

20. Affymetrix: **Affymetrix Statistical algorithms reference guide.**; 2001.

21. Binder H, Preibisch S: **Specific and nonspecific hybridization of oligonucleotide probes on microarrays**. *Biophys J* 2005, **89:**337-352.

22. Durbin DM, Rocke N: **A model for measurement error for gene expression analysis**. *J Comp Biol* 2001, **8:**557-569.

23. Tran PH, Peiffer DA, Shin Y, Meek LM, Brody JP, Cho KW: **Microarray optimizations: increasing spot accuracy and automated identification of true microarray signals**. *Nucleic Acids Res* 2002, **30:**e54.

24. Irizarry RA, Bolstad BM, Collin F, Cope LM, Hobbs B, Speed TP: **Summaries of Affymetrix GeneChip probe level data**. *Nucleic Acids Res* 2003, **31:**e15.

25. Li C, Wong WH: **Model-based analysis of oligonucleotide arrays: Expression index computation and outlier detection.**; 2001:31-36.

26. Irizarry RA, Hobbs B, Collin F, Beazer-Barclay YD, Antonellis KJ, Scherf U, Speed TP: **Exploration, normalization, and summaries of high density oligonucleotide array probe level data**. *Biostatistics* 2003, **4:**249-264.

27. Chen Y, Kamat V, Dougherty ER, Bittner ML, Meltzer PS, Trent JM: **Ratio statistics of gene expression levels and applications to microarray data analysis**. *Bioinformatics* 2002, **18:**1207-1215.

28. Quackenbush J: **Computational analysis of microarray data**. *Nat Rev Genet* 2001, **2:**418-427.

29. Yang YH, Dudoit S, Luu P, Lin DM, Peng V, Ngai J, Speed TP: **Normalization for cDNA microarray data: a robust composite method addressing single and multiple slide systematic variation**. *Nucleic Acids Res* 2002, **30:**e15.

30. Cleveland WS: **Robust locally weighted regression and smoothing scatterplots**. *J Amer Stat Assoc* 1979, **32:**829-836.

31. Bolstad BM, Irizarry RA, Astrand M, Speed TP: **A comparison of normalization methods for high density oligonucleotide array data based on variance and bias**. *Bioinformatics* 2003, **19:**185-193.

32. Tukey JW: *Exploratory Data Analysis*: Addison-Wesley; 1977.

33. van de Peppel J, Kemmeren P, van BH, Radonjic M, van LD, Holstege FC: **Monitoring global messenger RNA changes in externally controlled microarray experiments**. *EMBO Rep* 2003, **4:**387-393.

34. van BH, Holstege FC: **In control: systematic assessment of microarray performance**. *EMBO Rep* 2004, **5:**964-969.

35. Engelen K, Naudts B, De MB, Marchal K: **A calibration method for estimating absolute expression levels from microarray data**. *Bioinformatics* 2006, **22:**1251-1258.

36. Witten IH, Frank E: **Data Mining: Practical Machine Learning Tools and Techniques.**; 2005.

37. Eisen MB, Spellman PT, Brown PO, Botstein D: **Cluster analysis and display of genome-wide expression patterns**. *Proc Natl Acad Sci U S A* 1998, **95:**14863-14868.

38. Spellman PT, Sherlock G, Zhang MQ, Iyer VR, Anders K, Eisen MB, Brown PO, Botstein D, Futcher B: **Comprehensive identification of cell cycle-regulated genes of the yeast Saccharomyces cerevisiae by microarray hybridization**. *Mol Biol Cell* 1998, **9:**3273-3297.

39. Alon U, Barkai N, Notterman DA, Gish K, Ybarra S, Mack D, Levine AJ: **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* 1999, **96:**6745-6750.

40. Hartigan JA: *Clustering Algorithms*: John Wiley & Sons, Inc.; 1975.

41. Kaufman L, Rousseeuw PJ: *Finding Groups in Data: an Introduction to Cluster Analysis*: John Willey & Sons. Inc.; 1990.

42. Kohonen T: *Self-Organizing Maps*: Spring Series in Information Sciences. Spring.; 1995.

43. De Smet F, Mathys J, Marchal K, Thijs G, De MB, Moreau Y: **Adaptive quality-based clustering of gene expression profiles**. *Bioinformatics* 2002, **18:**735-746.

44. Hughes TR, Marton MJ, Jones AR, Roberts CJ, Stoughton R, Armour CD, Bennett HA, Coffey E, Dai H, He YD, Kidd MJ, King AM, Meyer MR, Slade D, Lum PY, Stepaniants SB, Shoemaker DD, Gachotte D, Chakraburtty K, Simon J, Bard M, Friend SH: **Functional discovery via a compendium of expression profiles**. *Cell* 2000, **102:**109-126.

45. Faith JJ, Driscoll ME, Fusaro VA, Cosgrove EJ, Hayete B, Juhn FS, Schneider SJ, Gardner TS: **Many Microbe Microarrays Database: uniformly normalized Affymetrix compendia with structured experimental metadata**. *Nucleic Acids Res* 2008, **36:**D866-D870.

46. Son CG, Bilke S, Davis S, Greer BT, Wei JS, Whiteford CC, Chen QR, Cenacchi N, Khan J: **Database of mRNA gene expression profiles of multiple human organs**. *Genome Res* 2005, **15:**443-450.

47. Su AI, Wiltshire T, Batalov S, Lapp H, Ching KA, Block D, Zhang J, Soden R, Hayakawa M, Kreiman G, Cooke MP, Walker JR, Hogenesch JB: **A gene atlas of the mouse and human protein-encoding transcriptomes**. *Proc Natl Acad Sci U S A* 2004, **101:**6062-6067.

48. Brazma A, Hingamp P, Quackenbush J, Sherlock G, Spellman P, Stoeckert C, Aach J, Ansorge W, Ball CA, Causton HC, Gaasterland T, Glenisson P, Holstege FC, Kim IF, Markowitz V, Matese JC, Parkinson H, Robinson A, Sarkans U, Schulze-Kremer S, Stewart J, Taylor R, Vilo J, Vingron M: **Minimum information about a microarray experiment (MIAME)-toward standards for microarray data**. *Nat Genet* 2001, **29:**365-371.

49. Sasik R, Woelk CH, Corbeil J: **Microarray truths and consequences**. *J Mol Endocrinol* 2004, **33:**1-9.

50. Fierro AC, Vandenbussche F, Engelen K, Van de Peer Y, Marchal K: **Meta Analysis of Gene Expression Data within and Across Species**. *Curr Genomics* 2008, **9:**525-534.

51. Bammler T, Beyer RP, Bhattacharya S, Boorman GA, Boyles A, Bradford BU, Bumgarner RE, Bushel PR, Chaturvedi K, Choi D, Cunningham ML, Deng S, Dressman HK, Fannin RD, Farin FM, Freedman JH, Fry RC, Harper A, Humble MC, Hurban P, Kavanagh TJ, Kaufmann WK, Kerr KF, Jing L, Lapidus JA, Lasarev MR, Li J, Li YJ, Lobenhofer EK, Lu X, Malek RL, Milton S, Nagalla SR, O'malley JP, Palmer VS, Pattee P, Paules RS, Perou CM, Phillips K, Qin LX, Qiu Y, Quigley SD, Rodland M, Rusyn I, Samson LD, Schwartz DA, Shi Y, Shin JL, Sieber SO, Slifer S, Speer MC, Spencer PS, Sproles DI, Swenberg JA, Suk WA, Sullivan RC, Tian R, Tennant RW, Todd SA, Tucker CJ, Van HB, Weis BK, Xuan S, Zarbl H: **Standardizing global gene expression analysis between laboratories and across platforms**. *Nat Methods* 2005, **2:**351-356.

52. Irizarry RA, Warren D, Spencer F, Kim IF, Biswal S, Frank BC, Gabrielson E, Garcia JG, Geoghegan J, Germino G, Griffin C, Hilmer SC, Hoffman E, Jedlicka AE, Kawasaki E, Martinez-Murillo F, Morsberger L, Lee H, Petersen D, Quackenbush J, Scott A, Wilson M, Yang Y, Ye SQ, Yu W: **Multiple-laboratory comparison of microarray platforms**. *Nat Methods* 2005, **2:**345-350.

53. Laule O, Hirsch-Hoffmann M, Hruz T, Gruissem W, Zimmermann P: **Web-based analysis of the mouse transcriptome using Genevestigator**. *BMC Bioinformatics* 2006, **7:**311.

54. Elfilali A, Lair S, Verbeke C, La RP, Radvanyi F, Barillot E: **ITTACA: a new database for integrated tumor transcriptome array and clinical data analysis**. *Nucleic Acids Res* 2006, **34:**D613-D616.

55. Rhodes DR, Kalyana-Sundaram S, Mahavisno V, Varambally R, Yu J, Briggs BB, Barrette TR, Anstet MJ, Kincead-Beal C, Kulkarni P, Varambally S, Ghosh D, Chinnaiyan AM: **Oncomine 3.0: genes, pathways, and networks in a collection of 18,000 cancer gene expression profiles**. *Neoplasia* 2007, **9:**166-180.

56. Pan F, Chiu CH, Pulapura S, Mehan MR, Nunez-Iglesias J, Zhang K, Kamath K, Waterman MS, Finch CE, Zhou XJ: **Gene Aging Nexus: a web database and data mining platform for microarray data on aging**. *Nucleic Acids Res* 2007, **35:**D756-D759.

57. Madeira SC, Oliveira AL: **Biclustering algorithms for biological data analysis: a survey**. *IEEE/ACM Trans Comput Biol Bioinform* 2004, **1:**24-45.

58. Bader GD, Heilbut A, Andrews B, Tyers M, Hughes T, Boone C: **Functional genomics and proteomics: charting a multidimensional map of the yeast cell**. *Trends Cell Biol* 2003, **13:**344-356.

59. Blais A, Dynlacht BD: **Constructing transcriptional regulatory networks**. *Genes Dev* 2005, **19:**1499-1511.

60. Faith JJ, Hayete B, Thaden JT, Mogno I, Wierzbowski J, Cottarel G, Kasif S, Collins JJ, Gardner TS: **Large-scale mapping and validation of Escherichia coli transcriptional regulation from a compendium of expression profiles**. *PLoS Biol* 2007, **5:**e8.

61. Greenbaum D, Luscombe NM, Jansen R, Qian J, Gerstein M: **Interrelating different types of genomic data, from proteome to secretome: 'oming in on function**. *Genome Res* 2001, **11:**1463-1468.

62. Chua G, Robinson MD, Morris Q, Hughes TR: **Transcriptional networks: reverse-engineering gene regulation on a global scale**. *Curr Opin Microbiol* 2004, **7:**638-646.

63. Ehrenberg M, Elf J, Aurell E, Sandberg R, Tegner J: **Systems biology is taking off**. *Genome Res* 2003, **13:**2377-2380.

64. Papin JA, Hunter T, Palsson BO, Subramaniam S: **Reconstruction of cellular signalling networks and analysis of their properties**. *Nat Rev Mol Cell Biol* 2005, **6:**99-111.

65. Wei GH, Liu DP, Liang CC: **Charting gene regulatory networks: strategies, challenges and perspectives**. *Biochem J* 2004, **381:**1-12.

66. Kerr MK, Martin M, Churchill GA: **Analysis of variance for gene expression microarray data**. *J Comput Biol* 2000, **7:**819-837.

67. Gentleman RC, Carey VJ, Bates DM, Bolstad B, Dettling M, Dudoit S, Ellis B, Gautier L, Ge Y, Gentry J, Hornik K, Hothorn T, Huber W, Iacus S, Irizarry R, Leisch F, Li C, Maechler M, Rossini AJ, Sawitzki G, Smith C, Smyth G, Tierney L, Yang JY, Zhang J: **Bioconductor: open software development for computational biology and bioinformatics**. *Genome Biol* 2004, **5:**R80.

68. Smyth GK: **Linear models and empirical bayes methods for assessing differential expression in microarray experiments**. *Stat Appl Genet Mol Biol* 2004, **3:**Article3.

69. Hilson P, Allemeersch J, Altmann T, Aubourg S, Avon A, Beynon J, Bhalerao RP, Bitton F, Caboche M, Cannoot B, Chardakov V, Cognet-Holliger C, Colot V, Crowe M, Darimont C, Durinck S, Eickhoff H, de Longevialle AF, Farmer EE, Grant M, Kuiper MT, Lehrach H, Leon C, Leyva A, Lundeberg J, Lurin C, Moreau Y, Nietfeld W, Paz-Ares J, Reymond P, Rouze P, Sandberg G, Segura MD, Serizet C, Tabrett A, Taconnat L, Thareau V, Van HP, Vercruysse S, Vuylsteke M, Weingartner M,

Weisbeek PJ, Wirta V, Wittink FR, Zabeau M, Small I: **Versatile gene-specific sequence tags for Arabidopsis functional genomics: transcript profiling and reverse genetics applications**. *Genome Res* 2004, **14:**2176-2189.

70. Smets B, De SP, Engelen K, Joossens E, Ghillebert R, Thevissen K, Marchal K, Winderickx J: **Genome-wide expression analysis reveals TORC1-dependent and -independent functions of Sch9**. *FEMS Yeast Res* 2008, **8:**1276-1288.

71. Surette MG, Miller MB, Bassler BL: **Quorum sensing in Escherichia coli, Salmonella typhimurium, and Vibrio harveyi: a new family of genes responsible for autoinducer production**. *Proc Natl Acad Sci U S A* 1999, **96:**1639-1644.

72. Surette MG, Bassler BL: **Quorum sensing in Escherichia coli and Salmonella typhimurium**. *Proc Natl Acad Sci U S A* 1998, **95:**7046-7050.

73. Taga ME, Miller ST, Bassler BL: **Lsr-mediated transport and processing of AI-2 in Salmonella typhimurium**. *Mol Microbiol* 2003, **50:**1411-1427.

74. Xavier KB, Miller ST, Lu W, Kim JH, Rabinowitz J, Pelczer I, Semmelhack MF, Bassler BL: **Phosphorylation and processing of the quorum-sensing molecule autoinducer-2 in enteric bacteria**. *ACS Chem Biol* 2007, **2:**128-136.

75. Taga ME, Semmelhack JL, Bassler BL: **The LuxS-dependent autoinducer AI-2 controls the expression of an ABC transporter that functions in AI-2 uptake in Salmonella typhimurium**. *Mol Microbiol* 2001, **42:**777-793.

76. Vendeville A, Winzer K, Heurlier K, Tang CM, Hardie KR: **Making 'sense' of metabolism: autoinducer-2, LuxS and pathogenic bacteria**. *Nat Rev Microbiol* 2005, **3:**383-396.

77. Gentleman RC, Carey VJ, Bates DM, Bolstad B, Dettling M, Dudoit S, Ellis B, Gautier L, Ge Y, Gentry J, Hornik K, Hothorn T, Huber W, Iacus S, Irizarry R, Leisch F, Li C, Maechler M, Rossini AJ, Sawitzki G, Smith C, Smyth G, Tierney L, Yang JY, Zhang J: **Bioconductor: open software development for computational biology and bioinformatics**. *Genome Biol* 2004, **5:**R80.

78. Tusher VG, Tibshirani R, Chu G: **Significance analysis of microarrays applied to the ionizing radiation response**. *Proc Natl Acad Sci U S A* 2001, **98:**5116-5121.

79. De Smet F, Moreau Y, Engelen K, Timmerman D, Vergote I, De MB: **Balancing false positives and false negatives for the detection of differential expression in malignancies**. *Br J Cancer* 2004, **91:**1160-1165.

80. Storey JD, Tibshirani R: **Statistical methods for identifying differentially expressed genes in DNA microarrays**. *Methods Mol Biol* 2003, **224:**149-157.

81. Pagano M, Gauvreau K: *Principles of Biostatistics*. second edition: Duxury Resource Center; 2000.

82. Karavolos MH, Bulmer DM, Winzer K, Wilson M, Mastroeni P, Williams P, Khan CM: **LuxS affects flagellar phase variation independently of quorum sensing in Salmonella enterica serovar typhimurium**. *J Bacteriol* 2008, **190:**769-771.

83. Tanay A, Sharan R, Kupiec M, Shamir R: **Revealing modularity and organization in the yeast molecular network by integrated analysis of highly heterogeneous genomewide data**. *Proc Natl Acad Sci U S A* 2004, **101:**2981-2986.

84. Van den Bulcke T, Lemmens K, Van de Peer Y, Marchal K: **Inferring transcriptional networks by mining 'omics' data**. *Current Bioinformatics* 2006**:**1-3.

85. Cheng Y, Church GM: **Biclustering of expression data.**; 2000:93-103.

86. Friedman N, Getoor L, Koller D, Pfeffer A: **Learning probabilistic relational models.**; 1999:1300-1309.

87. Getoor L, Friedman N, Koller D, Taskar B: **Learning probabilistic models of relational structure.** In *the 18th International Conference on Machine Learning.*; 2001:170-177.

88. Koller D, Pfeffer A: **Probabilistic frame-based systems.**; 1998:580-587.

89. D'Ambrosio B, Jorgensen J, Altenforf E: **Ecosystem analysis using probabilistic relational modeling.** In *The IJCAI-2330 Workshop on Learning Statistical Models from Relational Data*; 2003.

90. Getoor L, Segal E, Taskar B, Koller D: **Probabilistic models of text and link structure for hypertext classification.** In *IJCAI01 Workshop on Text Learning: Beyond Supervision.*; 2001.

91. Newton J, Greiner R: **Hierarchical probabilistic relational models for collaborative filtering.**; 2004:249-263.

92. Segal E, Taskar B, Gasch A, Friedman N, Koller D: **Rich probabilistic models for gene expression**. *Bioinformatics* 2001, **17 Suppl 1:**S243-S252.

93. Segal E, Battle A, Koller D: **Decomposing gene expression into cellular processes**. *Pac Symp Biocomput* 2003**:**89-100.

94. Segal E, Yelensky R, Koller D: **Genome-wide discovery of transcriptional modules from DNA sequence and gene expression**. *Bioinformatics* 2003, **19 Suppl 1:**i273-i282.

95. Ben-Dor A, Chor B, Karp R, Yakhini Z: **Discovering local structure in gene expression data: the order-preserving submatrix problem**. *J Comput Biol* 2003, **10:**373-384.

96. Bergmann S, Ihmels J, Barkai N: **Iterative signature algorithm for the analysis of large-scale gene expression data**. *Phys Rev E Stat Nonlin Soft Matter Phys* 2003, **67:**031902.

97. Dhollander T, Sheng Q, Lemmens K, De MB, Marchal K, Moreau Y: **Query-driven module discovery in microarray data**. *Bioinformatics* 2007, **23:**2573-2580.

98. Yang J, Wang H, Wang W, Yu P: **Enhanced biclustering on expression data.**; 2003:321-327.

99. Battle A, Segal E, Koller D: **Probabilistic discovery of overlapping cellular processes and their regulation**. *J Comput Biol* 2005, **12:**909-927.

100. Lazzeroni L, Owen A: **Plaid models for gene expression data**. *Statistica Sinica* 1999, **12:**61-68.

101. Schwarz G: **Estimating the dimension of a model.**; 1978:461-464.

102. Akaike H: **A new look at the statistical model identification.**; 1974:716-723.

103. Dempster AP, Laird NM, Rubin DB: **Maximum likelihood from incomplete data via the EM algorithm**. *Journal of the Royal Statistical Society Series B* 1977, **39:**1-38.

104. Wu JFJ: **On the convergence properties of the EM algorithm.**; 1983:95-103.

105. Prelic A, Bleuler S, Zimmermann P, Wille A, Buhlmann P, Gruissem W, Hennig L, Thiele L, Zitzler E: **A systematic comparison and evaluation of biclustering methods for gene expression data**. *Bioinformatics* 2006, **22:**1122-1129.

106. Owen AB, Stuart J, Mach K, Villeneuve AM, Kim S: **A gene recommender algorithm to identify coexpressed genes in C. elegans**. *Genome Res* 2003, **13:**1828-1837.

107. Salgado H, Santos A, Garza-Ramos U, van HJ, Diaz E, Collado-Vides J: **RegulonDB (version 2.0): a database on transcriptional regulation in Escherichia coli**. *Nucleic Acids Res* 1999, **27:**59-60.

108. Gama-Castro S, Jimenez-Jacinto V, Peralta-Gil M, Santos-Zavaleta A, Penaloza-Spinola MI, Contreras-Moreira B, Segura-Salazar J, Muniz-Rascado L, Martinez-Flores I, Salgado H, Bonavides-Martinez C, Abreu-Goodger C, Rodriguez-Penagos C, Miranda-Rios J, Morett E, Merino E, Huerta AM, Trevino-Quintanilla L, Collado-Vides J: **RegulonDB (version 6.0): gene regulation model of Escherichia coli K-12 beyond transcription, active (experimental) annotated promoters and Textpresso navigation**. *Nucleic Acids Res* 2008, **36:**D120-D124.

109. Segal E, Shapira M, Regev A, Pe'er D, Botstein D, Koller D, Friedman N: **Module networks: identifying regulatory modules and their condition-specific regulators from gene expression data**. *Nat Genet* 2003, **34:**166-176.

110. Cookson W, Liang L, Abecasis G, Moffatt M, Lathrop M: **Mapping complex disease traits with global gene expression**. *Nat Rev Genet* 2009, **10:**184-194.

Publication list

Journal papers

Published

- **Zhao H**., Engelen K., De Moor B., and Marchal K. *CALIB: a BioConductor package for estimating absolute expression levels from two-color microarray data.* Bioinformatics 2007; 23(13) 1700-1701. doi: 10.1093/bioinformatics/btm159

- Janssens JC, Steenackers H, Robijns S, Gellens E, Levin J, **Zhao H**, Hermans K, De Coster D, Verhoeven TL, Marchal K, Vanderleyden J, De Vos DE, De Keersmaecker SC. 2008. *Brominated furanones inhibit biofilm formation by Salmonella enterica serovar Typhimurium.* Appl Environ Microbiol. 74: 6639-6648.

- Thijs IVM, De Keersmaecker S, Fadda A, Engelen K, **Zhao H**, McClelland M, Marchal K, Vanderleyden J. *Delineation of the Salmonella Typhimurium HilA regulon through genome-wide location and transcript analysis.* J Bacteriol. 2007 Jul; 189(13) 4587-96

- Thijs IVM, De Keersmaecker S, Fadda A, Engelen K, **Zhao H**, McClelland M, Marchal K, Vanderleyden J. *Combining omics data to unravel the regulatory network controlling Salmonella invasion of epithelial cells.* Commun Agric Appl Biol Sci. 2007. 72: 55-59.

Submitted or in preparation

- Thijs IM, **Zhao H.**, De Weerdt A., Engelen K., Schoofs G., De Coster D., McClelland M., Vanderleyden J., Marchal K., De Keersmaecker SCJ.. *The AI-2 dependent regulator LsrR has a limited regulon in Salmonella Typhimurium.* Submitted.

- Anyiawung FC., Verbeke G., Valkenborg D., **Zhao H.**, Marchal K., Engelen K. *Statistical approach to screen gene expression data for regulators responsible for the differential gene expression observed between two conditions.* Submitted.

- **Zhao H**., Van den Bulcke T., De Smet R., Cloots L., Engelen K., De Moor B., and Marchal K. *Efficient query-driven biclustering of gene expression data using Probabilistic Relational Models.* In preparation.

- De Keersmaecker SCJ., **Zhao H.**, Sonck KAJ., Thijs IM., De Coster D., van Boxel N., Engelen K., Northen T., Vanderleyden J., Marchal K. *Integrating high-throughput data reveals important role for AI-2 in Salmonella enterica serovar Typhimurium flagellar phase variation.* In preparation.