

KATHOLIEKE UNIVERSITEIT LEUVEN
FACULTEIT TOEGEPASTE WETENSCHAPPEN
DEPARTEMENT ELEKTROTECHNIEK
Kasteelpark Arenberg 10, 3001 Leuven (Heverlee)

MICROARRAY COMPENDIA AND THEIR IMPLICATIONS FOR BIOINFORMATICS SOFTWARE DEVELOPMENT

Promotoren:
Prof. dr. ir. B. De Moor
Prof. dr. ir. Y. Moreau

Proefschrift voorgedragen tot
het behalen van het doctoraat
in de toegepaste wetenschappen

door

Steffen DURINCK

May 2006



KATHOLIEKE UNIVERSITEIT LEUVEN
FACULTEIT TOEGEPASTE WETENSCHAPPEN
DEPARTEMENT ELEKTROTECHNIEK
Kasteelpark Arenberg 10, 3001 Leuven (Heverlee)

MICROARRAY COMPENDIA AND THEIR IMPLICATIONS FOR BIOINFORMATICS SOFTWARE DEVELOPMENT

Jury:
Prof. dr. ir. G. De Roeck, voorzitter
Prof. dr. ir. B. De Moor, promotor
Prof. dr. ir. Y. Moreau, co-promotor
Prof. dr. ir. J. Vandewalle
Prof. dr. ir. J. Vanderleyden
Prof. dr. ir. K. Marchal
Dr. W. Huber
Prof. dr. F. Schuit

Proefschrift voorgedragen tot
het behalen van het doctoraat
in de toegepaste wetenschappen

door

Steffen DURINCK

U.D.C. 681.3*D, 519.23 and 681.3*J3 May 2006

©Katholieke Universiteit Leuven - Faculteit Toegepaste Wetenschappen Arenbergkasteel, B-3001 Heverlee (Belgium)
Alle rechten voorbehouden. Niets uit deze uitgave mag vermenigvuldigd en/of openbaar gemaakt worden door middel van druk, fotocopie, microfilm, elektronisch of op welke andere wijze ook zonder voorafgaande schriftelijke toestemming van de uitgever.
All rights reserved. No part of the publication may be reproduced in any form by print, photoprint, microfilm or any other means without written permission from the publisher.

D/2006/7515/26
ISBN 90-5682-692-1

Acknowledgements

Most of all, I would like to thank my promotor Prof. Dr. Bart De Moor. He gave me the opportunity to pursue a Ph.D. in his research group and supported me throughout the years. My co-promotor Prof. Dr. Yves Moreau has been equally important during my study as he kept me on the right track and provided new ideas and help at all times for which I am very grateful. I would like to thank my promotor and co-promotor for their careful reading of this thesis and providing corrections and suggestions for improvement of this work.
I would like to acknowledge Prof. Dr. Joos Vandewalle and Prof. Dr. Jos Vanderleyden who, as my assessors, followed my research progress, provided feedback on the draft of this thesis, and are also part of my jury. Further is it an honor to have Prof. Dr. Kathleen Marchal, Prof. Dr. Frans Schuit, Dr. Wolfgang Huber and Prof. Dr. Guido De Roeck as members of my jury.
Next, I would like to thank all my colleagues at ESAT-SCD and especially Joke Allemeersch. Joke and I were responsible to meet the deliverables demanded by the CAGE project. Joke's background in statistics was the perfect complement of my bioinformatics and biology knowledge which made us a great team. I would like to acknowledge Dr. Qizheng Sheng, a former colleague, for her help with LaTeX, the software system that was used to write this thesis.
The RMAGEML package was developed in collaboration with Dr. Vincent Carey of the Channing laboratory. I'm grateful to Dr. Carey for providing good ideas that made possible to get Java and R to work together.
During my Ph.D. I did two internships at the European Bioinformatics Institute (EBI) in Hinxton, Cambridge, UK. I would like to thank Dr. Alvis Brazma, group leader of the EBI microarray team for giving me this unique opportunity. I would like to thank Dr. Helen Parkinson and Dr. Gaurab Mukherjee and the rest of the microarray curation team for the many insightful discussions on MAGE-ML. Another microarray team member who I would like to acknowledge is Misha Kapushesky, who let me co-develop Expression Profiler.
During my second stay at EBI, I worked closely together with Dr. Wolfgang

Huber of the Huber group at EBI. I would like to acknowledge Dr. Huber for the great collaboration on the biomaRt package development. Further I would like to thank Dr. Arek Kasprzyk of the Ensembl team who heads the BioMart project for the fruitful discussions on BioMart systems. I would also like to thank Dr. Sean Davis of NIH, who as one of the first biomaRt users, provided very helpful user reports and later contributed some of the code.
Next, I would like to acknowledge Dr. Tinneke Denayer of the University of Ghent for collaborating on the *Xenopus* experiment and providing the data.
I would like to thank all partners of the CAGE consortium for their collaboration during the CAGE project.
Finally I would like to thank my parents who made this all possible, my two sisters for their support and corrections to the dutch summary, and my wife Ogechi, who has always been there for me even when living on different continents.

Abstract

In the last decade microarray technology has become a widely used technology for the study of gene expression. Early experiments contained only a few hybridizations, studying a small set of related samples. Now the age of high-throughput microarray data production has started and large data sets, also known as microarray compendia, covering hundreds to thousands of samples are being produced. The main project of this thesis was to preprocess microarray data generated by the European demonstration project, CAGE, which aimed to build a large compendium of gene expression data for the plant model species *Arabidopsis thaliana*. The transition to large-scale data production has brought new bioinformatics challenges, and tools are needed to efficiently exchange, preprocess, and analyze the generated data. These challenges were addressed in this Ph.D. and the CAGE project provided the ultimate setting for these developments. The project produced well-annotated microarray data that are stored and exchanged using the recently developed MAGE-ML data format. At the time no software was available to integrate this MAGE-ML data in a data analysis environment. Therefore we developed a software package (RMAGEML) that integrates MAGE-ML format data in the statistical programming environment, R. The RMAGEML package is now part of Bioconductor, a popular open source project for the analysis of genomics data in R. CAGE delivered microarray data in a high-throughput manner and an automatic preprocessing pipeline was developed to process these thousands of microarray experiments. Normalized CAGE microarray data are made available to the community via ArrayExpress, a microarray data repository located at the European Bioinformatics Institute (EBI). ArrayExpress is tightly linked to Expression Profiler, a web application for microarray data analysis. This thesis also contributed in the development of new highly interactive visualization modules for Expression Profiler. Together with the availability of high-quality genome sequence information and other resources, large-scale microarray data sets provide a powerful means to understand biology at a systemic level. Com-

prehensive microarray analyses integrate a variety of available public biological data other than microarray data. This data integration was another challenge addressed in this Ph.D. and a second Bioconductor package, biomaRt, was developed to integrate multiple genomic data resources in R. This software package creates a powerful link between public biological data resources and microarray data analysis.

Samenvatting

In de laatste tien jaar evolueerde microarray technologie pijlsnel van een nieuwe experimentele techniek tot een veel gebruikte technologie voor de studie van genexpressie. De eerste microarray experimenten waren kleinschalig en bevatten slechts enkele te onderzoeken stalen. Nu echter is het tijdperk van grootschalige microarray data productie aangebroken en huidige datasets bevatten honderden tot duizenden stalen. Het hoofdproject in dit doctoraat was de verwerking van microarray gegevens gegenereerd door het Europees demonstratieproject, CAGE. CAGE heeft als doel het bouwen van een grootschalig compendium van genexpressie gegevens voor *Arabidopsis thaliana*. De transitie naar grootschalige productie van microarray gegevens, bracht nieuwe uitdagingen mee in de bioinformatica. Nieuwe software is nodig om de gegevens op een efficiente wijze uit te wisselen, te verwerken en te analyseren. Deze uitdagingen werden in dit doctoraat uitgewerkt en CAGE vormde het ultieme project dat deze ontwikkelingen mogelijk maakte. CAGE produceert goed geannoteerde microarray gegevens, dewelke opgeslaan worden in het recentelijk ontwikkelde MAGE-ML formaat. Bij de start van het project was er geen software beschikbaar om MAGE-ML data te integreren in een data analyse omgeving. Hiervoor ontwikkelden we het RMAGEML softwarepakket, dat het MAGE-ML formaat integreert in de statistische programmeeromgeving, R. Het RMAGEML softwarepakket is nu onderdeel van Bioconductor, een populair open source project voor de analyse van genomische data in R. CAGE genereert microarray gegevens op een grootschalige wijze en een automatische voorverwerking pijplijn was ontwikkeld voor de verwerking van deze duizenden microarray experimenten. De verwerkte CAGE microarray gegevens worden voor de gemeenschap beschikbaar gemaakt via ArrayExpress, een microarray databank van het Europees Bioinformatica Instituut (EBI). ArrayExpress is nauw verbonden met Expression Profiler, een web applicatie voor de analyse van microarray experimenten. Deze thesis ontwikkelde nieuwe, interactieve visualisatie modules voor Expression Profiler.

Samen met de beschikbare genoomsequenties en andere biologische informatiebronnen, vormen grootschalige microarray experimenten een belangrijk middel voor biologisch onderzoek op een systemisch niveau. Bij de analyse van microarray gegevens speelt de integratie van verscheidene biologische informatiebronnen een grote rol. Deze gegevensintegratie was een uitdaging in dit doctoraat en een tweede Bioconductor software pakket, biomaRt, werd ontwikkeld. Het biomaRt softwarepakket maakt de integratie van verscheidene biologische gegevensbronnen in een statistische omgeving mogelijk en vormt op deze wijze een belangrijke link tussen biologische data bronnen en microarray data analyse.

Contents

Acknowledgements **i**

Abstract **iii**

Samenvatting **v**

Contents **vii**

List of publications **xi**

List of abbreviations **xiii**

1 Introduction **1**

1.1 Gene expression compendia and functional genomics 1

1.2 Microarray data exchange 5

1.3 The Bioconductor project 5

1.4 Dissertation outline 7

2 Gene expression and microarrays **11**

2.1 Biology of gene expression 12

2.1.1 Regulatory proteins and motifs 14

2.1.2 Genomic DNA variation and DNA methylation 15

2.1.3 Histone modification 16

2.1.4 DNA amplification and deletion 17

2.1.5 Translational control 17

2.1.6 Protein degradation 18
2.1.7 Importance of gene expression 18
2.2 Microarray technology 21
2.2.1 Introduction 21
2.2.2 Two-color spotted microarrays 22
2.2.3 High-density oligonucleotide arrays 22
2.3 Preprocessing of microarray data 25
2.3.1 Quality assessment 25
2.3.2 Normalization of two-color microarray data 27
2.3.3 Normalization of Affymetrix GeneChips 28
2.4 Finding differentially expressed genes 32
2.4.1 Fold change 33
2.4.2 *t*-test and ANOVA 33
2.4.3 Limma 33
2.4.4 SAM 34
2.4.5 Multiple testing 34
2.4.6 Visualizing differential expression 34

3 Microarray data exchange **37**
3.1 MIAME 38
3.2 MAGE 39
3.2.1 MAGE-OM 40
3.2.2 XML and MAGE-ML 45
3.2.3 MAGEstk 47
3.3 MAGE compliant microarray databases 49
3.3.1 ArrayExpress 49
3.3.2 MIAMExpress 49
3.3.3 BASE 49
3.3.4 Other MAGE compliant databases 52
3.4 RMAGEML 52
3.4.1 Description 53

3.4.2 Architecture: lightweight R to Java interface 54
3.4.3 Usage 56
3.5 Discussion 57

4 Compendium of Arabidopsis Gene Expression 59
4.1 The CAGE project 59
4.2 *Arabidopsis thaliana* 60
4.3 CATMA array 62
4.4 CAGE sample overview and sample annotation 64
4.5 CAGE experiment design 67
4.6 Normalization 69
4.7 Quality assessment 71
4.8 Preprocessing pipeline 73
4.8.1 CAGE data flow 73
4.8.2 CAGE pipeline architecture 73
4.8.3 CAGE preprocessing web application 75
4.9 CAGE sample production 75
4.10 AtGenExpress 77
4.11 Discussion 79

5 Integration of public data resources with biomaRt 81
5.1 Annotation databases 82
5.1.1 Ensembl 82
5.1.2 Human variation data 84
5.1.3 VEGA 84
5.1.4 UniProt 84
5.1.5 Wormbase 85
5.1.6 Gramene 85
5.2 BioMart 85
5.3 biomaRt 87
5.3.1 Annotation and Bioconductor 87

5.3.2 biomaRt 87
5.3.3 Simple biomaRt functions 89
5.3.4 Advanced biomaRt functions and data mining 91
5.4 Discussion 91

6 Contributions to microarray data analysis tools 95
6.1 Expression Profiler: next generation 96
6.1.1 Data visualization in Expression Profiler 96
6.1.2 Scalable Vector Graphics 96
6.1.3 Implementing SVG in EP:NG 98
6.1.4 Current status EP:NG visualization 98
6.2 At-Endeavour 99
6.2.1 Endeavour 99
6.2.2 At-Endeavour 100
6.3 Discussion 101

7 Discussion and conclusion 103
7.1 Achievements 103
7.2 Future directions 106

Nederlandse samenvatting 109

Appendix A: case study A-1

Appendix B: vignette RMAGEML software package B-1

Appendix C: vignette of biomaRt software package C-1

Bibliography 123

Curriculum vitae 139

List of publications

Kapushesky, M., Kemmeren, P., Culhane, A.C., **Durinck, S.**, Ihmels, J., Korner, C., Kull, M., Torrente, A., Sarkans, U., Vilo, J. and Brazma, A. (2004). Expression Profiler: next generation-an online platform for analysis of microarray data. *Nucleic Acids Research*, 32, W465-W470.

Durinck, S., Allemeersch, J., Carey, V.J., Moreau, Y. and De Moor, B. (2004). Importing MAGE-ML format microarray data into BioConductor. *Bioinformatics*, 20(18), 3641-2.

Hilson, P., Allemeersch, J., Altmann, T., Aubourg, S., Avon, A., Beynon, J., Bhalero, R.P., Bitton, F., Caboche, M., Cannoot, B., Chardakov, V., Cognet-Holliger, C., Colot, V., Crowe, M., Darimont, C., **Durinck, S.**, Eickhoff, H., Falcon de Languevialle, A., Farmer, E.E., Grant, M., Kuiper, M.T.R., Lehrach, H., Leon, C., Leyva, A., Lundenberg, J., Lurin, C., Moreau, Y., Nietfeld, W., Serizet, C., Tabrett, A., Taconnat, L., Thareau, V., Van Hummelen, P., Vercruysse, S., Vuylsteke, M., Weingartner, M., Weisbeek, P.J., Wirta, V., Wittink, F.R.A., Zabeau, M., Small, I. (2004) Versatile Gene-specific Sequence Tags for arabidopsis functional genomics : Transcript profiling and reverse genetics applications. *Genome Research*, 14(10B), 2176-89.

Allemeersch J, **Durinck, S.**, Vanderhaeghen R, Alard P, Maes R, Seeuws K, Bogaert T, Coddens K, Deschouwer K, Van Hummelen P, Vuylsteke M, Moreau Y, Kwekkeboom J, Wijfjes AH, May S, Beynon J, Hilson P, Kuiper MT. (2005). Benchmarking the CATMA Microarray. A Novel Tool for Arabidopsis Transcriptome Analysis. *Plant Physiology*, 137(2), 588-601.

Durinck, S., Moreau, Y., Kasprzyk, A., Davis, S., De Moor, B., Brazma A. and Huber W. (2005). BioMart and Bioconductor: a powerful link between

biological databases and microarray data analysis. *Bioinformatics*, 21, 3439-3440.

Mukherjee, G., Abeygunawardena, N., Parkinson, H., Contrino, S., **Durinck, S.**, Farne, A., Holloway, E., Lilja, P., Moreau ,Y., Oezcimen, A., Rayner, T., Sharma, A., Brazma, A., Sarkans, U., Shojatalab, M. (2005) Plant-based microarray data at the European Bioinformatics Institute. Introducing AtMIAMExpress, a submission tool for Arabidopsis gene expression data to ArrayExpress. *Plant Physiol.* 2005 Oct;139(2):632-6.

List of abbreviations

APC	Adenomatous Polyposis Coli
API	Application Programming Interface
CAGE	Compendium of *Arabidopsis* Gene Expression
cDNA	copy DNA
CSHL	Cold Spring Harbor Laboratory
DNA	DeoxyriboNucleic Acid
Dsh	Dishevelled
DTD	Document Type Definition
EBI	European Bioinformatics Institute
EP	Expression Profiler
EP:NG	Expression Profiler: Next Generation
FTP	File Transfer Protocol
GO	Gene Ontology
HTTP	HyperText Transfer Protocol
IM	Ideal Mismatch
JNI	Java Native Interface
JVM	Java Virtual Machine
LEF	Lymphoid Enhancer-binding Factor 1
MAGE	MicroArray Gene Expression group
MAGE-ML	MicroArray Gene Expression Modeling Language
MAGE-OM	MicroArray Gene Expression Object Model
MBEI	Model-Based Expression Index
MGED	Microarray Gene Expression Data society
MIAME	Minimum Information About a Microarray Experiment
MM	MisMatch
mRNA	messenger RNA
OMIM	Online Mendelian Inheritance in Man
PCR	Polymerase Chain Reaction
PO	Plant Ontology

PM	Perfect Match
qPCR	quantitative PCR
QTL	Quantitative Trait Loci
RMA	Robust Multi-array Average
RNA	RiboNucleic Acid
RNAi	RNA interference
SNP	Single Nucleotide Polymorphism
SQL	Structured Query Language
SVG	Scalable Vector Graphics
TCF	T-Cell specific transcription Factor
UTR	UnTranslated Region
W3C	World Wide Web Consortium
XML	eXtensible Mark-up Language

Chapter 1

Introduction

At present, genomes are being sequenced at a high rate and genomics research is gradually moving from gene discovery to functional characterization of the thousands of genes in these genomes, also known as functional genomics. Microarrays are effective tools that can be used in these functional genomics studies. Recently there has been a shift from small-scale microarray experiments, that only cover a few samples, to large-scale experiments covering hundreds to thousands of samples. These large-scale microarray studies are also known as compendia. The increase in data production brought new bioinformatics challenges such as efficient data exchange, large scale preprocessing and development of software tools that enable comprehensive analysis and make use of all available biological knowledge. These challenges have been addressed in this thesis and the outline with which this chapter ends, provides an overview of these achievements.

1.1 Gene expression compendia and functional genomics

Genes are regions of the genome that are associated with regulatory regions and are transcribed into RNA sequences. A majority of these genes, contain the information to build functional proteins, that in their turn are the building blocks of an organism. Since decades researchers have been studying the *expression* of genes, which is determining the concentration of a particular gene transcript or of the corresponding protein in the cell. Early techniques, such as Northern blotting, were mainly used to detect if one gene was expressed in the

sample or not. Later techniques, such as differential display [78], AFLP [7], and SAGE [117] enabled to study expression patterns of multiple genes at a time. However sequencing was needed to identify each gene of interest, making the process slow. At that time not many genes were known and genome sequences not yet available. In late 1990s, a novel technique was introduced to measure gene expression: the microarray. Microarrays involve amplification of sequences and spotting or synthesizing these sequences on a glass slide. With many genomes sequenced today, the use of microarrays has gone through an explosion. The advent of microarray technology has caused a revolution in the way we study the expression of genes and now thousands of genes can be studied in parallel. Currently the sequence of genes is usually known or can be predicted computationally but many genes still lack any information about their function. It is a widely accepted idea that genes with similar function also have similar expression patterns [36]. Thus if the expression pattern of a gene with unknown function, clusters* with the profiles of genes of which some are functionally characterized, the gene with unknown function can be expected to have a related function. As microarrays are a good means to measure these expression patterns, many microarray experiments now aim to functionally annotate coding sequences, this is also known as *functional genomics*.

In the early days of microarray technology, small-scale experiments where performed containing only few hybridizations. Soon however the scale of the experiments increased drastically. Recently the transcriptome† is becoming viewed as a separate biological entity. In contrast to the genome, of which the sequence is the same in every cell of an organism, there are many transcriptomes each specific to a certain cell and condition. Similarly as for the genome, collections of transcriptomes are being gathered from many species and called gene expression compendia. One of the first data sets that can be regarded as a true compendium was produced by Hughes *et al.* in 2000 [53]. This data set contained 300 samples of *Saccharomyces cerevisae* (budding yeast) covering diverse mutations and chemical treatments. 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. At that time, analysis of mutants depended on easily scored phenotypes, such as unusual appearance or sensitivity to certain culture conditions. Whole-genome expression arrays provide molecular phenotypes of the cell even for conditions were

*a cluster can be defined as a set of genes that show a similar expression behaviour over a set of conditions.

†The term *transcriptome* can be defined as the collection of all transcripts that are expressed in a given biological condition in a cell or tissue sample and the expression levels of those transcripts.

the conventional phenotypes do not exist. A change in nutrient conditions for example, might not result in a measurable phenotype as slower growth, however at the molecular level gene expression could have changed to adapt to the new growth circumstances. A fundamental advantage over the conventional assays available at that time was thus that a compendium approach substitutes a single genome-wide expression profile in place of many conventional, often tedious, assays that measure only a single cellular parameter [53]. Furthermore a compendium for example used to characterize mutants can also be used to characterize other perturbations, such as treatments with pharmaceutical compounds.
Soon after this first compendium paper, new efforts were started and delivered compendia for a variety of organisms.

In 2004, Su *et al.* [111] published a compendium containing 79 human samples and 61 mouse tissues, providing measurements of over 30,000 target sequences. They found that 16,454 and 17,924 of their target sequences were expressed in at least one tissue of human and mouse respectively. Less than 1% of the human target sequences were ubiquitously expressed in all tissues and on average about 8,200 genes were expressed in a single tissue [111]. By mapping the expression levels to genomic location of the genes, they could also identify genomic regions (loci) where genes that are located close to each other, show correlated expression profiles.
Son *et al.* [105] recently constructed a gene expression database capturing 18,927 mRNA transcript levels for 19 different organs from 158 normal human tissues from 30 donors. They showed that despite the diversity of the samples (e.g., donors had different age, sex, ethnicity), the expression profiles of the same organs cluster together. The gene expression profiles also reflected major organ-specific functions at the molecular level. Figure 1.1 depicts a result from this compendium, showing a heatmap with expression patterns of organ specific genes. Lastly they show how compendia of normal tissues can be used to identify targets for therapy or diagnosis by comparing disease state tissues with the compendium. For example the authors found 19 significantly differentially expressed genes that had associated druggable GO[‡] terms when comparing neuroblastoma samples with the normal expression compendium.

[‡]Gene Ontology (GO) is explained in Chapter 6.

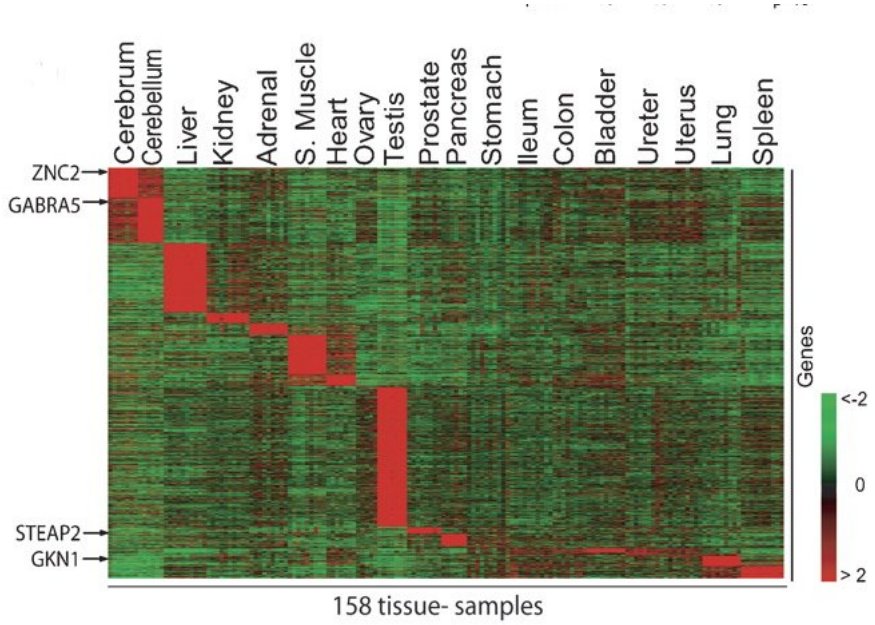


Figure 1.1: Example of a microarray compendium, involving 19 different organs from 158 normal human tissues from 30 donors. Only the expression profiles of genes that are found to be tissue specific are plotted. Red indicates upregulation, green indicates downregulation. Note that for each tissue there are a number of genes that are specifically upregulated in that tissue and not in other tissues. Figure from Son *et al.* [105].

1.2 Microarray data exchange

The introduction of microarray technology caused an explosion of data generated by these expression experiments and opened a new branch of bioinformatics research dedicated to the tracking, processing, and analysis of the experiments and the generated data. As experiments are gradually becoming more extensive, covering thousands of genes and a sizable number of samples, a single study mostly does not exploit the full content of the data anymore. It becomes more and more important that the data are well-annotated so that researchers, other than the data producers, can use the available microarray data in their own research. As microarray data are complex, a set of guidelines (MIAME) [18, 108] was set up, which determined what information should be available about a microarray experiment to fully understand and use data generated by others and confirm their findings. Together with these guidelines, a microarray data exchange format (MAGE-ML) [106] was developed that could hold this information and to communicate it to other researchers in a way directly processable by computers. ArrayExpress was the first microarray data repository to store this well-annotated MAGE-ML format data and currently contains over 30,000 hybridizations. A remaining challenge however was to develop software tools that can handle this type of data in a microarray data analysis environment.

1.3 The Bioconductor project

The most popular set of tools for processing and the analysis of microarray experiments is the Bioconductor project (http://www.bioconductor.org). Bioconductor is an open source and development software project for the analysis and comprehension of genomic data [41, 43, 55, 42, 30, 61]. The goals of Bioconductor can be summarized as

- To provide access to a wide range of powerful statistical and graphical methods for the analysis of genomic data.
- To facilitate the integration of biological metadata in the analysis of experimental data (e.g., annotation data from Ensembl).
- To allow the rapid development of extensible, scalable, and interoperable software.
- To promote high-quality documentation and reproducible research.

- To provide training in computational and statistical methods for the analysis of genomic data.

Bioconductor is implemented in the open source statistical computing environment R and has a six-month release schedule. Originally Bioconductor started as a set of packages centered around preprocessing microarray data but has since grown very fast. In recent years the popularity of this project has risen enormously. An estimate of the number of users can be made from the following statistics: in June 2005, the Bioconductor web site had 9,009 unique hits and 19,483 visits (data Seth Falcon). The success of the project also shows in the number of commercial analysis programs that on request of its users include a plug-in for R (e.g., SpotFire®, GeneSpring®) and Insightful®, which is a company that selects successful Bioconductor packages and implements them in their commercial S-Plus analogue ArrayAnalyzer®. The main goal of the Bioconductor project is the creation of an open source, durable and flexible software development and deployment environment [43]. This software development strategy started with the Free Software Foundation and the GNU project in the 1980s, which was a try-out to provide a free and open implementation of the UNIX operating system. The idea is that software should be available to everyone in order to test, justify, replicate, and further develop upon. This strategy led for example to the popular open source Linux kernel we know today. A key factor to the success of the Linux kernel is its modular design, which allows parallel development of the code. This way developers are responsible for only parts of the project and indirectly contribute to building a complex system. A similar strategy is used in the development of R, which consists of a central core to which individual add-on packages, each specialized at a specific task, can be added [43]. Many new analysis methods are developed in this software environment and when the method is published, the software is put into a package. For example many of the methods to normalize Affymetrix GeneChip data are collected in the *affy* package [39]. These add-on packages are put together in repositories, such as the Bioconductor project. This highlights one of the advantages of using Bioconductor: new methods are rapidly developed and soon after (or even before) publication, they can be used by the community in their own research. By contrast, integrating new methods in commercial software often is slow. Two of the software packages developed in this Ph.D. are now part of the Bioconductor project. During the development of these packages I had the opportunity to work in close collaboration with several Bioconductor core developers and was involved in national and international Bioconductor workshops. The presence of our software packages in Bioconductor exposes this work to a large user community and guarantees

its perennity.

1.4 Dissertation outline

Chapter 2 starts by introducing the term *gene expression* and then discusses different ways the cell uses to regulate the expression of genes. After showing the importance of gene expression in biology and some of its applications in human health, microarrays are introduced as a technology to study gene expression. Some of the main technologies are reviewed along with preprocessing of the data they generate and how differences in gene expression can be found.
Microarrays produce a huge amount of complex data and exchanging these data is a true challenge. In a following chapter (Chapter 3), standards in microarray data exchange are discussed. Public microarray data repositories, such as ArrayExpress, contain large volumes of MAGE-ML data as described above. Despite the availability of thousands of MAGE-ML format data sets, the use of this well-annotated data for subsequent analysis is limited by the lack of software tools that can handle this data format. The challenge that we addressed in Chapter 3 was to integrate this complex data structure in R, a popular environment for microarray data analysis.
In 2002 a European Framework 5 project started to build a compendium of *Arabidopsis thaliana* gene expression (CAGE). This project played a main role in this Ph.D. and aimed at producing 4,000 microarray datasets each reporting expression information on about 25,000 genes, creating a total of 100,000,000 datapoints. The CAGE compendium uses the MAGE-ML format and had as goal to provide a well-annotated compendium for the community to use in future plant research. Our group was responsible for preprocessing the microarray data generated by the CAGE project. Processing this large amount of data proved to be a major challenge in this Ph.D. and we developed an automated preprocessing pipeline for this purpose. This pipeline is to our knowledge the first that fully supports the MAGE-ML format. The CAGE project and development of the automatic preprocessing pipeline are described in Chapter 4.
We selected R and Bioconductor as environment to build a high-throughput microarray preprocessing pipeline. This way we could benefit from the advanced methods that are available and the possibility to modify and extend the existing method implementations to our own needs. Aware that our tools filled gaps in the existing software packages, we decided to make them as general as possible so that they could become Bioconductor packages of themselves. Chapter 3 and Chapter 5 describe the development of two such packages: RMAGEML

and biomaRt. Both these packages are now part of the Bioconductor project. Comprehensive microarray studies involve integration of biological data other than microarray data. This biological data is stored in public databases, such as Ensembl. A microarray analysis however may involve integration of data that is spread over several databases, which each have their own query interfaces. These interfaces are typically not in a data analysis environment, making integration of this data with data analysis cumbersome. In Chapter 5 we address this problem taking advantage of the recent development and growth of the BioMart database management systems. We developed a software package called biomaRt, that efficiently integrates massive amounts of biological data from a wide variety of public databases, opening the way for biological data mining.

Chapter 6 describes two contributions to popular microarray analysis software. Visualization of data is a very important part of data analysis. A first contribution describes the development of new flexible and highly interactive visualization modules for a popular web-based microarray data analysis suite Expression Profiler. In a second contribution, we describe the porting of a gene prioritization tool to use with *Arabidopsis thaliana* as part of the deliverables of the CAGE project described in Chapter 4.

Chapter 7 concludes this thesis by giving a general discussion together with directions for future developments. Figure 1.2 shows a graphical overview of this thesis and the achievements.

During this Ph.D. I was involved in several data analyses of microarray experiments. Appendix A describes one such analysis of a gene expression experiment in *Xenopus laevis* to discover new candidate Wnt target genes.

Detailed information on how the RMAGEML and biomaRt software packages can be used is provided in Appendix B and C.

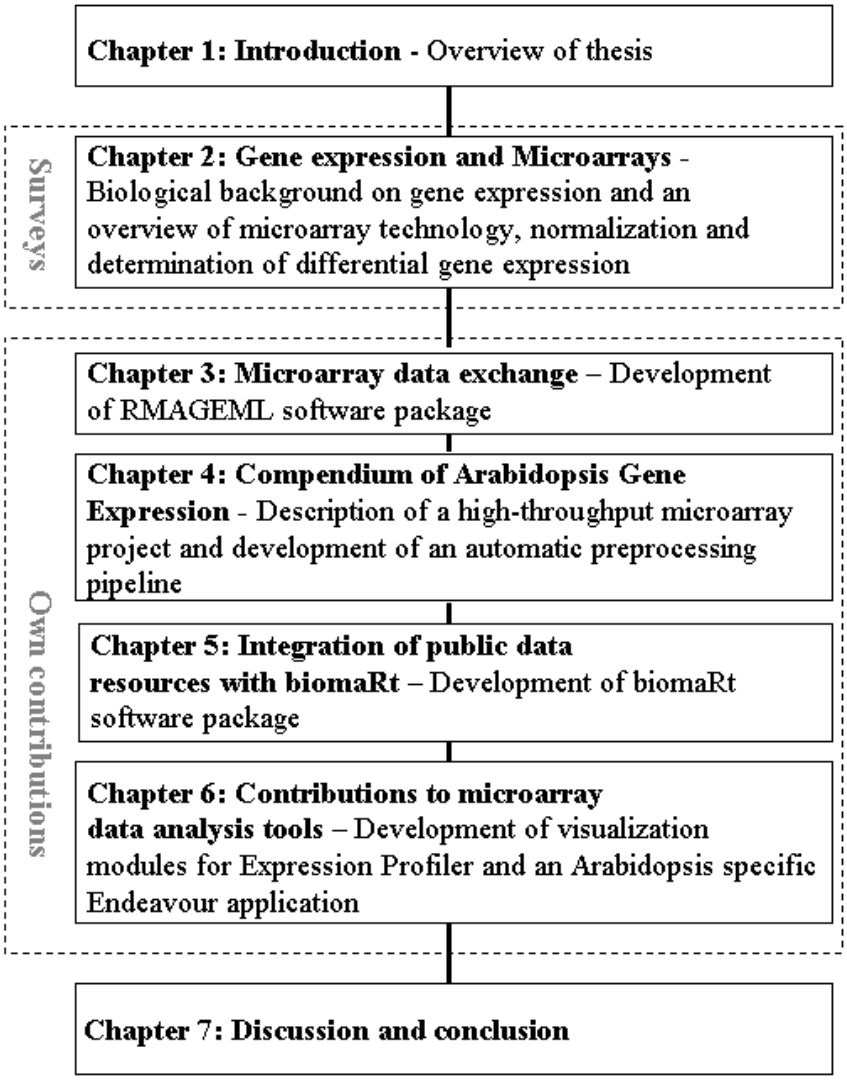


Figure 1.2: Overview of this thesis.

Chapter 2

Gene expression and microarrays

We briefly describe the fascinating discovery of the structure of DNA by Watson and Crick, which revolutionized our understanding of biology and enabled at fast pace, many new discoveries leading to the current state of our knowledge in molecular biology. From this structure we discuss gene expression, a tighly regulated process where genes that are encoded in the DNA, lead functional proteins in distinct cellular concentrations. The importance of gene expression will be highlighted as well as the multiple ways by which this process is regulated in the cell. Next we will discuss the microarray technology, a technique that molecular biologists use to study gene expression. Initially these experiments focussed on studying only a few genes at a time. The microarray era totally changed this type of research and enabled to study gene expression of thousands of genes in parallel, contributing to the data explosion in biology. Microarrays generate noisy data and one usually is interested in finding genes that have different expression levels when comparing multiple samples, for example, genes of which the expression levels increase after exposure to heat. After introducing the technology, methods for preprocessing microarray data and detection of differentially expressed genes are discussed. Later chapters will use the described methods and employ them in a high-throughput setting, where not only many genes are measured but also thousands of samples are involved.

2.1 Biology of gene expression

To explain *gene expression*, the terms DNA, RNA, and proteins need to be defined. To non-biologists, the genome can be explained as the hard disk of an organism. It contains all information to build and run the organism. The code consists of a sequence of four nucleotides of which two contain purines: *adenine* (A) and *guanine* (G) and two contain pyrimidines: *thymine* (T) and *cytosine* (C). The sequence formed by a chain of these nucleotides can be millions of bases long and is called *deoxyribonucleic acid* (DNA). In 1953 Watson and Crick [119] discovered that DNA forms a double helix where complementary base pairs form bridges between the two DNA strands (Figure 2.1). In this helical structure, the nucleotide A preferentially binds to T and the nucleotide G to C. The sugar phosphate backbone of DNA introduces a polarity on each strand, and since the two strands are complementary, one strand runs from 5' to 3', while the other runs from 3' to 5'.

The second term that needs an introduction is RNA, *ribonucleic acid*. RNA differs from DNA in several aspects. Instead of a *dexoyribose* sugar, RNA contains a *ribose* sugar and the DNA base *thymine* is replaced by *uracil* (U) in RNA. Different types of RNA exists: e.g. messenger RNA (mRNA), ribosomal (rRNA) and transfer RNA (tRNA).

Proteins are the building blocks and workhorses of an organism. They consist of a sequence of *amino acids*. There are twenty different amino acids and each one of them is encoded by a three-nucleotide sequence called a *codon*. As there is redundancy in the code, several codons can code for the same amino acid. However, not all codons code for amino acids and some serve as stop signals.

In eukaryotic organisms the genome is located in the nucleus and takes the shape of chromosomes. Going from this genomic information to proteins involves a number of distinct steps. A first step is called transcription and takes place in the nucleus. During this process, a protein complex called *RNA polymerase* uses the genetic code of the DNA as a template for the production of a primary transcript: heterogeneous nuclear RNA (hnRNA). The hnRNA consists of intronic and exonic sequences and undergoes RNA splicing, a process that removes the intronic sequences. Additional processing steps add a 5' cap structure and a 3' poly-Adenosine (poly-A) tail to the RNA sequence, leading to the formation of messenger RNA (mRNA). This mRNA then migrates

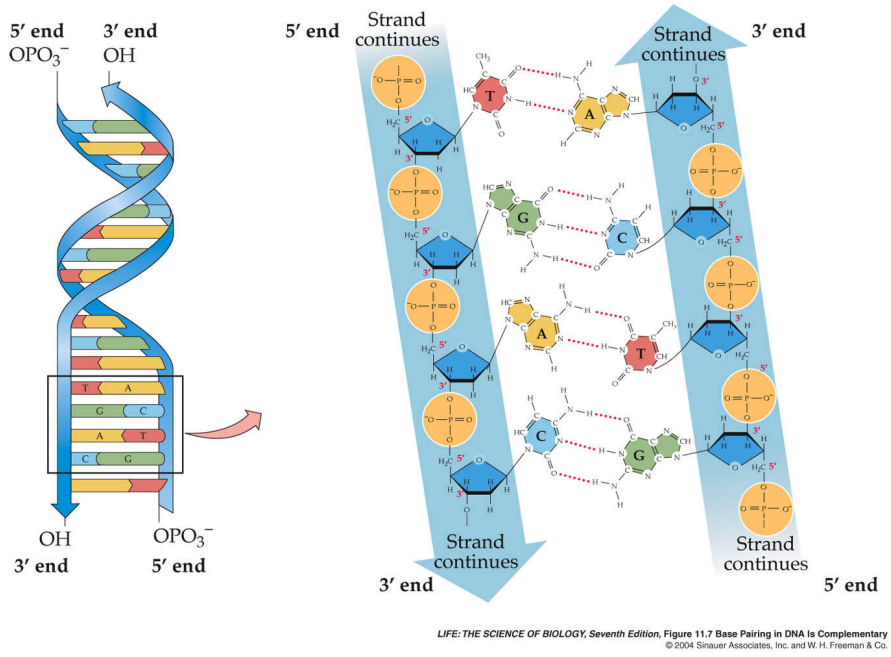


Figure 2.1: The structure of DNA. In 1953 Watson and Crick determined the structure of DNA and found that complementary base pairing is holding two strands together, forming a DNA double helix.

out of the nucleus into the cytoplasm where it can bind to ribosomes (i.e., organelles composed of RNA and ribosomal proteins). Ribosomes can be either free or bound to a membrane and are responsible for a process called translation. During translation, the codons encoded in the mRNA are translated into a chain of amino acids forming the proteins. This path from DNA to proteins via an mRNA intermediate is also known as the *central dogma of biology*. The process of production of mRNA and subsequent translation into proteins is also referred to as *gene expression*.

All cells in an organism contain the same genomic information. However the number of different cell types is large. A haematopoietic stem cell can for example differentiate into a variety of mature cells as B-cells, T-cells, erythrocytes, natural killer cells, megakaryocytes, granulocytes and monocytes. These different cell types arise because of different expression status of the genes in each cell type. During differentiation of for example haemapoietic stem cells, genes associated with self-renewal of the stem cells are downregulated and a selective expression of lineage-specific genes takes place [34]. Gene expression is regulated at multiple levels, these are discussed in the next sections.

2.1.1 Regulatory proteins and motifs

As described above, RNA polymerases are responsible for the transcription of DNA into mRNA. Recruitment and activation of the RNA polymerase to the transcription start region of genes is tightly regulated by regulatory proteins called *transcription factors*. Transcription factors are crucial for determining the expression status of a gene. They can activate or repress gene expression by binding to specific sequence motifs. In eukaryotic organisms, each gene contains a stretch of these regulatory motifs upstream but close to the transcription start site. This region is called the *promoter* and the regulatory sequences it contains are also called *cis-regulatory* elements or motifs. Transcription factors in their turn sometimes need additional co-factors (proteins forming a complex with the transcription factor without being directly bound to the DNA) in order to be active. A promoter can contain several motifs binding different transcription factors and these motifs can be clustered in groups, also referred to as *cis-regulatory modules*. These multiple regulatory sites lead to complex combinatorial regulation [76] of gene transcription, also known as *combinatorial control*. Figure 2.2 depicts combinatorial control in the promoter region of the β-globin gene.

Beside the promoter region there are also more distant regulatory motifs that can regulate the expression of a gene. These motifs, known as enhancers and

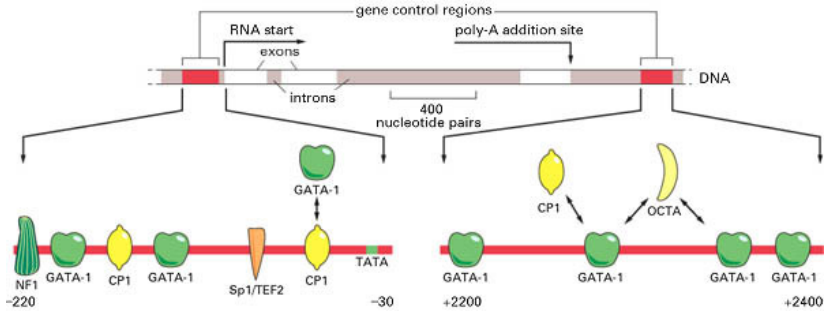


Figure 2.2: Regulatory motifs surrounding the β-globin gene. Different transcription factors can bind to the promoter site of the β-globin gene and regulate its expression. Figure from Alberts *et al*. (1994) [3]

silencers, can be thousands of nucleotide pairs away from the promoter. Regulatory proteins that bind to these distant motifs bend the DNA between them and the promoter and make contact with the RNA polymerase at the promoter to either activate or repress transcription. Another class of regulatory elements are insulators. They are nucleoprotein structures that form chromatin boundary elements and can also block the effect of enhancers [10]. This way they help assuring that an enhancer will only interact with an appropriate gene target.

2.1.2 Genomic DNA variation and DNA methylation

Epigenetics is a term used to refer to heritable changes in gene expression that do not result from change in nucleotide sequence. Methylation of promoter sequences can reduce (silence) or even block transcription of the corresponding gene [13]. Methylation is catalyzed by DNA methyltransferases (DNMTs) that methylate cytosine residues within cytosine-guanine dinucleotides (CpG) to form 5-methylcytosine (m5C). These methyl groups point toward the major groove of DNA and as such inhibit transcription. The CpG dinucleotides occur less often than expected throughout the human genome as m5C can undergo spontaneous deamination into thymine and thereby changes the sequence. CpG dinucleotides instead concentrate on small stretches of DNA sequence known as CpG islands. These islands are typically found near promoter regions of genes [47]. DNA methylation in normal cells helps in maintaining unexpressed and non-coding regions of the genome transcriptionally silenced. In general, sites near promoter regions are unmethylated, exceptions are regions where these sites are methylated to ensure transcriptional inactivation, (e.g., near genes on the inactivated X chromosome of females).

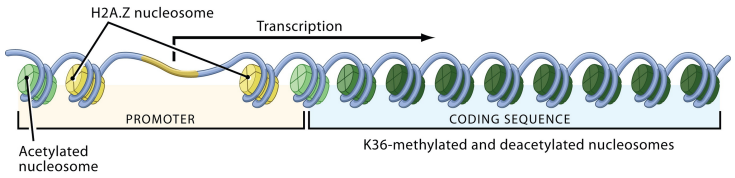


Figure 2.3: Chromatin-mediated control of transcription initiation. For the promoter site to be accessible for transcription factors and the RNA polymerase, the chromatin needs to open up. Figure from [79]

2.1.3 Histone modification

DNA within living eukaryotic cells exists in the form of chromatin. The basic building block of chromatin is the nucleosome, which consists of an octamer of four core histone proteins (H2A, H2B, H3 and H4) around which 147bp of DNA is wrapped [71]. Beside these four core histones there is also a fifth histone H1, which has a linker function but is no part of the nucleosome. The chromatin fibers undergo several levels of folding resulting in increasing degrees of condensation and the formation of chromosomes. During transcription initiation, interactions between DNA and histones need to be disrupted for RNA polymerase to access the template strand [84] (see Figure 2.3). At the same time downstream nucleosome occupancy and stability must be maintained to prevent transcriptional initiation at other inappropriate sites [66]. During elongation, nucleosomes are disassembled in front of the polymerase to allow passage and are rapidly reassembled as soon as the polymerase passes [98, 112]. This requirement opens possibilities of transcription regulation at the chromatin level.

A striking property of histones is that they are subject to a variety of covalent modifications such as ubiquitination, acetylation, methylation and phosphorylation. The modification status of histones influences the folding state. Because of this influence, methylation and acetylation of histones play an active role in regulating gene expression. For example it is found that both hyper- and hypoacetylation of histones on individual lysine residues generates groups of biologically related genes. Interestingly the genes within these groups are significantly coexpressed, mediate similar physiological processes, share cis-regulatory DNA motifs, and are enriched for binding of specific transcription factors [73].

2.1.4 DNA amplification and deletion

Genetic defects can influence gene expression. In tumorigenic cells, certain DNA regions can be deleted or copied multiple times and inserted into other parts of the genome in a process called genomic DNA deletion and amplification. These genomic changes can contribute to a significant change in gene expression of the genes that are encoded in these regions, which sometimes can be linked to their malignant phenotype [48].

2.1.5 Translational control

The stability of mRNA sequences can differ by over a 100 fold, some mRNAs have a half-life of 15 minutes others can have a half-life of up to three days. This tremendous difference in stability can have a big impact on gene expression. Similarly as transcription is regulated by transcription factors, translational control is achieved by binding of regulatory factors to the mRNA. The 3' UTR (untranslated region) of mRNA often contains several regulatory elements that bind translational regulators that regulate spatial and temporal translation of the mRNA. Translational regulators can be highly conserved and control many different mRNAs [72].

Only recently, RNA molecules have been discovered as one of the more abundant classes of gene regulatory molecules in multicellular organisms, called miRNAs [9] . These miRNAs are short, typically 22 nucleotides long, noncoding RNA sequences that are complementary to part of the mRNA sequence of a specific gene or a sequence shared by many genes. miRNAs are expressed as hairpins and their expression results in rapid degradation of their target mRNAs and a corresponding reduction of gene expression.
Once an miRNA duplex is cleaved from the pri-mRNA hairpin, the pathway leading to degradation of target mRNAs is indistinguishable from the RNA silencing pathways known as RNA interference (RNAi) in animals, post-transcriptional gene silencing (PTGS) in plants, and quelling in fungi. The double stranded RNA species become incorporated as single stranded RNAs into a ribonucleoprotein complex, known as RNA-induced silencer complex (RISC). RISC identifies the target sequence and it's endonuclease cleaves the target mRNA [45].
Most miRNAs are encoded in the genome at locations distant from annotated genes, which implies that they have their own transcription regulatory system. However a sizable fraction of miRNAs are encoded in the introns of genes, preferentially in the same direction as the gene, suggesting that these miRNA

do not have their own promoters but are processed from the introns [74]. Another fraction of miRNAs tends to appear as clusters in the genome and show co-expression, implying transcription as a multi-cistronic primary transcript. These miRNAs within a genomic cluster are often functionally related.
Since their discovery in *C. elegans* [75], where they were shown to play a role in development, miRNAs have been found to be important players in a wide variety of processes, such as neuronal patterning and life span in nematodes [64, 15], flower development in plants [22], skeletal muscle proliferation and differentiation [21], and potentially memory and learning [97].

2.1.6 Protein degradation

The last level of gene expression regulation is at the protein level. Proteasomes are protein complexes that consist of a central cylinder formed from multiple distinct proteases. The combined action of these proteases degrade proteins into short peptides. Proteasomes specifically degrade proteins that are ubiquitinated. Ubiquitination of proteins therefore tags proteins for degradation by proteasome complexes. These proteolytic mechanisms ensure to dispose the cell of misfolded or damaged proteins and also provide a means to quickly change the concentration of a protein in the cell [3]. The presence of proteins however does not directly imply that they are active, the activity of proteins can be regulated by phosphorylation and dephosphorylation of active sites.

2.1.7 Importance of gene expression

To highlight the impact of gene expression and the importance of studying this phenomenon, a selected set of experiments is described below showing the involvement of gene expression in response to environmental changes, development, and disease.
Individual cells react to their environment by changing gene expression, this has been studied extensively in yeast. In one of the first experiments to investigate this with microarrays, Gash *et al.* exposed yeast cells to a diverse set of environmental transitions and measured gene expression changes over time [38]. The cells changed the expression of many genes in response to temperature shocks, hydrogen peroxide, the superoxide-generating drug menadione, the sulfhydryl-oxidizing agent diamide, the disulfide-reducing agent dithiothreitol, hyper- and hypo-osmotic shock, amino acid starvation, nitrogen source depletion, and progression into stationary phase. A large set of genes showed a similar drastic response to almost all of these environmental changes. In addition to this, unique expression changes were recorded for specific conditions.

In multicellular organisms, every cell has the same genome, however these organisms consist of many different cell types. Differential gene expression is the driving force in the generation of these different cell types. Gene expression regulation is important during animal development. A classical example is *Drosophila* anteroposterior axis specification and segmentation of the *Drosophila* embryo. In these embryos mRNA coding for the regulatory protein *bicoid*, is located at the anterior end, causing a gradient of the protein over the embryo with a high concentration at the anterior end and a low concentration at the posterior end [37]. Similarly the mRNA of *nanos* is located at the posterior end creating a *nanos* protein gradient with high *nanos* concentrations at the posterior and low *nanos* concentrations at the anterior end [40]. *Bicoid* mRNA translation is inhibited by *nanos*. A third mRNA, coding for the *hunchback* protein is evenly distributed over the embryo but its translation is enhanced by *bicoid* and inhibited by *nanos*. This creates a gradient of *hunchback* from the anterior to the middle of the embryo. In it's turn, high concentrations of *hunchback* protein inhibit the transcription of the *Kruppel* gene, lower concentrations activate *Kruppel* expression and even lower concentrations do not activate. This produces a band of *Kruppel* expression [54]. These expression patterns set boundaries that lead to stripes of specific gene expression (see Figure 2.4), which are the forerunners of segmentation of the *Drosophila* embryo and give rise to the body segments of the adult fly.

Many human diseases can be linked or classified by changing patterns in gene expression. One of the first studies demonstrating the use of microarrays to classify tumor samples was published by Perou *et al.* in 2000 [88]. By using cDNA microarrays covering 8102 human genes, the authors showed that breast tumor samples taken from individuals at different times, have in most cases, more similar expression patterns when coming from the same individual than when compared to expression patterns of samples from different individuals. Furthermore a hierarchical clustering of the obtained expression patterns could be used to classify the different tumors according to their subtype, see Figure 2.5.

Breast cancer patients with the same stage of disease can have different responses to treatment and overall outcome. Chemotherapy or hormone treatment can reduce the risk of metastasis, but 70-80% of the patients receiving this treatment would have survived without it. In 2002, van't Veer *et al.* [116] performed a DNA microarray analysis on 117 primary breast tumors and identified an expression profile signature that could be used to predict poor prognosis, thus patients who should receive treatment. Additionally the signature

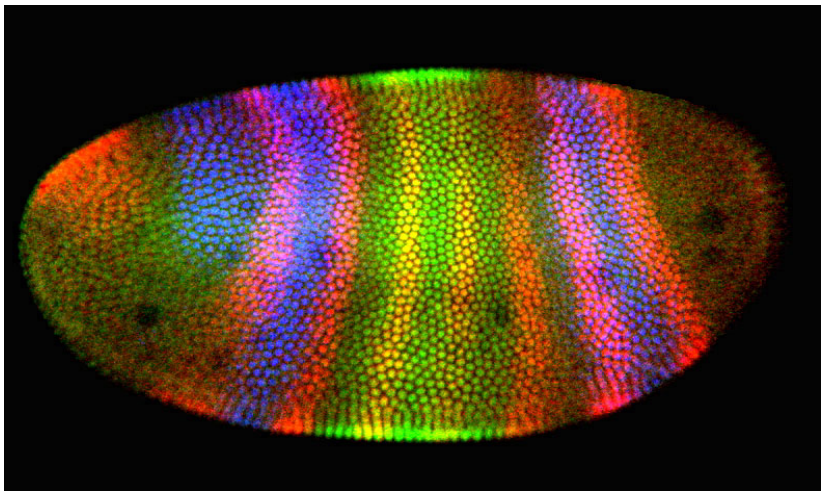


Figure 2.4: Gene expression in a developing *Drosophila* embryo is regulated so that a stripe pattern of different expressed transcription factors is created. This pattern is the basis of forming the segments, characteristic for the developed embryo and adult fly. Colored in green is the stripe corresponding to the the region where the *Kruppel* transcription factor is expressed. Figure from http://www.bio.davidson.edu/courses/Molbio/fly/Flymontage.html.

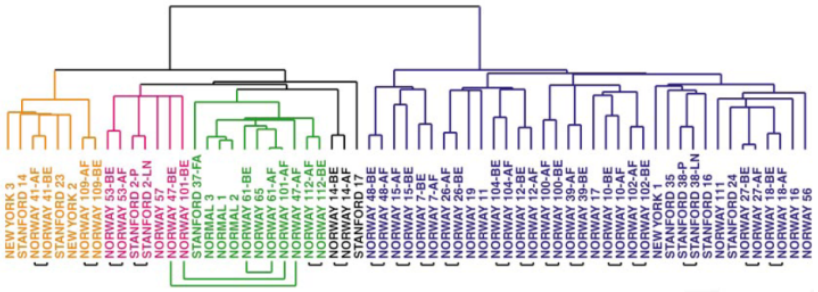


Figure 2.5: One of the first studies showing that gene expression profiles of tumor samples taken from one patient at different times are in most cases more similar to each other then when compared to a sample from another patient (black lines under sample names) and that clustering of tumor samples by their expression profile reveals different tumor subtypes that cluster together. The branches of each identified subtype were plotted with a different color: basal like, orange; Erb-B2+, pink; normal breast like, green; luminal epithelial/ER+, dark blue. Figure from Perou *et al.* (2000) [88].

could also identify patients that were BRCA1 mutation carriers. In some cancer types, early prediction of metastasis can be crucial for applying the appropriate treatment. A last clinical example is a study by Roepman *et al.* The authors show that DNA microarray gene-expression profiling can be used to detect lymph node metastases for primary head and neck squamous cell carcinomas that arise in the oral cavity and oropharynx [91]. Their predictor was build by using a 82 sample tumor training set and uses the expression status of 102 genes. They also show that the predictor based on gene expression outperforms the current clinical diagnosis methods used for this disease.
These examples illustrate that gene expression plays a pivotal role in life and maintaining a cells fate and in addition that expression profiles can have a great diagnostic value.

2.2 Microarray technology

As discussed in the previous section, every nucleic acid strand has the capacity to recognize complementary sequences through base-pairing. This recognition process by base-pairing is also called *hybridization* and can be highly parallel. Thus a complex mixture containing several thousands of different nucleic acid strands can be interrogated simultaneously. This principle is used in microarray technology.

2.2.1 Introduction

Microarrays consist of nucleotide sequences, referred to as *probes*, attached to a solid surface such as a silicon wafer. The nucleotide sequences are such that they are complementary to sequences whose abundance we want to determine. Recent advances in this technology have made a variety of microarrays platforms to become available. Examples are microarrays to measure gene expression, to detect genomic instabilities (array-CGH), to measure transcription factor binding (ChIP-chip), to analyze Single Nucleotide Polymorphisms (SNP array), and arrays to do sequence analysis. As the main focus of this thesis is on expression microarrays, only these will be discussed in detail, although many of the principles also apply to the other microarray applications. Expression microarrays measure gene expression at the transcript level so nothing about the expression regulation at the post-transcriptional level can be determined.

2.2.2 Two-color spotted microarrays

Spotted microarrays are arrays where the sequences of the probes have been spotted on the array. Two types of spotted microarrays can be distinguished: cDNA (copy DNA) arrays and long oligonucleotide arrays. The difference between these two types is the length of the probe and mode of probe creation. The probes of cDNA arrays [95] are made from cDNA clones containing a cDNA corresponding to specific a gene. The cloned cDNA fragments are created by a process called *RT-PCR*, were the mRNAs are reverse transcribed from the 3' end of the mRNA into cDNA. These cDNAs usually have a length between 0.6 and 2.4kb [31]. In long oligonucleotide arrays, the probes are synthetically produced and are usually 40-90 bp in length [65, 8]. These amplified cDNAs and oligonucleotides are then spotted on a glass slide to construct the microarray. The initial spotting robots relied on contact printing, since then many variations such as non-contact piezo-electric or ink-jet printing devices [52] have been developed. The number of spots spotted on one slide can vary from a few thousands up to about 40,000, depending on the experimental goals. Experiments with spotted microarrays usually involve mRNA from two samples to be compared on the same slide. Typically the mRNA from one sample is converted cDNA labeled with the Cy3 fluorescent dye and the mRNA of the second sample is converted to Cy5 labeled cDNA. The two labeled products are then hybridized together on the same slide. After hybridization and stringent washing, the slides are scanned with a laser using two different wavelengths to obtain the hybridization intensities for each dye. Image analysis will produce different measured quantities, such as the mean foreground and background values for each probe on the array. Figure 2.6 illustrates the different steps of a spotted microarray experiment as described above.

The spotted microarrays are popular as they can be created in a molecular biology lab in a fairly easy way. This gives them the advantage of a low price and the ability to create custom arrays according to the needs of the lab. A disadvantage however is that the data produced usually can be only interpreted as ratios and that no absolute expression values are obtained.

2.2.3 High-density oligonucleotide arrays

In contrast to spotted arrays, the probe sequences on high-density oligonucleotide arrays are synthesized on the array. This way many more probes can be fitted on an array. The first commercial high-density oligonucleotide

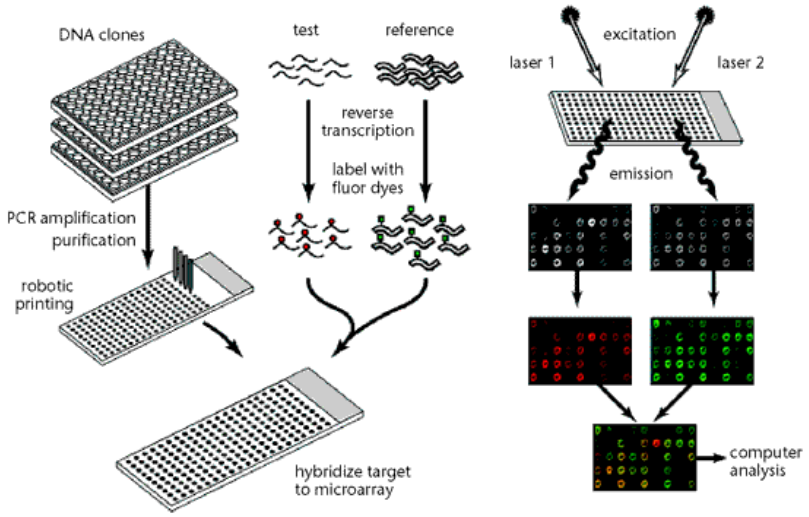


Figure 2.6: Different steps in a two-color cDNA experiment. DNA clones provide the sequences that, after amplification by PCR, are spotted on a glass slide. Two samples are hybridized together on the array and are both labeled with a different dye. After hybridization of the labeled samples to the array, a laser excites the dyes at their own excitation wavelength. This way fluorescence intensity measurements are obtained of each dye for each spot on the array. Figure derived from [31].

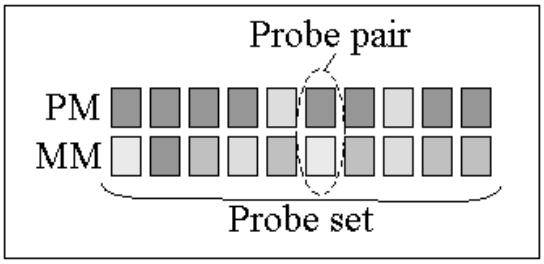


Figure 2.7: Affymetrix GeneChip array design is characterized by a perfect match (PM) and a paired mismatch (MM) probe. A probe set typically consists of 11 to 20 PM and MM probes and measures the expression levels of a target sequence

platform was made by Affymetrix. Recently a new such platform has been produced by Nimblegen^{TM}. Both platforms are discussed in the next sections.

Affymetrix GeneChips

Affymetrix GeneChip® are high density synthetic oligonucleotide arrays [81, 80]. The probes, which typically are 25bp in length, are synthesized on the chip in a photolithographic process. Photochemically removable protecting groups protect new nucleotides to be included. Only when light is directed through a photolithographic mask to specific areas on the chip, local photodeprotection occurs and these deprotected strands will incorporate the extra nucleotide. Different sequences can be grown in parallel by using a series of different masks [80].

Affymetrix probe sets consist of a series of probes, which individually measure a perfect match (PM) or a paired mismatch (MM) signal. The sequence of the PM and paired MM probe is the same except for a nucleotide change in the middle of the probe. The PM probes aim to measure the effective gene expression status and the MM probes are there to estimate the amount of cross-hybridization and thus background noise. For each probe set there are usually 11 to 20 PM and MM probes. An example of the Affymetrix GeneChip design is shown in Figure 2.7.

The photolithographic procedure allows the construction of arrays with an extremely high number of probes. Current chip densities are as such that one chip can contain as many as 5.3 million different probes (Exon GeneChip®).

NimbleGen^{TM}

The NimbleGen^{TM} platform is the most recently developed microarray platform. The production of these arrays is highly similar to that of the Affymetrix platforms. The sequences are again synthesized on the array using photolithograpic methods but using a maskless technology [103]. Instead of being built on a silicon wafer, the wafer of NimbleGen^{TM} arrays is a digital micromirror device consisting of 786,000 micromirrors that can be individually addressed. Protected nucleotides are facing away from a light source, by turning a micromirror toward the light however, photodeprotection occurs on that mirror and a nucleotide can be added. Figure 2.8 shows the NimbleGen^{TM} array production.

The advantage that NimbleGen^{TM} arrays offers over Affymetrix GeneChips is that customized arrays can be built in a few days as no real mask is required to photodeprotect the growing sequences.

2.3 Preprocessing of microarray data

After samples are hybridized to the array, the arrays are scanned and image analysis software is used to convert the pixel intensities of the scanned images into probe-level data. The data however consist of noisy signal measurements. The real gene expression measure is masked by different sources of noise, such as labeling efficiency, print-tip effects, between slide variation and other factors [123, 125]. Normalization of microarray data aims to correct the raw intensity data for these systematic biases.
Normalization methods of two-color and high-density oligonucleotide arrays differ. Below we discuss preprocessing of data coming from both platforms.

2.3.1 Quality assessment

To assess the quality of the microarray experiments and need for normalization, different methods exist. Graphical representations of the array data help to quickly identify bad arrays and need for normalization. Hybridization problems can be viewed by plotting images of the foreground and background intensities for each channel for spotted arrays and of the PM and MM values of

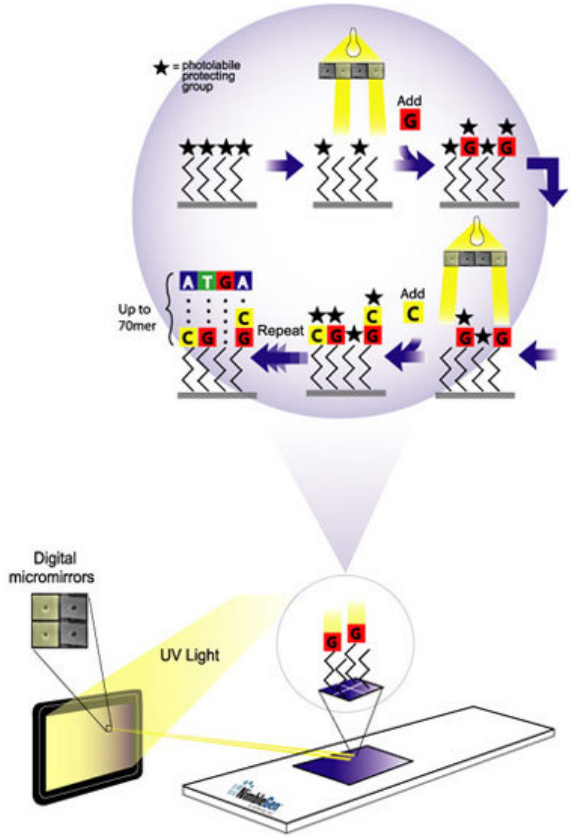


Figure 2.8: Synthesis of a NimbleGen^{TM} array. By turning the digital micromirrors to the light source, probe sequences are photodeprotected and can add a new nucleotide to their sequence. This way hundreds of thousands of sequences can be synthesized in parallel on the array. Figure from NimbleGen^{TM} web site.

Affymetrix GeneChips. Strikes or other patterns that show up indicate faults and contamination during hybridization. Depending on the severity of the pattern, this can be corrected during normalization or not. For cDNA arrays, the effect of failing or sub-optimal print tips can be visualized by plotting a boxplot of the intensities per print tip. A general assessment of hybridization quality is to identify how many spots are above the background. If this number is too low then the hybridization might have to be redone. Between-array differences can be visualized by plotting boxplots of the raw intensities grouped per array. Failing arrays will show up as outliers and the need for between-array normalization will be visualized by the boxplot pattern.

A second set of plots useful for quality assessment of different arrays is a scatter plot of all against all or a correlation heatmap. Usually one has at least one repeat of each hybridization and we can expect the correlation between these hybridizations to be high and higher than compared with a different hybridization. When the correlation between repeats is low this usually means a bad hybridization or in some cases a mix-up of samples.

Hybridizations failing these quality assessments should be discarded and replaced by a new hybridization if possible or get a lower weight in the subsequent analysis.

2.3.2 Normalization of two-color microarray data

Two-color microarray experiments are characterized by probes spotted on a microarray slide and the use of two labeled samples that are hybridized simultaneously on the same array and labeled with a different dye, usually Cy3 (green) and Cy5 (red). Image analysis software gives for each channel background and foreground intensity measurements. One can choose to subtract the background measurements from the foreground or not. Print-tip loess normalization is a robust local regression based normalization that accounts for intensity and spatial dependence in dye biases [123] and is frequently used to normalize cDNA microarray experiments. The loess fit adjusts each feature with a different normalization value depending on its overall intensity. By generating a loess fit for each print-tip separately we correct for print-tip effects. This type of normalization can be used under the assumption that only a small subset of genes is differentially expressed between the two samples. If this is assumption does not hold, one can rely on spiked-in RNA to calculate the loess fit using the spike signals only and then use that fit to adjust the data [115]. Other frequently used methods to normalize cDNA microarrays are qspline [120] and VSN [51] normalization (see below). A between slide

normalization, which corrects for a difference in scale can be applied if the variance of the ratios differs a lot between the different slides. A boxplot of the ratios grouped per slide will reveal the necessity of such a between slide normalization. ANOVA [69] is an alternative to the types of normalization methods described above and aims to estimate the size of different effects including dye, gene, array, sample and together with these effects one estimates the sample $\times$ gene interaction effect, which is the true differential expression we are interested in. The model Kerr *et al.* propose is

$$\log(y_{ijkg}) = \mu + A_i + D_j + V_k + G_g + (AG)_{ig} + (VG)_{kg} + \epsilon_{ijkg}, \quad (2.1)$$

where y_{ijkg} denotes the measurement from the ith array, j^{th} dye, k^{th} variety (sample) and g^{th} gene; μ is the overall average signal, A_i represents the effect of the ith array, D_j is the effect of the jth dye, V_k the effect of the kth variety (sample), G_g represents the effect of the gth gene, $(AG)_{ig}$ represents the effect of the combination of array i and gene g and $(VG)_{kg}$ is the interaction between the kth variety and the gth gene. The error terms ϵ_{ijkg} are assumed to be independent and identically distributed with mean zero.

2.3.3 Normalization of Affymetrix GeneChips

We can think of Affymetrix GeneChip preprocessing as a three-step procedure, although steps are sometimes combined depending on the algorithm. These three steps are (1) estimation of the background signal, (2) possible background correction normalization, and (3) summarizing the separate normalized probe measurements into a single value per probe set [63].

Methods such as *MAS5.0*, *MBEI*, *GCRMA*, *RMA* and *VSN* are used frequently as normalization methods and depending on the method used, the final expression values will differ [16, 24, 101]. Up to date none of these methods is set as the standard best method and conflicting reports have been published about the performance when comparing the different methods. In a recent study by Irizarry *et al.* [63], 31 different preprocessing algorithms were compared. The authors found that background correction is the main factor that explains the differences between the performance of the different methods. In their study the found that GCRMA was amongst the better performers as it appears to find the best balance between accuracy and precision while most other methods seem to be developed to perfect the accuracy only [63]. However the authors do not wish to make a judgment and declare one single method as the

'best'. Below is an overview of the most popular normalization methods for Affymetrix GeneChips.

MAS 5.0

MAS 4.0 was the original normalization method implemented by Affymetrix. The expression index was defined by the average of the PM-MM differences for all probe pairs in a probe set, which was called the *average difference*. However it was noticed that there was a dramatic variation in the PM-MM differences of the 20 probes within a probe set. Many of the MM values were also larger than the PM values which resulted in negative adjusted intensity values. These facts led to the implementation of a new procedure: *MAS5.0*, which makes use of the MM measurements to compute an *Ideal Mismatch (IM)* [2]. The biweight specific background (SB) for probe pair j in probe set i can be calculated as

$$SB_i = T_{bi}(\log_2(PM_{i,j}) - \log_2(MM_{i,j}) : j = 1, ..., n_i), \tag{2.2}$$

where T_{bi} is a *one-step Tukey's biweight*. This one-step Tukey's biweight is a robust estimation of the mean of a sample $x = x_1, ..., x_n$ and is thus not affected by outliers. Denote the median of $x_1, ..., x_n$ by M, and the median of the absolute distances from M by S. To obtain the one-step Tukey's biweight estimate, first a uniform measure u_i of the distance of $x_i, i = 1, ..., n$, from the center is computed by

$$u_i = \frac{x_i - M}{c.S + \epsilon}. \tag{2.3}$$

In this formula, c is a tuning constant, and $\epsilon > 0$ is a small value to ensure u_i is always defined. In MAS5.0, $c = 5$ and $\epsilon = 0.0001$ are used by default.
Next, for each datapoint i, a weight is computed by

$$w(u_i) = \begin{cases} \left(1 - u_i^2\right)^2, \text{if} |u_i| \leq 1 \\ 0, \text{if} |u_i| > 1 \end{cases} \tag{2.4}$$

The one step Tukey's biweight is now defined by

$$T_{bi}(x) = \frac{\sum_i w(u_i) x_i}{\sum_i w(u_i)} \tag{2.5}$$

MAS 5.0 uses the biweight specific background, computed by Equation 2.2, to calculate the Ideal Mismatch values (IM). If SB_i is large then the values of

the probes are trusted and SB_i is used to calculated the IM for a probe pair. If however SB_i is small ($SB_i \leq$ contrast τ) more of the PM value is used to construct the IM. The following three cases determine the value of IM, where scale τ is the cutoff describing the variability of the probe pairs in the probe set:

$$IM_{i,j} = \begin{cases} MM_{i,j}, & MM_{i,j} < PM_{i,j} \\ \frac{PM_{i,j}}{2^{SB_i}}, & MM_{i,j} \geq PM_{i,j} \text{ and } SB_i > \text{contrast } \tau \\ \dfrac{PM_{i,j}}{2^{\left(\frac{\text{contrast } \tau}{1+\left(\frac{\text{contrast } \tau - SB_i}{\text{scale } \tau}\right)}\right)}}, & MM_{i,j} \geq PM_{i,j} \text{ and } SB_i \leq \text{contrast } \tau. \end{cases}$$

The PM values are then corrected with these Ideal Mismatch values instead of using the MM values. After background correction, MAS5.0 uses a one step Tukey Biweight algorithm, a method to determine an average robust to outliers, as summarization method [2].

MBEI

Li and Wong were the first to implement a model-based background correction method referred to as the Model Based Expression Index (MBEI) (also called dChip algorithm) [77].

The individual probe responses can be modeled by

$$MM_{ij} = \nu_j + \theta_i \alpha_j + \varepsilon \tag{2.6}$$

and

$$PM_{ij} = \nu_j + \theta_i \alpha_j + \theta_i \phi_j + \varepsilon, \tag{2.7}$$

where PM_{ij} and MM_{ij} denote the PM and MM intensity values for the *i*th array and the *j*th probe pair for a particular gene. ν_j is the baseline response of the *j*th probe pair due to non-specific hybridization, α_j is the rate of increase of the MM response of the *j*th probe pair, ϕ_j is the additional rate of increase in the corresponding PM response and ϵ is random error. This model implies the following simple model for the PM-MM differences:

$$y_{ij} = PM_{ij} - MM_{ij} = \theta_i \phi_j + \varepsilon_{ij}. \tag{2.8}$$

RMA

The Robust Multi-array Average (RMA) [62] is a second model-based procedure and unlike MBEI and MAS does not use the MM values. The authors found that the default noise adjustment, PM-MM, results in expression estimates with a big variance. To avoid this, they estimate the background by convoluting the signal and noise distributions from the PM values in which the observed PM intensity is modeled as a sum of normal background and exponential signal:

$$O = S + N. \tag{2.9}$$

Thus the corrected signal is given by

$$E(S|O=o) = a + b\frac{\phi\left(\frac{a}{b}\right) - \phi\left(\frac{o-a}{b}\right)}{\Phi\left(\frac{a}{b}\right) - \Phi\left(\frac{o-a}{b}\right) - 1}, \tag{2.10}$$

and $a = o - \mu - \sigma^2\alpha, b = \sigma$

After background correction, RMA performs a quantile normalization. Quantile normalization aims to make the distribution of probe intensities for each array the same for a set of arrays and can be applied in practice by the following algorithm [16]:

1. Given n arrays of length p, form X of dimension $p \times n$ where each array is a column
2. Sort each column of X to give X_{sort}
3. Take the means across rows of X_{sort} and assign this mean to each element in the row to get X'_{sort}
4. Get $X_{\text{normalized}}$ by rearranging each column of X'_{sort} to have the same ordering as the original X

After quantile normalization, RMA uses median polish as summarization method.

By not using the MM values, Irrizary *et al.* sacrify some of the accuracy but attain large gains in precision. This technique was used to normalize the AtGenExpress data and was put in the EBI data warehouse [85].

GCRMA

Wu *et al.* found that the loss in accuracy of the RMA method is due to inappropriate adjustment for the presence of non-specific hybridization and proposed a modified RMA procedure, GCRMA [122], which improves the background adjustment step by including probe sequence information. Wu *et al.* suggest the following model

$$PM = O_{PM} + N_{PM} + S \tag{2.11}$$

and

$$MM = O_{MM} + N_{MM} + \Phi S, \tag{2.12}$$

where O represents the optical noise, N represents the Non-Specific Binding (NSB) noise and S is a quantity proportional to the RNA expression. The parameter $0 < \Phi < 1$ accounts for the fact that for some probe-pairs the MM detects signal. This normalization method was used in the analysis of Wnt targets described in Appendix A.

VSN

The last method we want to discuss is the Variance Stabilizing Normalization (VSN) method which combines background correction and normalization and can be used for two-color experiments as well [51]. The transformation $h(x)$ that is applied in VSN is given by

$$h(x) = \text{arcsinh}(a + bx). \tag{2.13}$$

The parameters a and b are estimated from the microarray data. For large intensities this transformation approximates the logarithmic transformation and Δh to the log-ratio. In summary, VSN returns normalized data, which has an approximate constant variance independent of the spot intensity.

2.4 Finding differentially expressed genes

One of the main goals of microarrays is to determine which genes are differentially expressed when comparing their expression according to the phenotype or experimental condition in sets of samples. Usually the intensities are transformed into logarithmic scale to analyze differential expression. One of the difficulties is that there are many more genes than there are samples. The

number of genes is often on the scale of tens of thousands, while the number of samples might be one hundred or even less. In this section, different methods to find differentially expressed genes will be discussed.

2.4.1 Fold change

The simplest method to define a gene as differentially expressed or not is to use a threshold on the fold-change M defined as

$$M = \log_2(\text{intensity}_{\text{sample}_1}) - \log_2(\text{intensity}_{\text{sample}_2}). \tag{2.14}$$

This criterion was used in early microarray analyses [27, 95, 94] and is now only applied when there are none or only a few replicates. In the case of no replicates, a more statistical approach is not possible and thus no significance can be assigned to the observed differential expression in the presence of biological and experimental variation.

2.4.2 t-test and ANOVA

If one has a sufficient number of replicates (e.g. three or four good quality replicates), a t-test comparison or ANOVA can be used. However these parametric tests are often used even when the number of replicates is too low to fulfill the underlying assumptions of normality.

2.4.3 Limma

Many microarray experiments involve only a few replicates per sample, which makes it difficult to estimate the gene-specific variances used for example in the t-test. To cope with this, different methods have been developed that exploit the variance provided by the data of all genes. The bioconductor package *limma* implements an Empirical Bayes approach that uses the global variance estimate s_0^2 on the basis of the variances of all genes. The resulting test statistic is a moderated t-statistic, that instead of using single-gene estimated variances s_g^2, uses a weighted average of s_g^2 and s_0^2. Under certain distributional assumptions, this test statistic is shown to follow a t-distribution under the null hypothesis [104].

The weighted average of s_g^2 and s_0^2 is calculated by

$$\tilde{s}_g^2 = \frac{d_g s_g^2 + d_0 s_0^2}{d_0 + d_g}, \tag{2.15}$$

where d_g and d_0 are weights. The moderated t-statistic can then be calculated by

$$t = \frac{\bar{M}_g}{\sqrt{\tilde{s}_g^2/n}}, \tag{2.16}$$

where $\bar{M}_g$ is the mean of the M values of a particular gene g and n the number of replicate arrays.

2.4.4 SAM

Similarly to *limma*, Significance Analysis of Microarrays (SAM) [114] also calculates a modified t-statistic:

$$t = \frac{\bar{M}_g}{(a + s_g)/\sqrt{n}}, \tag{2.17}$$

where $\bar{M}_g$ is the mean of the M values of a particular gene g, a a positive constant and n the number of replicate arrays.

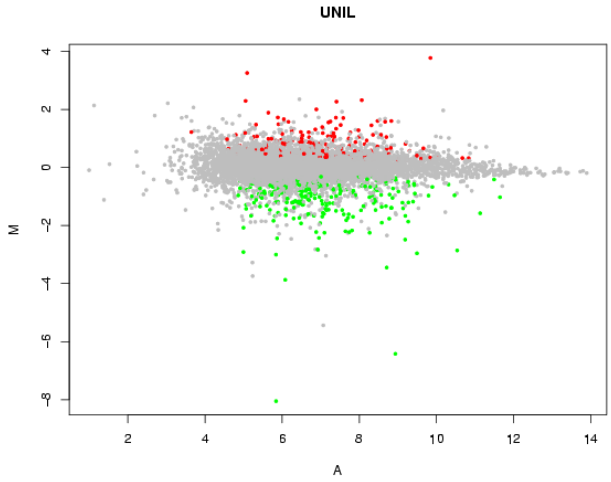
2.4.5 Multiple testing

The statistical methods for differential expression conduct a statistical test for each gene separately. This has the drawback that a large number of hypothesis tests are performed, possibly leading to a large fraction of falsely significant results or false positives (also *type I error*). Multiple testing procedures allow one to assess the overall significance of the results of a family of hypothesis tests by controlling the type I error rates. The methods that are mostly applied for correction for multiple testing are *Bonferroni, family-wise error rate* [29]. Alternatively the *false discovery rate* [109] can be used.

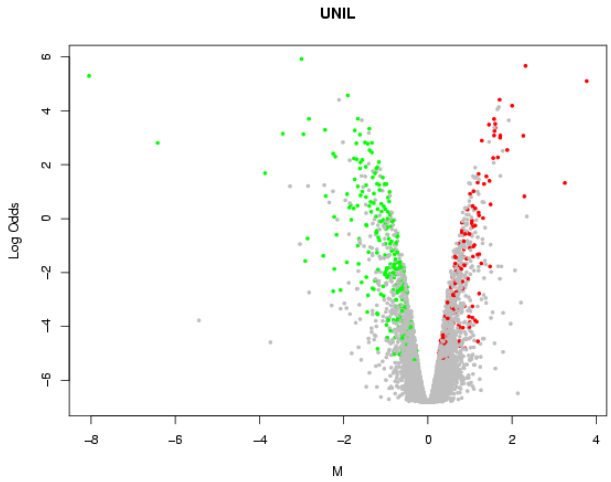
2.4.6 Visualizing differential expression

A frequently used way to visualize differential expression between two samples is an M versus A plot (also known as R versus I plot). In this plot, which is depicted in Figure 2.9a, the vertical axis is the difference between the logarithms of the signals (called M from minus) and the horizontal is the average (A) of the logarithms of the signals. The logarithms are of base two as this gives the useful property that a unit change in M corresponds to a two-fold change in expression. Non-differential genes are centered around $M = 0$ and differentially expressed genes are scattered around this line. A second type

of plot, called a volcano plot, combines the amount of differential expression with its significance. Volcano plots, depicted in Figure 2.9b, plot the M value on the x-axis and its significance for instance the $\log$ odds value calculated by limma on the y-axis.



(a) M versus A plot



(b) Volcano plot

Figure 2.9: An M versus A plot and a volcano plot are two frequently used plots to visualize differential expression between two samples. This figure demonstrates these two plots using data from the CAGE project (see Chapter 4) provided by the CAGE partner UNIL. In an M versus A plot (a), the $\log_2(\text{ratio})$ (M) is plotted in the y-axis and the average intensity on the x-axis. Non-differentially expressed genes are on the $M = 0$ line, the differentially expressed genes are scattered around this line. A volcano plot (b), plots the $\log_2(\text{ratio})$ on the x-axis and its significance on the y-axis. By having both the fold change and its significance, a volcano plot provides an excellent means to view the results of a differential expression analysis.

Chapter 3

Microarray data exchange

The previous chapter showed how microarrays produce vast amounts of data. Unlike for example sequence data, microarray data is only meaningful when we know clearly how the sample that has been hybridized to the array was produced. With the increase in generated data volumes, usually one study only uses part of the information it generates and it becomes important that the data can be efficiently shared with researchers other than the data producers for subsequent analysis. We start by discussing MIAME, a minimal set of information that has to be available to unambiguously interprete a microarray experiment. Next we discuss how this well-annotated data can be exchanged by using the MAGE-ML format. The high-thoughput microarray project CAGE, which is discussed in the next chapter, was the first large scale microarray project that uses these new standards to exchange data between the different project partners, for example between the partner who provides data submission and data repository systems, and the partner who was responsible for data preprocessing. In this Ph.D. we developed a software package that provides a means to exchange these complex data structures to and from R, a widely used statistical programming environment for microarray data preprocessing and analysis. This work was published in the journal Bioinformatics [32].

3.1 MIAME

Microarray experiments often provide more data than what can be analyzed in a single study. In this way analyses of the same data set by different researchers as well as data analyses that compare gene expression patterns generated in different systems with one another, have delivered new insights [44, 107, 89, 90]. Rhodes et al. [89], for example performed one of the first meta-analysis on microarray data and combined the results of four prostate cancer gene expression data sets. Their study found a cohort of genes that were consistently and significantly dysregulated in prostate cancer, which revealed a network of transcriptional regulation in the polyamine and purine biosynthesis pathways. This shows that efficient data sharing of microarray data is important for the field. In the early years of microarrays, microarray data used to be poorly annotated and if the data was published it was almost impossible for other researchers to use this data set for an analysis of their own or reproduce the published results. The data published at those times solely consisted of a gene expression matrix of log intensities or ratios, where each row represents a gene and each column a different sample. Usually the steps taken leading to this transformed data were unknown to other researchers who wanted to use it in their own research. Gene expression data is complex in the sense that for the data to be meaningful, a detailed description of the conditions under which the samples were generated, including the particular state of the living system under study and the perturbations to which it has been subjected, has to be known. Beside a simple expression matrix one also needs other measured information that enable assessment of the quality of the obtained data. Raw image data is typically quantified by image analysis software and the output from this is usually subjected to several preprocessing steps, which eventually lead to the final gene expression matrix. All these transformations are not standardized in this young and rapidly developing field of science and thus there is a need to report what exactly has been done at each step. This is why the Microarray Gene Expression Database (MGED) society (http://www.mged.org) developed the MIAME guidelines [18, 108]. The Minimum Information About a Microarray Experiment (MIAME) describes the minimum information required to ensure that data can be easily interpreted and results from data analysis can be independently verified. This information should also be structured such that it enables easy querying and automated data analysis which implies the need for controlled vocabularies and ontologies to represent the data.

The minimum information about a published microarray gene expression experiment should include a description of the following six sections:

- Experimental design: the set of hybridization experiments as a whole. We should know the author, data submitter, contact information, the type of experiment, the conditions that were tested and the experimental relationships (which sample was hybridized with which array).
- Array design: reporting each array used and each element on the array.
- Samples: describes the samples used, extract preparation and labeling.
- Hybridizations: reports the procedures and parameters of the hybridization.
- Measurements: images, quantifications, and specifications.
- Normalization controls: specify parameters relevant to normalization and control elements.

After the endorsement of this standard by several major journals such as *Nature* and *Cell*, microarray data should comply with the standard before it can get published. At the same time, the largest public microarray repositories, such as GEO from the National Center for Biotechnology [35], ArrayExpress at the European Bioinformatics Institute [19], and the Center for Information Biology Gene Expression Database (CIBEX) at the DNA Data Bank of Japan [57], started to provide support to submit, store, and view this extra information.

3.2 MAGE

MIAME can be seen as a set of rules that must be fulfilled for a microarray dataset to become public and be used by other researchers. However MIAME does not provide a means to model and encode the information. The MicroArray Gene Expression group (MAGE) was set up to tackle these issues and aims to set standards for data representation [106]. The MAGE group consists of pioneering microarray scientists from different academic institutions and industry. The group established a data object model (MAGE-OM), a data exchange format (MAGE-ML) and a freely available software toolkit that enables integration of MAGE-ML into end-users' systems: MAGEstk.

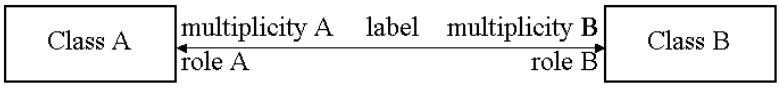


Figure 3.1: UML representation of an association between classes. Associations can have a label and role and are of a certain multiplicity.

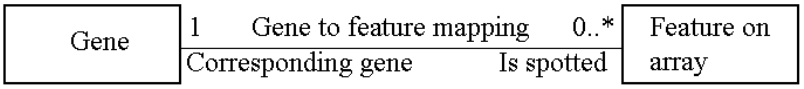


Figure 3.2: Example of UML representation of the association between genes and the features corresponding to these genes spotted on the array. This UML example can be read as follows: there are genes that are not represented on the array, genes can be represented many times on the array by different features and each feature represents exactly one gene.

3.2.1 MAGE-OM

The Microarray GeneExpression Object Model has been modeled using the Unified Modeling Language (UML). UML is a specification of the Object Modeling Group (OMG) that can be used to model application structures, behavior, and architecture as well as business process and data structure (http://www.uml.org). Before going through some of the main features of the MAGE Object Model (MAGE-OM), UML diagrams will be briefly explained.
A class is a representation of an object, it can be seen as the template from which objects are created. In UML, classes can be depicted as boxes with three sections, one section for the class name, one for its methods and a third section containing its attributes. Classes can be associated or related to other classes, these associations between classes are depicted as lines, linking the respective classes. Associations can connect one class to exactly one other class or to none or multiple instances of the other class. This multiplicity of the association is labeled on either end of the association line, one multiplicity indicator for each direction can have a label (see Figure 3.1).

An example of an UML diagram is given in Figure 3.2.

An association can be directed, in this case a filled triangle indicates the direction of association. Often similarities between classes exist and classes may share features of one common class through inheritance. Inheritance is depicted in UML by a line with a closed arrowhead, pointing from the subclass

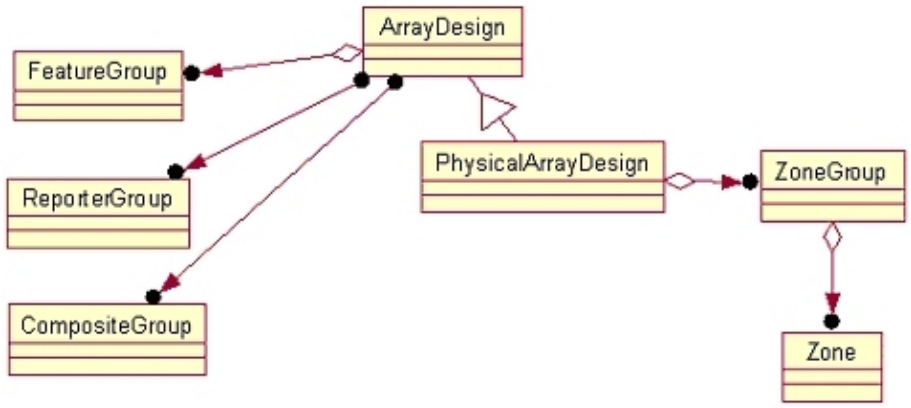


Figure 3.3: ArrayDesign package - UML model. The ArrayDesign package describes the layout of the array. The actual ArrayDesign is a PhysicalArray Design which is a subclass of ArrayDesign. The FeatureGroups that are associated with the ArrayDesign class contain the features of the Array. Similarly ReporterGroups and CompositeGroups contain reporter and CompositeSequences respectively. Features, Reporters and CompositeSequences are described in the DesignElement package. Figure from http://www.mged.org

to the superclass. A class can be built up from other classes, in UML this is depicted as a diamond. Using UML, we discuss some of the main features of the MAGE-OM model which will later be used in the RMAGEML package to import and handle MAGE-ML documents in Bioconductor. As the MAGE-OM contains many classes that cannot be all represented in one page, the classes are divided into several packages.

ArrayDesign package

The ArrayDesign package describes the layout of the array, and thus the way target sequences are arranged on the array. It consists of design element groups, which are either reporter groups, feature groups, or composite sequence groups. *ArrayDesigns* can be virtual for instance the output of an analysis at the composite level without features and reporters associated with it. A *PhysicalArrayDesign* is a subclass of *ArrayDesign* and represents an actual array. In addition to the design element groups, it also contains information about the zone layout (grid).

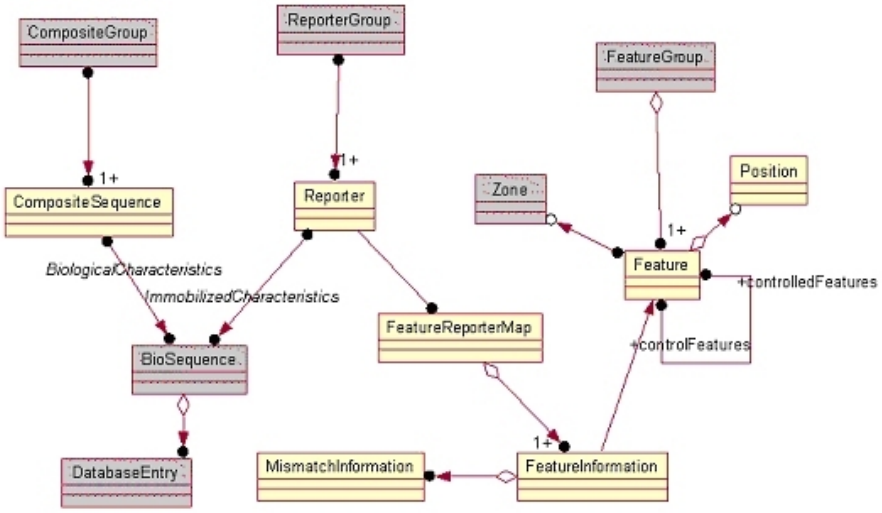


Figure 3.4: DesignElement package - UML model. There are three types of design elements: Features, Reporters, and CompositeSequences. Features have an actual location on the array and can be regarded as the spotted or synthesized material. Reporters associate with these Features and have a BioSequence class linked to them which can be characterized by a database entry, for example a HUGO ID. A Reporter can be represented many times on the array by different Features, which corresponds with placing the same probe many times on the array. CompositeSequences are again one level higher and represent for example different splice variants of the same gene. Figure from http://www.mged.org

DesignElement package

There are three types of design elements: *Features*, *Reporters*, and *CompositeSequences*. A *Feature* has a position on the array. A *Reporter* corresponds to the probe spotted or synthesized on the array and can be characterized by one or more *BioSequence* objects that in turn can be characterized by database entries. There can be many features that correspond to the same reporter, this translates to spotting the same probe many times on the same array. The *CompositeSequence* class is more abstract and can be represented by multiple reporters on the same array. They represent for example different splice variants, thus composite sequences are characterized again by sequences with corresponding database entries.

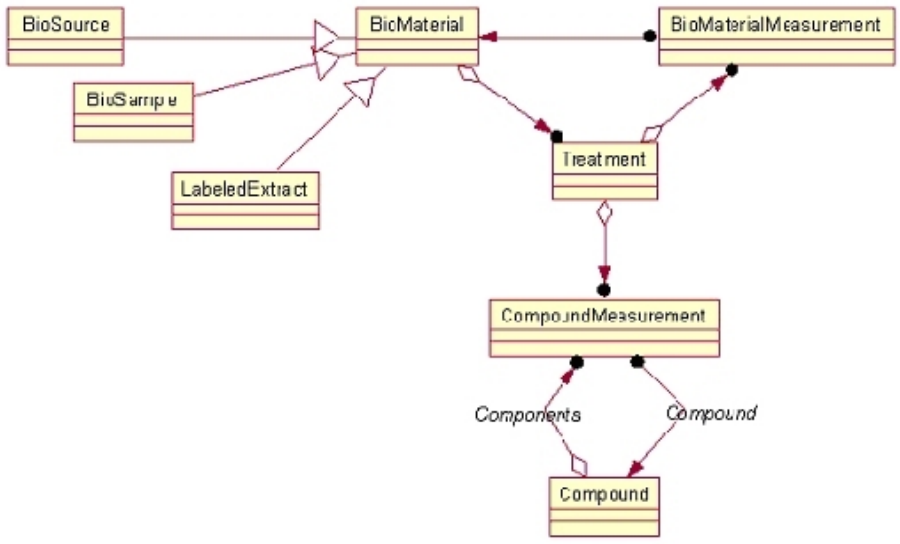


Figure 3.5: BioMaterial package - UML model. The BioMaterial package models the different biological materials that are used at various stages of the microarray experiment. An example of BioMaterial is a tissue sample. This tissue sample will be modeled as a BioSource, which is a subclass of BioMaterial. The tissue sample can undergo different Treatments which can involve different Compounds which in the end for example lead to Cy3 labeled cDNA derived from the original tissue sample one started from. This labeled cDNA can be modeled as a LabeledExtract, which is also a subclass of BioMaterial. Figure from http://www.mged.org

BioMaterial package

Biological materials used at various stages of the microarray experiment, can be modeled with the *BioMaterial* class. Cell lines or tissue or other initial sources of material eventually used in the hybridization are modeled as *BioSource*. The *BioSample* is created after extracting material from the *BioSource*. The *LabeledExtract* is the last state of the biomaterial before it is hybridized to the array. A biomaterial can be obtained as the result from a chain of treatments, involving a certain amount of one or more biomaterials. If an additional compound was used in the treatment, this can be modeled using the *Compound* class.

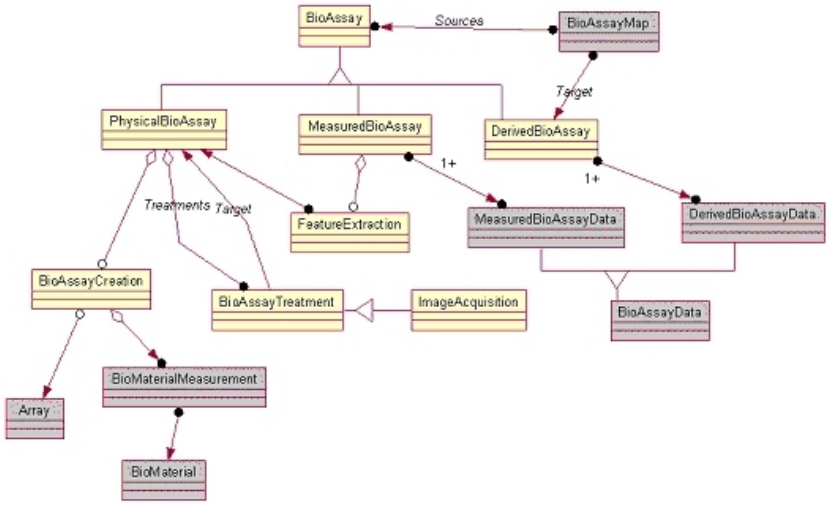


Figure 3.6: BioAssay package - UML model. The BioAssay package represents a a wet-lab or computational assay. An example of a BioAssay is the acquisition of images by scanning the hybridized microarray slides. Another example would be the image analysis of these images which result in obtaining data per feature based on these images. This would be represented by a MeasuredBioAssay (as we do a measurement), which is a subclass of BioAssay. Figure from http://www.mged.org

BioAssay package

The *BioAssay* package represents a wet-lab or computational assay. Examples of such assays are the event of joining an *Array* with a *BioMaterial* preparation, the acquisition of images or data per feature based on those images. The subclasses of *BioAssay* are *PhysicalBioAssay*, *MeasuredBioAssay* and *DerivedBioAssay*. The *PhysicalBioAssay* represents the actual hybridization of the samples to the array. The *MeasuredBioAssay* represents the image analysis results, where several *QuantitationTypes* were measured. From here data can be preprocessed or transformed by a *Transformation BioEvent*, which is performed in a *DerivedBioAssay*. A package related to the BioAssay package is the BioAssayData package. A *MeasuredBioAssay* for example generates a *MeasuredBioAssayData* object.

BioAssayData package

A *BioAssayData* class represents the data created when a *BioAssay* is performed. Two subclasses of the *BioAssayData* class can be defined. The *MeasuredBioAssayData* class represents the data associated with a *MeasuredBioAssay*, produced by a feature extraction. The second subclass: *DerivedBioAssayData* is the output of a transformation described in the *DerivedBioAssay* class. This transformation may involve a *QuantitationType* mapping, from one or more *QuantitationTypes* to another (e.g., from raw intensities to mean $\log_2$ ratios). Similarly design element mappings can be applied. In MAGE, data objects are regarded as 3-dimensional matrices. In this representation *BioAssays* are along one dimension, *DesignElements* along a second dimension and *QuantitationTypes* along the third dimension.

QuantitationType package

A *QuantitationType* represents the type of data that is measured or calculated. There are two types of *QuantitationTypes* that are distinguished: *StandardQuantitationTypes* and *SpecializedQuantitationTypes*. *StandardQuantitationTypes* usually come from feature extraction and include *MeasuredSignals*, *DerivedSignals* and *PresentAbsent* calls. *SpecialQuantitationTypes* are user-defined *QuantitationTypes*: an example is a p-value.

3.2.2 XML and MAGE-ML

XML is a markup language similar to the WWW's Hypertext Markup Language (HTML) and like HTML, XML is based on the Standard Generalized Markup Language (SGML). XML was designed by the World Wide Web Consortium (W3C) to have a flexible and widely used extensible markup language. XML version 1.0 was first defined in a 1998 W3C Recommendation, and from then on was used as an 'informal' standard. The term 'markup' refers to a method to provide meta data, thus data about other data. In contrast to HTML, XML is extensible and can further be considered a meta-language because it can be used to create other metadata languages, such as XHTML (the new HTML) and MAGE-ML. SGML-based markup languages use literal strings of characters, called *tags* to delimit the major components of the meta-data, called *elements* [12]. Furthermore, elements can contain additional attributes. In contrast to HTML, XML is all about the description of the data and contains no instructions about its presentation. The use of descriptive tag and

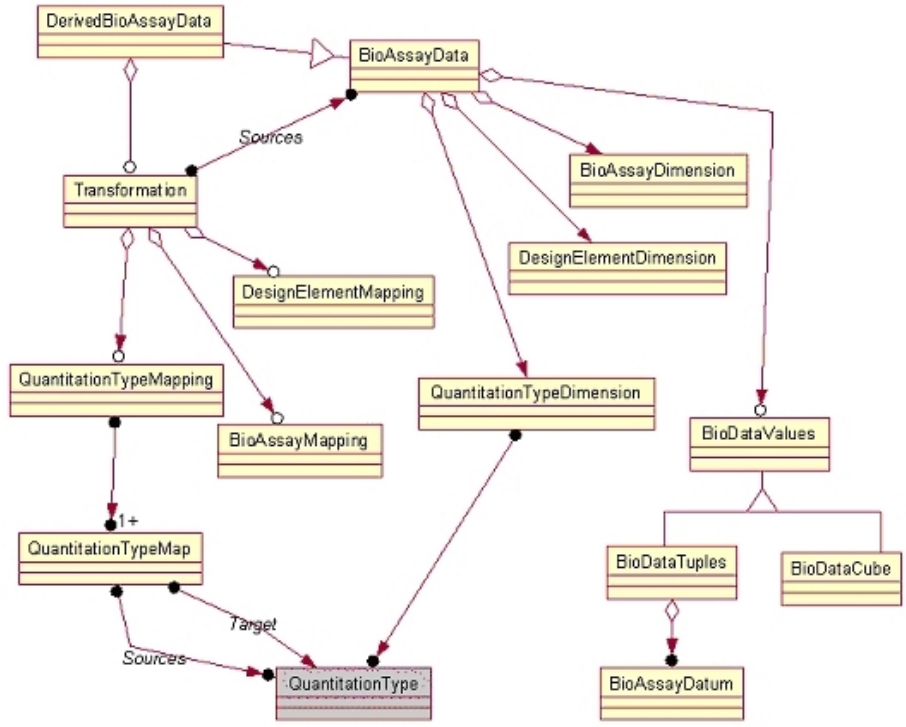


Figure 3.7: BioAssayData package - UML model. BioAssayData objects are obtained when a BioAssay is performed. An example is the MeasuredBioAssayData class, which is a subclass of BioAssayData and represents the data obtained from scanning the microarray and subsequenty image processing. Figure from http://www.mged.org

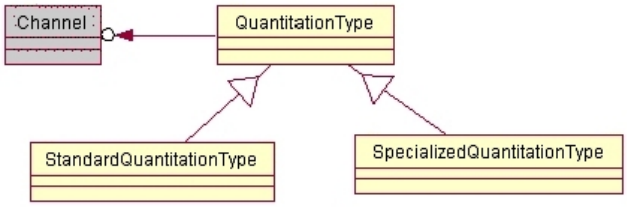


Figure 3.8: QuantitationType package - UML model. The QuantitationType package represents the type of data that is measured or calculated. Two types of QuantitationTypes can be distinguished. StandardQuantitationTypes are a subclass of QuantitationType and represent common datatypes which are usually present in all microarray output such as median Cy3 foregound intensities for two-color hybridizations. SpecializedQuantitationTypes are another subclass of QuantitationType and represent for example less common or user defined quantities. Figure from http://www.mged.org

attribute names allows software to extract structured data from XML. XML documents that are conform with the XML syntax specification are said to be *well-formed*. Further, XML documents can be validated against a schema, which is a document describing constraints and the grammar that can be used in the XML document. A frequently used schema is the Document Type Definition (DTD). Based on the excellent properties of XML to describe data, XML was used to develop the Microarray Gene Expression Markup Language (MAGE-ML). MAGE-ML is an XML language designed to describe and communicate information about microarray experiments [106]. MAGE-ML has been automatically derived from Microarray Gene Expression Object Model described above. Below, a small example of MAGE-ML code is shown, making a reference to the quantitation types that are measured.

3.2.3 MAGEstk

The MAGE Software Toolkit is a collection of open source packages that implement the MAGE Object Model in Perl and Java and recently also C#. This toolkit is developed for developers of applications that need to manage instances of MAGE-OM and is available from http://mged.sourceforge.net/software/MAGEstk.php. The Java implementation was used in the RMAGEML package [32] (see Section 3.4).

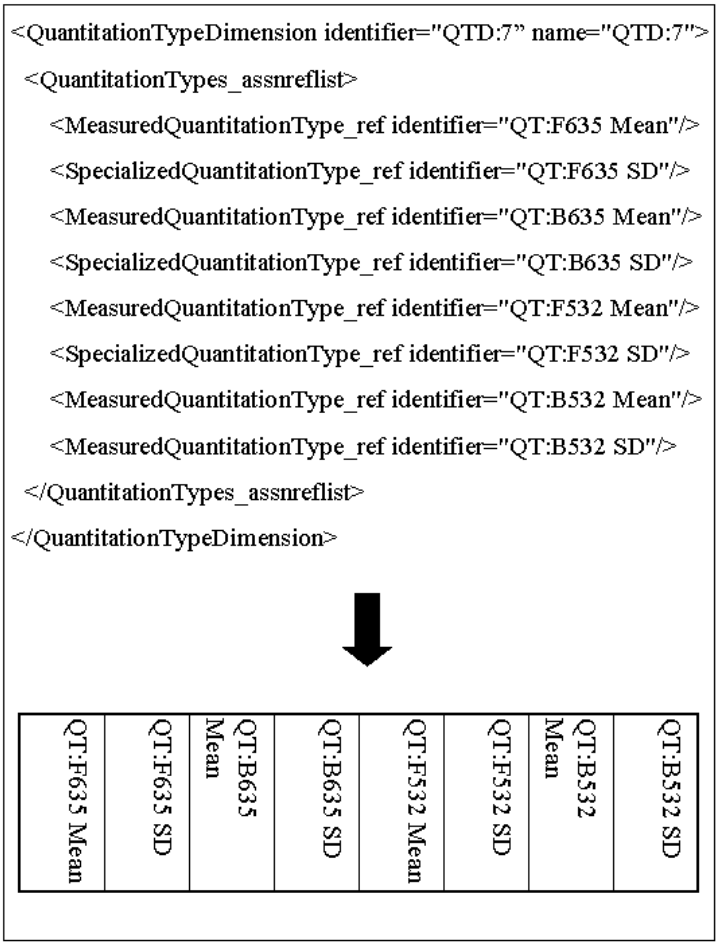


Figure 3.9: A simple example of MAGE-ML code. This example shows how quantitation types referenced in the MAGE-ML document, map to a data matrix containing these quantition types as columns. The definition of these quantitation types can be found elsewhere in the document.

3.3 MAGE compliant microarray databases

MAGE-compliant microarray databases support the MAGE-OM and can handle MAGE-ML data to import or export data. These databases are also MIAME compliant and thus contain the MIAME information. The next sections give an overview of the currently available MAGE compliant databases.

3.3.1 ArrayExpress

The most important microarray data repository that is fully MAGE compliant is ArrayExpress [19] (http://www.ebi.ac.uk/arrayexpress/), maintained at the European Bioinformatics Institute (EBI), located in Cambridge, UK. At the time of writing ArrayExpress contained 35,591 hybridizations coming from 1,271 experiments. All this data can be queried through the ArrayExpress interface and this data is also made available in MAGE-ML format together with the MAGE-ML files describing the arrays. Recently an ArrayExpress data warehouse has been developed where users can perform gene specific queries. Tightly linked with ArrayExpress is Expression Profiler, a web tool for gene expression analysis, which will be discussed in Chapter 6.

3.3.2 MIAMExpress

MIAMExpress [87] (http://www.ebi.ac.uk/miamexpress/) is a web-based MIAME supportive data submission tool developed at the EBI. It is written so that local installations of the tool are possible. Different web pages lead the data submitter through the data submission and previously submitted data (such as protocols) can be reused in subsequent submissions. The MIAMExpress database comes with additional Perl scripts that enable to create MAGE-ML documents from the data in the database. Recently an *Arabidopsis*-specific version of MIAMExpress was developed: At-MIAMExpress [85].

3.3.3 BASE

One of the most popular and open source microarray database systems is the BioArray Software Environment (BASE, http://base.thep.lu.se/), developed at the University of Lund, Sweden. The system manages biomaterial information, raw data and images, and provides integrated tools for normalization, data viewing and analysis tools [93]. BASE is MIAME supportive and will in the future support export of MAGE-ML format data and import of MAGE objects.

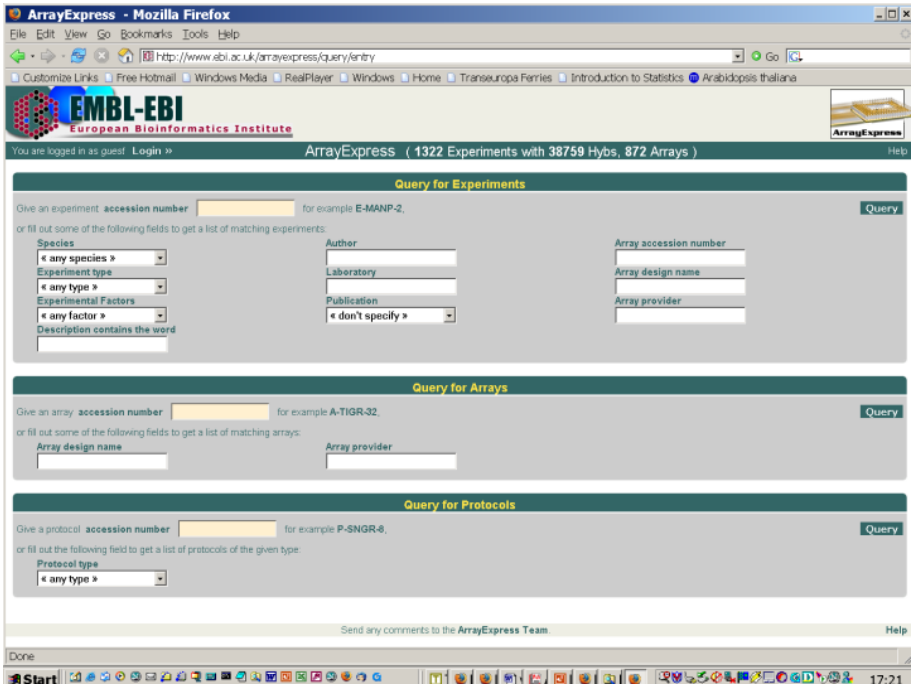


Figure 3.10: ArrayExpress: a MIAME compliant microarray data repository at EBI. This screenshot shows a query page, after querying the repository for example all experiments having a name that contain ‘CAGE’, a results page is obtained from which MAGE-ML data can be downloaded or one can export the data to the Expression Profiler suite for further analysis.

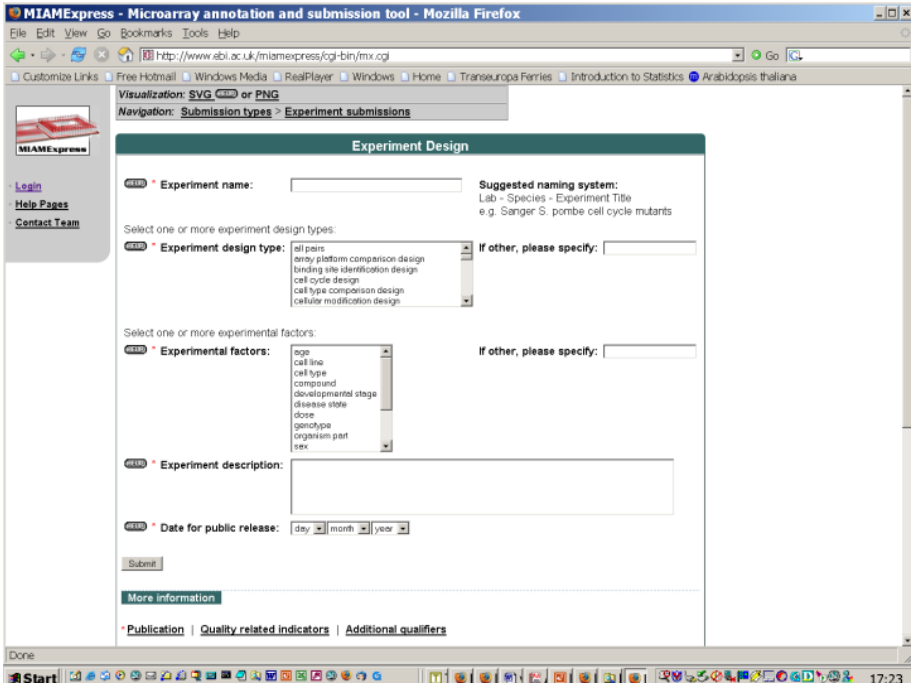


Figure 3.11: MIAMExpress: a MIAME compliant data submission tool at EBI. This screenshot shows a page to enter MIAME information about the experiment such as the experiment name, a description and experimental factors (what is tested in the experiment). Once data is submitted into MIAMExpress, EBI will run a script that creates a MAGE-ML document of the experiment containing the submitted information.

3.3.4 Other MAGE compliant databases

- maxdload2: maxdLoad2 (http://umber.sbs.man.ac.uk/microarray/maxd/maxdLoad2/) is a relational database schema and Java application for microarray experimental annotation and storage developed by the University of Manchester [46]. It offers full support for the MIAME standard, uses the MGED Ontology and exports data in MAGE-ML format.

- MARS: the Microarray Analysis and Retrieval System (MARS, https://mars.genome.tugraz.at/marsWEB/) provides a comprehensive MIAME supportive suite for storing, retrieving, and analyzing multi channel microarray data [83]. The system comprises a laboratory information management system (LIMS), a quality control management, as well as a sophisticated user management system. It incorporates ontologies and can export MAGE-ML.

- MiMiR: The MiMiR database (http://microarray.csc.mrc.ac.uk/_private/activities/data_warehouse_text.htm) provides a comprehensive infrastructure for microarray data annotation, storage and exchange and is based on the MAGE format [86]. MiMiR is MIAME supportive and data can be exported as MAGE-ML

- Longhorn Array Database: The Longhorn Array Database (LAD, http://www.longhornarraydatabase.org/) [70] is an open-source, MIAME compliant microarray database based on PostgreSQL and Linux. The database currently is not yet able to export MAGE-ML but the authors state that with a minimal effort their system can export MAGE-ML and developing this functionality is one of their main future goals.

- MADAM: the MicroArray DAtabase Manager (MADAM, http://www.tm4.org/madam.html), which is part of the TM4 software suite developed by the Dana-Farber Cancer Institute, is currently being updated to support MAGE-ML import and export.

3.4 RMAGEML

As discussed above, microarray data has been generated in many different formats and often lacks standardized annotation and documentation. The MIAME (minimum information about a microarray experiment) guidelines define a set of fields for describing and annotating microarray data [18]. Major journals, such as *Nature, Cell,* and *the Lancet* have adopted these guidelines and

have made submission of microarray expression data, coupled with MIAME-compliant documentation, compulsory for publication [25].

The MIAME recommendations define an information set, but does not address the formatting and exchangeability of numerical microarray data. Formatting and exchangeability issues have been confronted in the microarray gene expression markup language (MAGE-ML), which is an XML standard for serializing data structured according to the MAGE object model (MAGE-OM) [106] (i.e., representing the MAGE-OM database object as a file). This data model incorporates the MIAME information and addresses many aspects of microarray experiment documentation and data encoding that are not covered by MIAME. Several databases have been developed that contain these MIAME-compliant data sets in MAGE-ML format. Despite the large availability of MAGE-ML format data, efficient tools to handle this format were lacking even in the most popular microarray analysis project, Bioconductor. Within Bioconductor, various tools supporting the generation of MIAME-compliant documentation had already emerged, for example the `tkMIAME` widget of Bioconductor *tkWidgets*.
Our Bioconductor package *RMAGEML* extracts information from MAGE-ML documents for cDNA microarray experiments and maps this information to Bioconductor R objects for the analysis of cDNA microarrays. Existing MAGE-ML documents can also be updated with normalized data. Though generic tools for XML processing in R can be used to work with MAGE-ML documents, such an approach is fairly laborious and would likely require recoding whenever the MAGE-ML information model or language is revised. The current version of *RMAGEML* transforms documents and data for cDNA experiments; work is in progress for a wider variety of microarray platforms.

3.4.1 Description

MAGE-ML is derived from the MAGE-OM model, which is an object model specification standardized by the Object Management Group (OMG) and maintained by the MGED Society. MAGE-OM consists of several classes of data structure. Examples of classes include `BioSequence`, `ArrayDesign`, `Array`, `Protocol`, `AuditAndSecurity`. The MGED society supplements the MAGE-OM with Software ToolKits (MAGEstk) in Java and Perl that specify data structure classes for programming within MAGE-OM, and serialization and deserialization between MAGE-OM structures and MAGE-ML. The *RMAGEML* package currently allows import to two Bioconductor data structures: `limma::RGlist` and
`marrayClasses::marrayRaw`. The MAGE-OM classes describing the

features needed to create these objects are ArrayDesign, BioAssay, BioAssayData, BioMaterial, BioSequence, and DesignElement. The DesignElement package contains a mapping of *Features* (which are the actual locations on the array) to *Reporters* (what is present on those locations). The DesignElement package also provides a mapping from *Reporters* to their corresponding *BioSequence* references. These *BioSequence* objects are characterized by their name and database entries in the BioSequence package. The ArrayDesign package contains information on the layout of the array. From this package, we can derive the position of each *Feature* on the array in terms of *Zone* (block or grid) and row and column within each *Zone*. The BioAssayData package describes the feature references that were assayed and the measured and derived *QuantitationTypes*. The BioAssay package describes the different steps in the microarray experiment. Finally, the BioMaterial package describes how a sample is treated (e.g. to obtain labeled samples for hybridization).

3.4.2 Architecture: lightweight R to Java interface

The *RMAGEML* package implements a three-tiered architecture, transforming MAGE-ML to R objects via middleware that interfaces R to a Java software kit created expressly for working with MAGE-ML documents. The Bioconductor *RMAGEML* package is written in R and Java, and makes use of the Java-MAGEstk (http://mged.sourceforge.net/software/MAGEstk.php) application programming interface. Initially, the use of MAGEstk took place via the *SJava* interface of Duncan Temple Lang (http://www.omegahat.org). *SJava* facilitates creation and manipulation of Java classes and methods in R as *references*, allowing efficient reuse of Java software with minimal copying of information between Java and R. The overall architecture of *RMAGEML* minimizes the involvement of R with details of XML processing and maximizes the use of MGED-sponsored software for coupling data structure and serialization processes to the MAGE-OM and MAGE-ML specifications. However, about one year after the first release of *RMAGEML* the development of *SJava*, on which our package depends, stopped. To keep *RMAGEML*, which is an essential component of our pre-processing pipeline (discussed in the next chapter), in the running we had to develop our own R-Java interface. To avoid the problems other R-Java interfaces have run into (because they tried to be very general), we chose to develop a lightweight package-specific bridge.

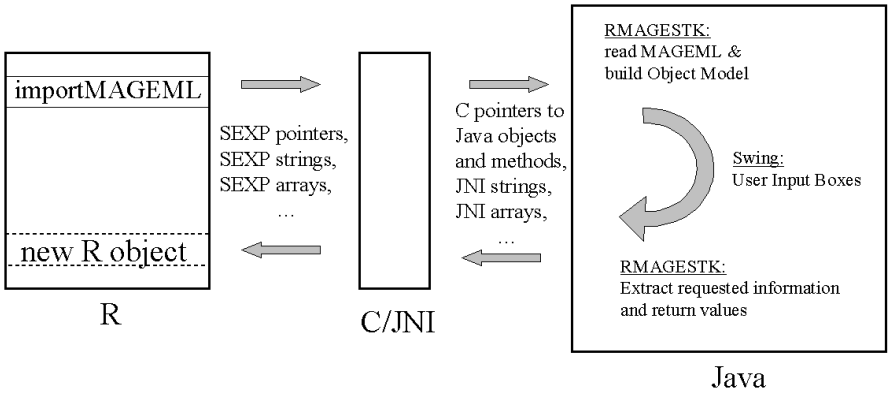


Figure 3.12: RMAGEML: R - C/JNI - Java interface. A lightweight C/JNI bridge enables communication between the R programming environment and the Java Virtual Machine. At the R side, the RMAGEML functions that can be employed by the users to create R objects from MAGE-ML data. At the Java side, different methods are implemented to extract and add MAGE-ML data from and to the MAGE-OM in memory.

Interfacing R and Java

JNI is the *Java Native Interface* and interfaces C and Java. One way to use JNI is to call Java from C. As calling C from R is easy (R is written in C and methods exists to smoothly include new C code), we chose to implement a C/JNI bridge interfacing R and Java. The only task of this bridge is to relay R to C-calls and Java object creation and method calls.

Lightweight interface

The only functionality of the C code is to forward the calls it gets from R to Java and return data and object references from Java back to R. To limit complexity we choose to use one Java class that serves as main interface to the other Java classes.

Starting the Java Virtual Machine

We start the Java Virtual Machine (JVM) using the JNI function *JNI_CreateJavaVM*. This function returns a pointer to the JVM in C and we return that C pointer to R.

Creating a new Java object

As we only have one Java class that takes all the C-calls, we only need to create one Java object from R and this will be our main interfacing object. Using the Java Virtual Machine pointer, we look for our main Java class, named 'RMAGESTK' using the JNI method *FindClass*. Then we look for it's 'new' method by using the JNI method *GetMethodID* with '<init>' as name of the method to look for. Then we generate the new Java object by calling its '<init>' method using the JNI method *NewObject*. At last we return a reference of this object to R so it can be used for later method calls to this object.

Calling Java methods from R

To call Java methods we first pass the Java object reference from R back to C and there we use again the *GetMethodID* method to find the id of the Java method we want to call. Note that together with its name, we also have to specify its arguments and return type. Once we have the method id, we can call the method using the JNI method *CallObjectMethod*. To pass the string arguments, we first have to convert them to jstrings, which are Java strings, by using the JNI method *NewStringUTF*.

Returning data from Java to R

The data coming from Java methods that do have a return type need to go back to R. Again we first identify the Java method and call the method. Then we use the JNI methods *GetArrayLength*, *GetObjectArrayElement* and *GetStringUTFChars* to determine the length of the returned array, get an object from each array position and convert this object to a character respectively.

3.4.3 Usage

The *RMAGEML* package runs on Linux and Windows. It has a manual and vignette, which is an interactive document containing executable code chunks giving a more problem-oriented style of help. As an example, you can go to EBI ArrayExpress (http://www.ebi.ac.uk/arrayexpress/) and query the database for the experiment with accession E-MEXP-5. Download the E-MEXP-5 MAGE-ML file, together with the MAGE-ML document describing the array used (accession A-MEXP-7). Place these files in a common directory `dir`, serving as the current working directory of R, with the *RMAGEML* package and all dependencies loaded. You can now import the E-MEXP-5 data to *limma* using the call `importMAGEML(dir, package="limma")`. When

a data set containing raw microarray data is imported and normalized in R, *RMAGEML* can update the MAGE-OM with the normalized data, the description of the normalization protocol used, and the newly generated derived quantitation types. After the MAGE-OM is updated, the actual MAGE-ML document can be written to a file. More examples on how to use the RMAGEML software package can be found in Appendix B.

3.5 Discussion

In recent years the way we communicate and store microarray data has changed. Microarray data sets often contain more information than what is used by the initial data analysis and thus can be used by other researchers to do new discoveries. However microarray data is complex and to be useful to others, the experiment and data have to be well annotated. The MIAME standards established a minimum set of descriptions that has to be available to make the data interpretable by the community. In a next phase the MAGE-ML data format was developed to provide a data exchange format for microarray data, including the MIAME information. One of the first databases to make microarray data available in MAGE-ML format is ArrayExpress at the European Bioinformatics Institute (EBI) [19]. Depending on the size of the data file, the data in ArrayExpress are distributed in one or more XML files, which each include one or more MAGE-ML packages. The raw data from the measurements are stored in separate tab-delimited text files. ArrayExpress currently contains 1,271 experiments in MAGE-ML format, together representing tens of thousands of hybridizations. However, tools to import and handle these MAGE-ML documents in a data analysis environment were lacking. To fill this gap, we developed an add-on software package (*RMAGEML*) for the statistical programming environment R, which enables to exchange these complex data structures to and from R. This R package was a crucial part of the CAGE preprocessing pipeline described in the next chapter. The package was made generally applicable and is now part of the Bioconductor project where it can be downloaded. Since the release of MAGE-ML as data exchange format, major microarray groups that are part of the MGED society, have been pushing this format forward to become the standard microarray exchange format. The lack of software tools to handle these documents and the fact that of the major microarray repositories only ArrayExpress makes available MAGE-ML documents, shows that this standard is too complex and requires an grand effort to support.

Chapter 4

Compendium of Arabidopsis Gene Expression

The CAGE project, has been the major project during this Ph.D. and aims to build a compendium of Arabidopsis Gene Expression covering 1,000 different biological conditions and generating 100,000,000 datapoints. Firstly a new microarray platform (CATMA) was introduced [49]. Next we published a benchmark of this new platform [4], comparing it to already existing commercial platforms for Arabidopsis gene expression studies. The MAGE-ML exchange format, discussed in the previous chapter, was chosen as data format to exchange microarray data between the CAGE partners and to make the compendium available to the community in a well-annotated standardized manner. The challenge in this Ph.D. and major deliverables for CAGE were to develop an automated system to preprocess these large volumes of data, to provide quality assessment results to the data producing partners and use, to our knowledge, for the first time MAGE-ML as data exchange format between different partners working together in a large-scale microarray project.

4.1 The CAGE project

In October 2002 a European Framework V project, Compendium of Arabidopsis Gene Expression (CAGE, http://www.cagecompendium.org) started. CAGE is a three-year collaboration between eight European data producers (see Table 4.2), the VIB Microarray Facility, the European Bioinformatics In-

stitute, and ESAT-SCD. The project aims to demonstrate the utility and technical quality of a microarray platform under development for analysis of gene expression in *Arabidopsis*. A total of 4,000 microarray hybridizations were scheduled to be produced. This will cover 1,000 different biological conditions samples for each of which two biological and two technical replicates will be performed. These samples consist of wild type growth stages, stress conditions, mutants, and research specific samples. Ultimately this will result in a compendium of *Arabidopsis thaliana* gene expression. At the time CAGE was the first large-scale genomewide expression analysis project undertaken in Europe and would set the stage for future implementation of functional genomics platforms to generate integrated databases. The project will as such generate a unique knowledge base for analysis and improvement of plant function. Our group was responsible for two workpackages: (1) statistical pre-processing of the extracted data and data curation and (2) compendium analysis and precomputing of results.

4.2 *Arabidopsis thaliana*

Arabidopsis thaliana (Figure 4.1) is an annual plant of the *Brassicaceae* family that includes cultivated species such as radish and cabbage. The plant grows in temperate climate regions of the world. The plant has since long been the model species for studying genetics in plants because of several features such as a small genome size, a rapid life cycle (about 6 weeks), prolific seed production, ease to cultivate in restricted space, and the availability of a large number of mutant lines. Not surprisingly, it was also the first plant to have its genome fully sequenced [58]. The 125Mb *Arabidopsis* genome consists of 5 chromosomes and it is estimated to code for about 26,000 genes (TIGR release 5.0). The function of only a small fraction of these genes is known [11]. To increase our knowledge on the *Arabidopsis* gene functions, it is crucial to obtain data from complex phenotypes, such as transcriptome, proteome, and metabolome; and to adopt computational approaches for mining and modeling this information [4]. The CAGE consortium would contribute by providing a large transcriptome resource that can be used for functional characterization of the *Arabidopsis* genome.

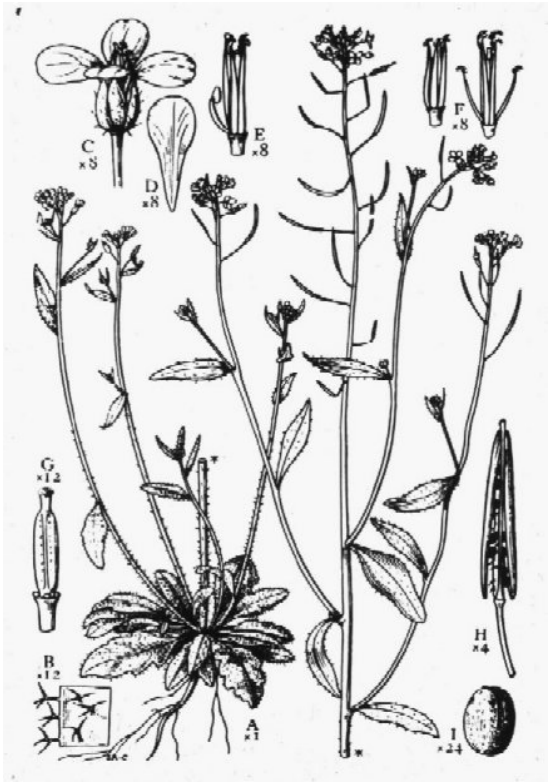


Figure 4.1: The plant model species *Arabidopsis thaliana*. Because of many properties, such as rapid life cycle, ease to cultivate and small genome size, *Arabidopsis thailiana* has been since decades a model species to study plants. It was also the first plant to have its genome fully sequenced [58].

4.3 CATMA array

The CAGE consortium is a follow-up of the CATMA consortium with EBI and ESAT-SCD as new partners. The CATMA consortium (Complete Arabidopsis Transcript MicroArray) earlier developed a new microarray for *Arabidopsis* gene expression studies. The goal of CATMA was to create Gene-specific Sequence Tags (GSTs), which can be spotted on an array to do microarray expression studies but in addition to this, the GSTs have also been cloned in bacterial plasmids designed for gene silencing in plants by RNAi [49] (see Chapter 2). For both applications it is required that the GSTs are chosen such that they are highly specific to their target gene with minimal overlap to other genes to avoid cross-hybridization and silencing of related genes. This was achieved by designing GSTs with the gene prediction algorithm and primer design software SPADS [113]. The SPADS GST selection can be divided into four steps [113, 49]. First, the exons for each gene model are matched against the entire genome sequence using BLASTn [5] and segments with significant homology hits are discarded. In a second step, PCR primers pairs are designed to the selected exonic regions by the Primer3 software [92]. Then these primer pairs are filtered by eliminating any pair that has a significant match in a surrounding window of minimum 400bp, this to avoid amplifying wrong PCR products. In a last step the amplicons are matched against the full genome sequence to see if there are potential paralogous gene sequences. For the GST sequences, care was taken so that more than 85% of the GSTs have less than 70% homology to any other sequence in the *Arabidopsis* genome. The length of the selected sequences ranges from 150bp to 500bp, the length distribution of the sequences spotted on the first CATMA array, CATMAv1, are given in Figure 4.2.

To produce the GST sequences spotted on the array, several rounds of PCR amplification are performed. First, the initial genomic fragments are amplified using the primers designed by SPADS. These primers contain a 5' extension to which a second set of primers can bind for a subsequent amplification round. These 5' extensions were chosen such that the GST amplicons can be stored in 384 well plates and that 16 + 24 primers are needed to amplify every GST on the plate, corresponding to the number of rows and columns of the plate. These secondary primers reduce the risk of cross-contamination from adjacent wells (a major problem for cDNA microarrays). With nucleotide lengths varying from 150bp to 500bp, the CATMA array fills the gap between the short oligonucleotide Affymetrix array and the spotted cDNA arrays. Each of

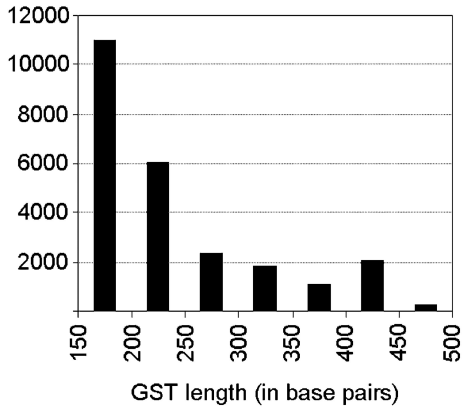


Figure 4.2: Length distribution of sequences spotted on the CATMA v1 array. Figure from [49]

these platforms has its advantages and disadvantages. It is difficult to get full genome coverage from cDNA libraries to create full genome cDNA arrays as all genes are not expressed simultaneously and at the same level. Thus many different samples are required to obtain full coverage. Amplification of fragments with PCR, as is used for cDNA and CATMA array, make the process prone to contamination. cDNA arrays also have a limited specificity, for example genes belonging to the same gene family might cross hybridize on a probe that was intended to measure expression of one of the gene family members. An advantage of cDNA arrays however is that they are cheap to produce and it is feasible for labs to make their slides. Microarrays platforms such as the CATMA platform, can be more specific than cDNA arrays as the length of the probe does not cover the full cDNA length and one can chose specific sequence regions that show minimal homology with other expressed sequences in the genome. Another advantage of the CATMA array is that the GST sequences are cloned in bacterial plasmids, that can be used in functional studies by RNAi (see Chapter 2). Affymetrix GeneChips on the other hand, are expensive but have the advantage that absolute measurements are acquired and a high number of probes can be put on one slide.
The pilot CATMA array, CATMAv1 (accession A-MEXP-10 in ArrayExpress) consisted of 19,749 unique GSTs and 768 positive/negative controls. A second version of the array, CATMAv2 (accession A-MEXP-58 in ArrayExpress) contains 22,495 unique GSTs and 768 positive/negative controls. The CATMAv1 array was used in a benchmarking paper where we compared the novel CATMA array with already existing Affymetrix and Agilent *Arabidopsis* ar-

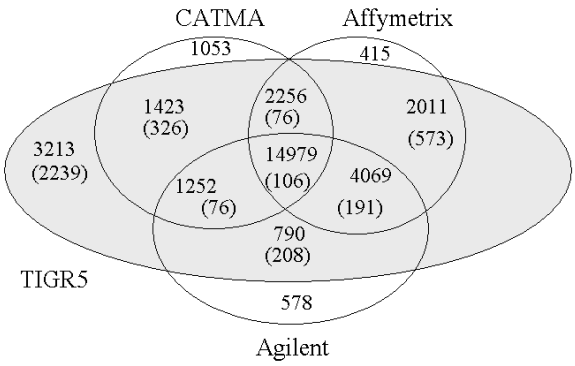


Figure 4.3: The coverage of the CATMAv1 microarray was compared in silico with Affymetrix and Agilent microarray platforms and the TIGR 5.0 gene models. Numbers between parentheses refer to non-protein coding sequences.

rays [4]. The sensitivity and specificity of the different platforms was compared by a spike-in experiment and it was concluded that the CATMA platform compares well to its commercial competitors. An in silico coverage of the different platforms compared to each other and to the TIGR 5.0 gene models was performed using BLASTn (see Figure 4.3). The three platforms have 14,979 genes in common and each covers a unique set of extra genes.

Figure 4.4 shows the top four grid rows of a hybridized CATMA array.

4.4 CAGE sample overview and sample annotation

The CAGE consortium aimed to produce a well-annotated data set that can easily be used by the research community. The use of MAGE-ML has been discussed in Chapter 3 and provides an excellent way to capture this detailed annotation. To correctly annotate the samples, controlled vocabularies (ontologies) were used. Plant structures were annotated using the Plant Ontology (PO, http://www.plantontology.org). The Plant Ontology was developed by the Plant Ontology Consortium and can be used to describe plant structures and growth and developmental stages. CAGE only used the plant structure ontology of PO, which is controlled vocabulary of botanical terms describing morphological and anatomical structures representing organ, tissue and cell

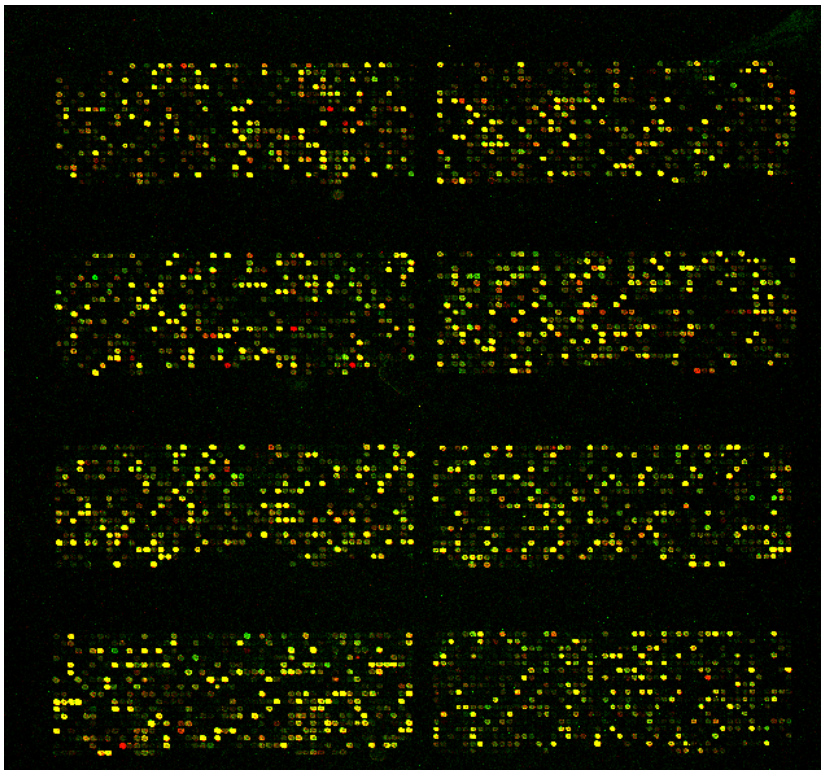


Figure 4.4: Top four grid rows of a hybridized CATMA array used in two-color mode. Each block is printed by the same print tip. (picture provided by Little, D., University of Lausanne, Swiss)

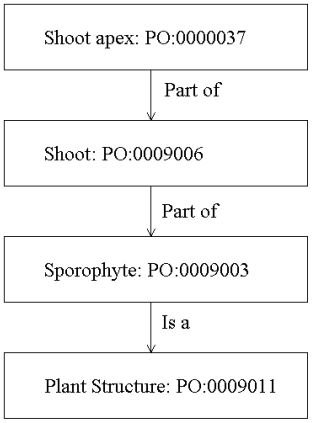


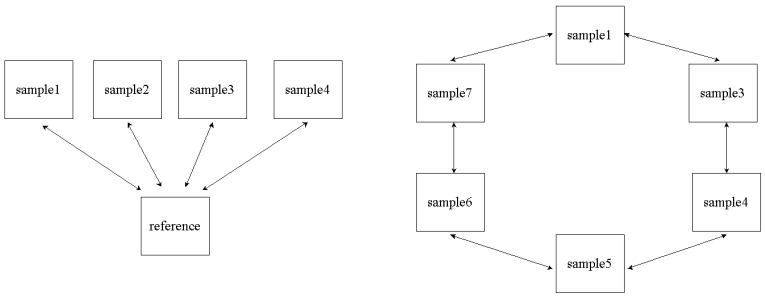
Figure 4.5: Plant Ontology term lineage for the term *shoot apex*. Typically for these ontologies, terms are linked to each other with *part-of* and *is-a* relationships

types, and their relationships. An example of plant structure from PO is

- Term: Shoot Apex
- Accession: PO:0000037
- Definition: The topmost part of the shoot, situated at the distal extremity of the shoot axis, consists of apical meristem and the youngest leaf primordia.

Additional to this information Plant Ontology gives the term lineage with *is a* and *part of* relationships between the terms as shown in Figure 4.5.

For gene expression profiling it is more useful if the tissues are collected from plants at similar developmental stages rather than from plants of similar chronological age. To describe the growth stages the Boyes scale [17] was used, it provides a finer scale than the growth stages provided by the Plant Ontology Consortium. The growth stages described by Boyes *et al.* serve both as



(a) Reference design (b) Loop design

Figure 4.6: Two popular experiment designs for two-channel microarray experiments: the reference design (a) and loop design (b). The CAGE project decided to use the reference design as design for its wild-type compendium samples.

developmental landmarks and as triggers for the collection of sample material. Table 4.1 shows the Boyes scale for *Arabidopsis thaliana* growing in soil.

4.5 CAGE experiment design

The CATMA array is a spotted array and two samples are hybridized together on the same array. Here the question arises which two samples should be hybridized against each other. Two experimental designs were considered: the loop design and the reference design, see Figure 4.6.

The reference design is the most common for two-color microarray experiments. Its advantages are that new samples can be easily added and it is robust to multiple chip failure. A strong disadvantage is that most of the data gathered by such a design comes from the sample that is of least interest: the reference sample. A good choice of reference is also required to assure that all genes of interest are expressed in the reference. Genes not present in the reference sample will give an unusable ratio of sample over reference. The loop design is more efficient as all samples hybridized are of interest and only one array is needed to compare two samples. In a simple loop design, each sample is

Table 4.1: Boyes scale of Arabidopsis Col-0, growing on soil. This scale was used to harvest CAGE samples and annotate the growth stages. The table gives the stage, a description of the stage, the average numbers of days from data of sowing and the standard deviation (in days) of this average. Table from [17]

Stage	Description	Days	sd
Principal growth stage 1	Leaf development		
1.02	2 rosette leaves >1 mm in length	12.5	1.3
1.03	3 rosette leaves >1 mm in length	15.9	1.5
1.04	4 rosette leaves >1 mm in length	16.5	1.6
1.05	5 rosette leaves >1 mm in length	17.7	1.8
1.06	6 rosette leaves >1 mm in length	18.4	1.8
1.07	7 rosette leaves >1 mm in length	19.4	2.2
1.08	8 rosette leaves >1 mm in length	20.0	2.2
1.09	9 rosette leaves >1 mm in length	21.1	2.3
1.10	10 rosette leaves >1 mm in length	21.6	2.3
1.11	11 rosette leaves >1 mm in length	22.2	2.5
1.12	12 rosette leaves >1 mm in length	23.3	2.6
1.13	13 rosette leaves >1 mm in length	24.8	3.2
1.14	14 rosette leaves >1 mm in length	25.5	2.6
Principal growth stage 3	Rosette growth		
3.20	Rosette is 20% of final size	18.9	3.0
3.50	Rosette is 50% of final size	24.0	4.1
3.70	Rosette is 70% of final size	27.4	4.1
3.90	Rosette growth complete	29.3	3.5
Principal growth stage 5	Inflorescence emergence		
5.10	First flower buds visible	26.0	3.5
Principal growth stage 6	Flower production		
6.00	First flower open	31.8	3.6
6.10	10% of flowers to be produced have opened	35.9	4.9
6.30	30% of flowers to be produced have opened	40.1	4.9
6.50	50% of flowers to be produced have opened	43.5	4.9
6.90	Flowering complete	49.4	5.8
Principal growth stage 8	Silique ripening		
8.00	First silique shattered	48.0	4.5
Principal growth stage 9	Senescence		
9.70	Senescence complete; ready for seed harvest	-	-

hybridized to two arrays. A more complex loop design is needed to make the experiment more robust against hybridization failures [124]. Although the loop design is more efficient, CAGE chose to use the reference design. The main reasons for this is that it is easier to implement and the goal of CAGE was to provide a reference data set that could be used by the community to compare their experiments to, which would be difficult to do with a loop design. After deciding on the experiment design, a suitable reference sample needed to be chosen. Taking a biological sample as reference would give the disadvantage that any gene that is not expressed in this reference cannot be used in the compendium. As the compendium covers many different developmental stages and many different conditions, finding a good biological reference is almost impossible. A study by Dudley *et al.* [28], showed that instead of using a 'real' sample as reference, labeled oligos complementary to every feature on the array can also efficiently be used as a reference. They also found that this strategy is more accurate and better in retaining the information on absolute abundance information than when using a conventional reference sample. As explained in the section on the CATMA array, all GSTs on the array have 1 of the 16 primers which were used in the second amplification round corresponding to the number of rows in a 384-well plate. A reference made of a mixture of these 16 primers would thus give a constant signal in the reference channel and is further refered to as the *CAGE oligo reference*.

4.6 Normalization

Two-color spotted arrays are usually normalized using a loess fit to remove dye bias and correct for print tip differences as explained in Chapter 2. As the CAGE project decided to use a mixture of primers that were used to amplify the GSTs as a reference instead of a common reference sample, loess normalization is not possible as one of the channels is constant. Before normalization a filtering step is applied to flag spots that are below the background intensities. The following threshold was used as a filter:

$$\text{foreground} > \text{background} + 2\sqrt{\frac{\text{var(background)}}{2} + \frac{\text{var(foreground)}}{2}}. \quad (4.1)$$

A first step in the normalization is to perform background subtraction. We subtract the median of the background from the mean foreground channel. Next we correct for primer effect using ANOVA. The need to normalize for this primer effect in the oligo reference channel is shown in an MA-plot of primer

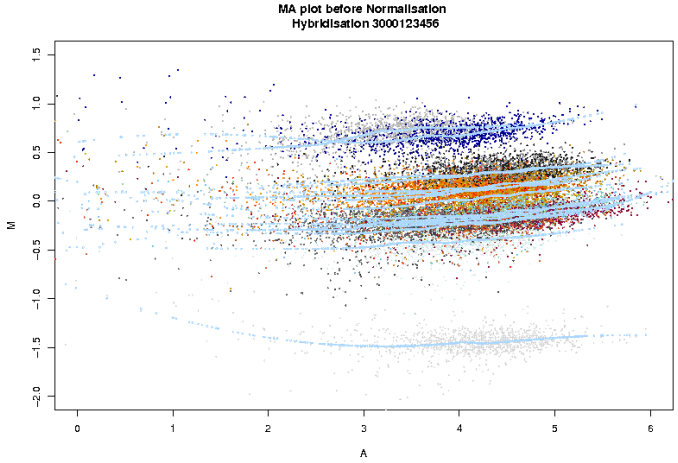


Figure 4.7: An M versus A-plot of a primer versus primer hybridization showing the need to normalize for primer effect. Each colors corresponds to probes amplified with a different primer, light blue lines show a loess fit per primer.

versus primer (Figure 4.9). In this figure the spots are colored according to the primer that was used to amplify the GST, which is also the primer from the oligo reference mix that will bind to the spot. The different primer groups can be clearly distinguished.

The ANOVA model used for correction for the factor primer (P) is

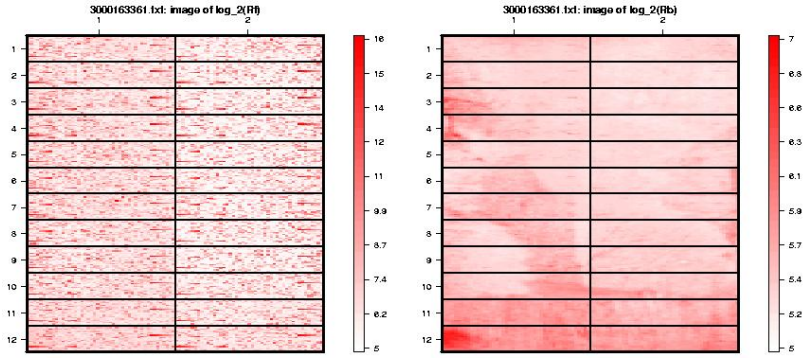
$$y_{pj} = \mu + P_p + \epsilon_{pj}, \tag{4.2}$$

where P has 16 levels corresponding to the 16 different primers. Note that this correction is only applied to the oligo reference channel.

In a second ANOVA step, both channels are corrected for print tip effect (T) using the following model:

$$y_{tj} = \mu + T_{tj} + \epsilon_{tj}. \tag{4.3}$$

The residuals of this model are our normalized expression measurements.



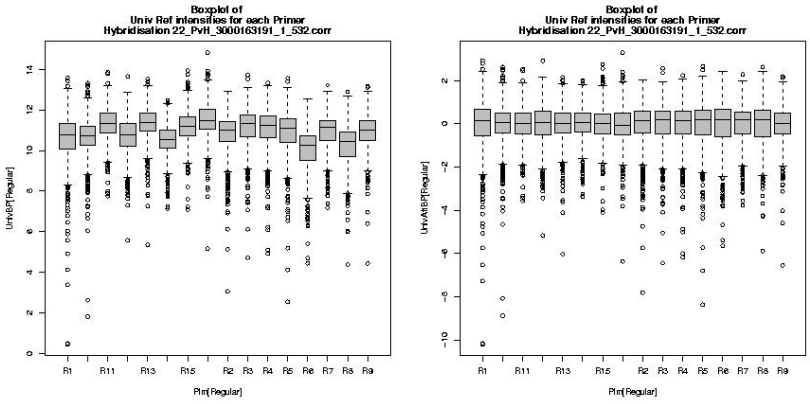
(a) Cy5 foreground intensities (b) Cy5 background intensities

Figure 4.8: Fore- and background intensities plotted according to the position of the features on array to visualize hybridization problems.

4.7 Quality assessment

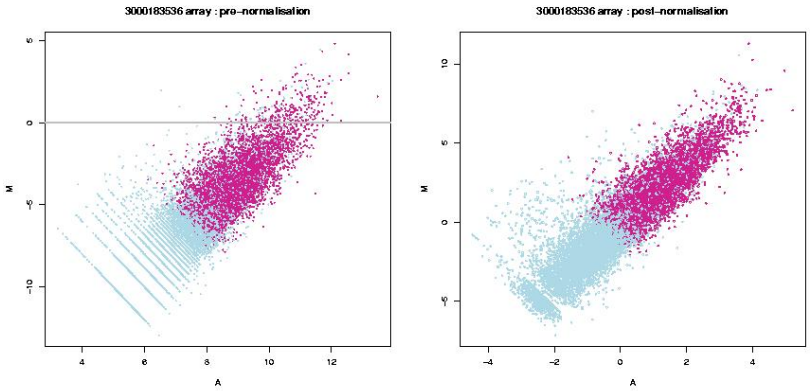
Different quality assessment measures are produced. A first measure of the hybridization quality is the percentage of features on the array that have measurements above our background threshold. A good experiment should have about 50% of the features above this threshold. Images showing the background and foreground intensities of each channel separately plotted against their xy-coordinates on the array, provide a means to detect faults during hybridization, such as strikes, dust, and fingerprints. A pair of these images is shown in Figure 4.8.

To assess the result of normalization for correction of primer and print tip effect, box-plots showing the intensities grouped per primer and per print tip before and after the ANOVA step are created. Figure 4.9 shows one of these box-plots where the effect of primer normalization can be viewed. Note the distinct pattern per primer before normalization, which is effectively removed. To assess the global effect of normalization and show the spots above and below the background threshold, MA-plots of the sample versus the reference are plotted and the features with a foreground signal above our threshold are highlighted (see Figure 4.10).



(a) Before normalization (b) After normalization

Figure 4.9: Box-plots showing the intensities grouped per primer and the effect of normalization



(a) Pre-normalization (b) Post-normalization

Figure 4.10: M versus A plots before and after normalization, features expressed above background are colored blue.

4.8 Preprocessing pipeline

To preprocess data from 4,000 microarray hybridizations an automatic high-throughput microarray preprocessing pipeline had to be build. The next sections describe the development of this pipeline.

4.8.1 CAGE data flow

The data flow we came up with for the CAGE project is depicted in Figure 4.13. CAGE data producers submit their data through the web-accessible MIAMExpress database located at EBI. Once the experiment is fully submitted the curators at EBI run a Perl script that creates a MAGE-ML document describing the submitted experiment and containing the microarray data. The data then gets curated, after which the file is placed at an FTP site accessible to ESAT. When running, the pipeline automatically detects new files placed at the EBI's FTP site, downloads them and preprocesses the experiment. The pipeline produces web pages that show the quality assessment results and updates the MAGE-ML documents with the normalized data. Then these documents are placed back at the FTP site of EBI. At that point the EBI curators will load the MAGE-ML file into ArrayExpress, from which it is accessible again to the CAGE data producers and after a waiting period is released to the public domain.

4.8.2 CAGE pipeline architecture

To perform the normalization we chose to use the open source and open development statistical programming environment R. At the time when we started to develop the pipeline the Bioconductor project had just started and provided some structures and methods we could use. Keeping in mind that all code is open source we could if necessary modify the existing functions and use them in our own pipeline. As discussed in Chapter 3, one of the first requirements was that our pipeline could handle MAGE-ML format data and have these documents as input and output. Our initial focus was thus to develop an R package that made this possible: RMAGEML [32]. Central in the preprocessing pipeline is a local MySQL database, which keeps track of processed experiments, file locations, hybridizations, created HTML pages, and processing progression. A Perl script is the main control module. When started, it will check the FTP site of EBI and compare the experiment identifiers with the ones already in our database. When new experiments are found, the script will download these and start an R script that queries our local database to check the

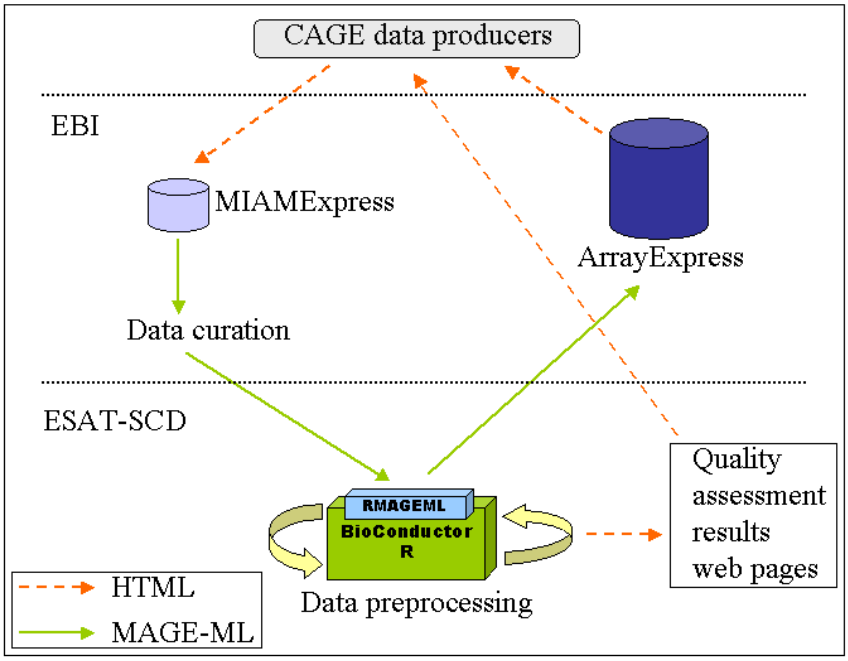


Figure 4.11: Overview of the CAGE data flow. CAGE partners submit their data to MIAMExpress at EBI. Upon complete submission of an experiment, EBI creates an experiment MAGE-ML document. The EBI data curators check completeness and validity of the MAGE-ML documents. The automatic preprocessing pipeline at ESAT-SCD detects the new MAGE-ML documents and retrieves the data via FTP. The data is preprocessed at ESAT and HTML pages are provided for the CAGE partners to assess the quality of the experiment. At the same time the MAGE-ML document is updated and placed back at EBI. The MAGE-ML document is loaded into ArrayExpress. CAGE partners can access their normalized data through ArrayExpress.

location and name of the new experiment. The R script uses the RMAGEML package to read the MAGE-ML files and fills the local database with information, such as sample names, array used, and data submitter. The R script then continues and performs the ANOVA normalization as discussed above. To do this normalization, it is essential that the data submitter is known because different partners use different scanners, which provide different quantitation types. It is also important to retrieve the array identifiers because there are so far four versions of the CATMA array, each with a different number of features. The pipeline also had to be capable of dealing with multiple CATMA versions within one experiment. After normalization the MAGE-ML documents are updated with the normalized data, again using the RMAGEML package. At the same time the R script generates HTML pages and images for quality assessment, which will be used by our preprocessing web site. Once the R script finished, the Perl script takes over again, zips the experiment and puts it back at the EBI FTP site via FTP. Then the pipeline continues with normalizing other new experiments. A summary of the CAGE preprocessing pipeline is depicted in Figure 4.12.

4.8.3 CAGE preprocessing web application

The quality assessment (QA) measures described above are made available to the CAGE partners via the CAGE preprocessing web application http://tomcat.esat.kuleuven.be/cage. This restricted access web site, build with Java servlet technology, enables to view the QA results of each CAGE hybridization individually. Hybridizations can be retrieved by slide number, data producer, experiment identifier, submission date, and tissue. The results page contains all plots described in the QA section of this chapter.

4.9 CAGE sample production

An overview of the different European data producers and the number of planned hybridizations per partner is given in Table 4.2.

The goal of CAGE was to produce the first compendium hybridizations after 6 months of the start of the project in November 2002 and within a few months

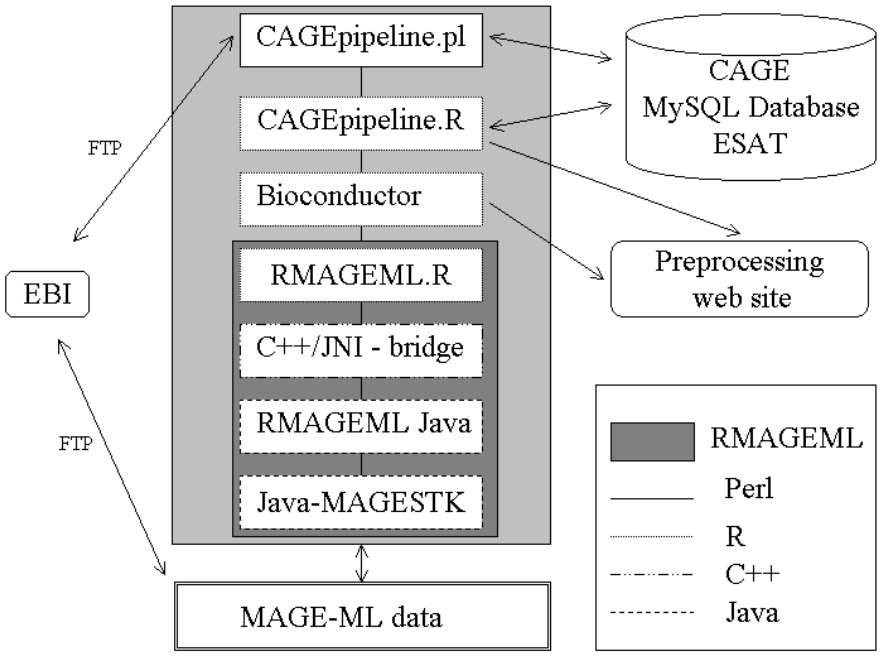


Figure 4.12: Architecture of the CAGE preprocessing pipeline. A local MySQL database keeps track of processed experiments, file locations, hybridizations, created HTML pages, and processing progression. A Perl script is the main control module, it checks the FTP site of EBI to look for new MAGE-ML data. When new experiments are found, the script will download these and start an R script that queries our local database to check the location and name of the new experiment. The R script uses the RMAGEML package to read the MAGE-ML files and fills the local database with information such as sample names and data submitter. The R script then continues and performs the ANOVA normalization as discussed before. After normalization the MAGE-ML documents are updated with the normalized data, again using the RMAGEML package. At the same time the R script generates HTML pages and images for quality assessment, which will be used by our preprocessing web site. Once the R script finishes, the Perl script takes over again, zips the experiment, and puts it back at the EBI FTP site via FTP.

Table 4.2: CAGE data producers and number of hybridizations per partner

Partner	Country	Number of Hybridizations planned
PSB	Belgium	800
UNIL	Swiss	400
MPI	Germany	400
URGV	France	400
SLU	Sweden	400
RUU	Netherlands	800
CSIC	Spain	400
HRI	UK	400

from data submission the data would be made available to the community via ArrayExpress. However the data production was slowed down because a variety of reasons for example growth chamber problems for the partner PSB. Another factor slowing down the data reaching ESAT-SCD was a lag period between the actual experiment and submission of the data to MIAMExpress. Data producing partners experienced that the web-based submission can be slow for large experiments. EBI later provided a batch upload tool that enabled faster data submission in the form of Excel sheets. A third bottleneck was to have good MAGE-ML, as frequently the MAGE-ML files we received contained errors (e.g., wrong sample names or quantitation types), preventing automated preprocessing. An overview of the CAGE sample production is given in Figure 4.13. This plot shows the number of months passed since the start of the project and the number of hybridizations that were processed by our automatic preprocessing pipeline. Note that some data was available earlier but this data did not come as well-formed valid MAGE-ML or was given to us mainly as text files.

4.10 AtGenExpress

At about the same time of CAGE another collaborative effort started to build a compendium of Arabidopsis gene expression data: AtGenExpress [96]. This consortium would use the Affymetrix ATH1 platform to produce a large gene expression compendium in about one and a half year time and is coordinated by Detlef Weigel, Thomas Altmann and Lutz Nover with funding from the German Arabidopsis Functional Genomics Network (AFGN). Initially they would generate a gene expression profiling database covering about 230 data

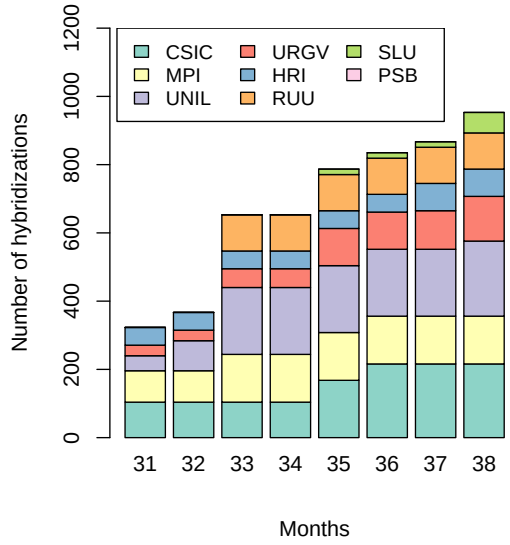


Figure 4.13: Overview of the CAGE sample production. The goal of CAGE was to start producing the first compendia data six months after the start of the project. However due to several problems, the first real compendia data was ready for preprocessing after thirty-one months.

sets derived from 500 RNA samples. This initial part includes samples for *Arabidopsis* development, stress responses, light response and pathogen interaction. It will be complemented by 80 data sets from the NSF 2010 project (USA) on pathogen responses, by 170 data sets from RIKEN (Japan) on hormone responses and by about 50 data sets from the MPIs in Tübingen and Golm. The overall database derived from more than 1,300 microarrays would be released to the public in the Gene Expression Omnibus (GEO) and ArrayExpress databases. Data processing and publication would be managed by TAIR (http://www.arabidopsis.org). The project stayed well on schedule and published their first compendium data and paper in 2004. To make the AtGenExpress data easily accessible, the data was made available via the web application GENEVESTIGATOR [127] and was deposited in ArrayExpress [85].

4.11 Discussion

The CAGE project was a European collaboration between eight data producing and two bioinformatics partners. One of the challenges that CAGE posed was the efficient data exchange between the partners and preprocessing of large scale microarray data. Within our group, an automatic preprocessing pipeline was developed that could preprocess the CAGE data in a high-throughput manner and could deal with the differences in array design and quantitation types. CAGE's aim was to provide a well-annotated compendium of *Arabidopsis thaliana* microarray data to the community. The generated data was annotated as such that it contained the minimal information (MIAME compliant) necessary for comprehension and reproducibility of the data and analysis results. This MIAME compliant data was stored and exchanged in the MAGE-ML format and the preprocessing pipeline we developed is to our knowledge the first of its kind that fully integrates the MAGE-ML format.
The CAGE partners should have started producing the first compendium data six months after the start of the project. However it took more than one and a half years before the real sample production finally started. Due to this lack of data, the pipeline was initially developed using some test experiments. In the last few months however, data production finally took off and by now around 1,000 hybridizations have been preprocessed by our automatic pipeline. This result shows that the pipeline is capable of efficiently preprocessing microarray data in a high-throughput manner. However manual intervention was needed from time to time as the microarray data submitted by the partners contained errors (e.g., switched channels), or the MAGE-ML encoding contained errors (e.g., wrong name for the oligo reference sample or quantitation types). It is

expected that such errors would be easier to detect at curation time (which takes place before preprocessing), if the MAGE-ML format was less complex. The data producing partners also delayed submitting new experimental data as they experienced submitting their data to MIAMExpress was laborious especially for large experiments. EBI solved this by providing a batch data upload tool, by which data submission could occur via Excel sheets, for many hybridizations at once. CAGE data production was hampered by problems at the sample production side (e.g., growth chambers) and array production. The combination of using a new microarray platform and at the same time producing a large compendium with this new platform, has proven to be a daunting task. Though novel, perhaps using an already established microarray platform, as did the AtGenExpress consortium, might have been a more efficient and productive strategy for the partners involved in the provided time.

Chapter 5

Integration of public data resources with biomaRt

Comprehensive analysis of microarray data coming for example from projects such as described in the previous chapter, involves integrating the gene expression measurements with a variety of biological data that can be retrieved from different public databases. A simple example is to annotate features that were found differentially expressed with corresponding gene names and genomic locations. Most public databases provide multiple ways to access their data, for example via web browsers. However, a major bioinformatics challenge that remains is how to efficiently have access to this biological data from within a microarray data analysis environment. BioMart is a generic, query oriented data management system, capable of integrating distributed data resources developed by the EBI. We first describe a number of important public biological databases, such as Ensembl and Uniprot, that implement the BioMart system. And then describe biomaRt, a software package we developed to integrate data from BioMart systems into the statistical programming environment R, enabling biological data mining. Our software package (biomaRt) was incorporated in the Bioconductor project (described in the introduction) and was published in the journal Bioinformatics [33].

Table 5.1: Organisms present in Ensembl version 36

Homo sapiens	Danio rerio
Pan troglodytes	Takifugu rubripes
Macaca mulatta	Tetraodon nigroviridis
Mus musculus	Ciona intestinalis
Rattus norvegicus	Drosophila melanogaster
Canis familiaris	Anopheles gambiae
Bos taurus	Apis mellifera
Monodelphis domestica	Caenorhabditis elegans
Gallus gallus	Saccharomyces cerevisiae
Xenopus tropicalis	

5.1 Annotation databases

To make sense of microarray data, information from public databases is frequently integrated in microarray data analysis. In recent years, biological databases have become repositories containing a huge amount of heterogeneous data. Most of these databases provide access via the web and some provide direct SQL* access or specific APIs† to query them. The next sections discuss some of these data resources and how they are used in microarray data analysis.

5.1.1 Ensembl

The completion of the human genome sequencing project and other sequencing efforts resulted in a surge of available sequence data. Dedicated databases and genome browsers have been set up that make these fully or partially sequenced genomes available to the community together with annotation information. A major such genome browser is Ensembl: a software system that produces and maintains automatic annotation on selected eukaryotic genomes [50]. At the time of writing, Ensembl (http://www.ensembl.org) covers 19 annotated genomes shown in Table 5.1.1.

*Structured Query Language

†Application Programming Interface

There are many ways were meta-data from Ensembl is used in microarray data analysis. An obvious but one of the most used applications is to annotate a variety of identifiers with the corresponding gene name and a description of the gene if available. Other use cases are for example to retrieve the position on the genome to order a set of genes according to their position or retrieve all genes located on the Y chromosome and study their expression profile.
Ensembl also stores meta-data coming from other projects, such as Gene Ontology (GO) and OMIM[‡]. The Gene Ontology consortium provides structured, controlled vocabularies covering several domains of molecular and cellular biology that can be used to annotate genes, gene products or sequences [23]. These vocabularies are structured in classification supporting 'is-a' and 'part-of' relations. GO consists of three such vocabularies that describe gene products, according to associated biological processes, molecular functions, and cellular components. The molecular function of a gene product describes activities of this gene product, for example '*microtubule motor activity*' or '*antioxidant activity*'. Biological process describes the result where molecular functions lead to, for example, '*growth*'. Cellular component indicates where the gene product resides in the cell and within molecular complexes. An example of an cellular component is '*cytoplasmic vesicle*'. GO identifiers are associated to gene products by the Gene Ontology Annotation database [20]. Ensembl includes these GO annotations as well as their descriptions in its database. In microarray analysis GO annotation is frequently used to look for over-represented GO terms in a list of differentially expressed genes or just to see which GO terms are in this set.
Another useful annotation in Ensembl are OMIM identifiers. OMIM stands for Online Mendelian Inheritance in Man and is a catalog for human genes and genetic disorders developed by NCBI. Ensembl has mapped this information to its own database and makes possible to retrieve genes implicated in inheritable diseases, by querying genes associated with the corresponding OMIM term for the disease. In a microarray analysis this would enable one to focus on all genes known to be associated with a certain disease or look for overrepresentation of OMIM terms in a set of differentially expressed genes.
The ability to calculate and display integrated comparative genomics resources is an important part of Ensembl [14]. Different than the use cases described above are queries that take advantage of the multiple species present in Ensembl. One can imagine an experiment done in mouse but we would be interested in the human genes corresponding to the genes found differentially expressed in mouse.

[‡]Online Mendelian Inheritance in Man

5.1.2 Human variation data

Genomes display natural polymorphisms, which are variations in the genome sequence. Two copies of the human genome differ in about 1 base pair per 1000 bases. Most of these variations are Single Nucleotide polymorphisms (SNPs), where one nucleotide is changed into another. Other sources of the variation is attributable to deletions and insertions [59], and copy number polymorphisms [100].
dbSNP is a database located at the NCBI, and stores this variation data. Ensembl also mirrors the dbSNP data. A second resource of variation data has been delivered by the HapMap project [60] of which Phase I was completed in November 2005. The HapMap project aims at determining common patterns in human genome variation by characterizing sequence variants and their frequencies in human populations. The HapMap database (http://www.hapmap.org) contains more than one million single nucleotide polymorphisms, which were obtained by studying 269 DNA samples from four populations: 90 individuals from the Yoruba population in Nigeria, 90 individuals from Utah (USA), 45 Han Chinese individuals and 44 Japanese individuals [60]. This database provides a unique data collection to find associations with the reported SNPs and for example gene expression differences. A recent microarray study by Stranger *et al.* already showed associations between differential gene expression and SNPs discovered in the HapMap project [110].

5.1.3 VEGA

The Vertebrate Genome Annotation database (VEGA, http://vega.sanger.ac.uk/index.html) stores high quality manual annotations of finished vertebrate genomes [6] and thus differs from Ensembl, which shows computationally derived gene predictions on a large number of genomes. At the time of writing the VEGA database contained four species: *Homo sapiens*, *Mus musculus*, *Danio rerio* and *Canis familiaris*. Manually annotated information that is available is standard information about the gene such as name, description, location, and cDNA length. Other information includes OMIM identifiers, HUGO[§] identifiers, and InterPro[¶] protein domains.

5.1.4 UniProt

The UniProt database (http://www.ebi.uniprot.org/index.shtml) [121] is the most annotated protein sequence database [20]. The database provides a resource

[§]Human Genome Organization
[¶]InterPro is a database of protein domains.

for protein sequences and functional annotation. The annotations provided by uniprot include protein name and function, protein domains, taxonomy, and post-translational modifications.

5.1.5 Wormbase

Wormbase (http://www.wormbase.org) is a database dedicated to the model species *Caenorhabditis elegans* [99]. Ensembl also covers *C. elegans* but wormbase contains much more data on this organism. In addition to gene-centric queries, Wormbase supports querying over-expression patterns, RNAi phenotypes, mutant phenotypes, genome variations, and literature citations.

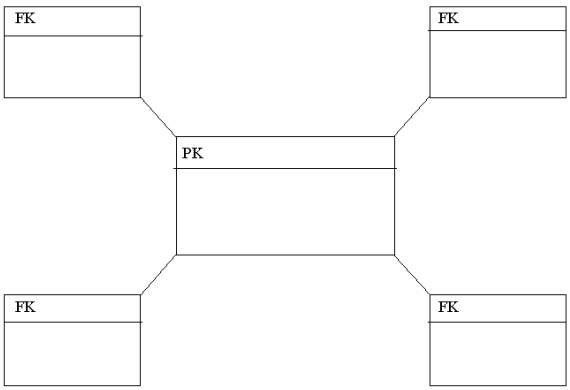
5.1.6 Gramene

The Gramene database (http://www.gramene.org) is a resource for comparative genomics of grasses [118] and currently contains data on *Zea Mays* and *Oryza Sativa*. Additional to the grasses, Gramene also stores information on the plant model species *Arabidopsis*. Examples of data present in Gramene is Gene Ontology and Plant Ontology associations, genome sequences, Trait Ontology terms (an ontology describing associated traits of QTL), and phenotypic descriptions. Gramene also contains the Catma identifiers, which would make it an ideal resource to provide metadata in CAGE data analysis.

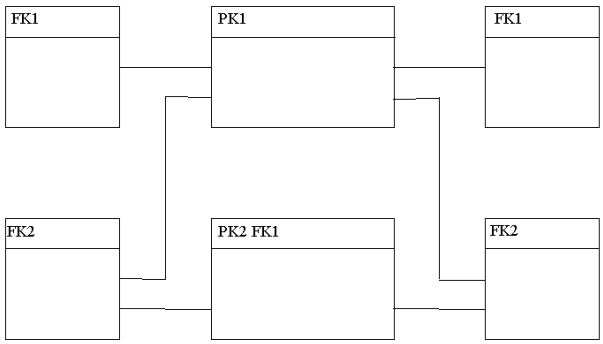
5.2 BioMart

BioMart (http://www.biomart.org) is a joint project between the European Bioinformatics Institute (EBI) and Cold Spring Harbor Laboratory (CSHL) and aims to develop a generic, query oriented data management system, capable of integrating distributed data resources. BioMart systems have a three-tier architecture. A first tier consists of one or multiple relational databases. Central to these BioMart database systems is the concept of the star and the reverse-star schemas (see Figure 5.2), of which the former consist of a single main table linked to different dimension tables and the latter is a variant [68]. The overall simplicity of these schemas avoids complex joins and enables fast data retrieval.

The second tier consists of two APIs, one written in Java and the other in Perl. The third tier consists of query interfaces. BioMart comes with web-based query interfaces (MartView), a stand-alone query interface (MartExplorer) and



(a) Star schema



(b) Reverse star schema

Figure 5.1: BioMart simple database schemas avoid complex joins. The star schema consists of just one main table containing the primary key (PK) and has a number of dimension tables where each tuple contains a foreign key (FK) which references the primary key of the main table. The reverse star schema is a little bit more complex and contains two main tables each with their own primary key and associated foreign keys in the dimension tables. On top of a primary key, one of the main tables also contains for each tuple a foreign key referencing to the other main table. Figure modified from http: //www.biomart.org

a shell-like interface (MartShell). In addition to this, the BioMart system comes with the MartEditor tool to edit the database configuration. This configuration is stored as XML within the database making meta-data (a description of the database, including the tables and fields) available to third-party applications. Ensembl was one of the first databases to deploy a full-featured BioMart for Ensembl in Spring 2005 [14, 68]. Since then more databases have implemented a BioMart version and by now all of the databases described in the previous sections are accessible via a BioMart implementation.

5.3 biomaRt

With the development of the biomaRt package we wanted to take advantage of the fast query capabilities in BioMart systems and the presence of major databases in a uniform system, to make a large collection of data available in R, an environment ideal for biological data mining. The *biomaRt* package is an add-on package for R that provides query ability to BioMart databases. Before going into the details of biomaRt, a brief summary of annotation packages available in Bioconductor is given.

5.3.1 Annotation and Bioconductor

Before our development of biomaRt, there were 3 annotation packages available in the Bioconductor project: *annotate*, *AnnBuilder*, and *annaffy*. They use either precompiled annotation tables derived from NCBI and stored as hash tables in R [126] or retrieve information from NCBI over the web. Precompiled annotation packages are convenient when working with one or a few array types with relative constant designs, however this approach has limitations. When multiple chip designs are used in for example a meta-analysis study, different metadata packages need to be installed, which will contain redundant information. Large gene sets make the metadata packages sizable and the number of features present on microarrays is growing fast. The data contained in these packages is rather limited comapred to what is available in public databases.

5.3.2 biomaRt

The first release of our package covered the following BioMart databases: Ensembl [50], a software system which produces and maintains automatic annotation on selected eukaryotic genomes; VEGA [6], the manually annotated

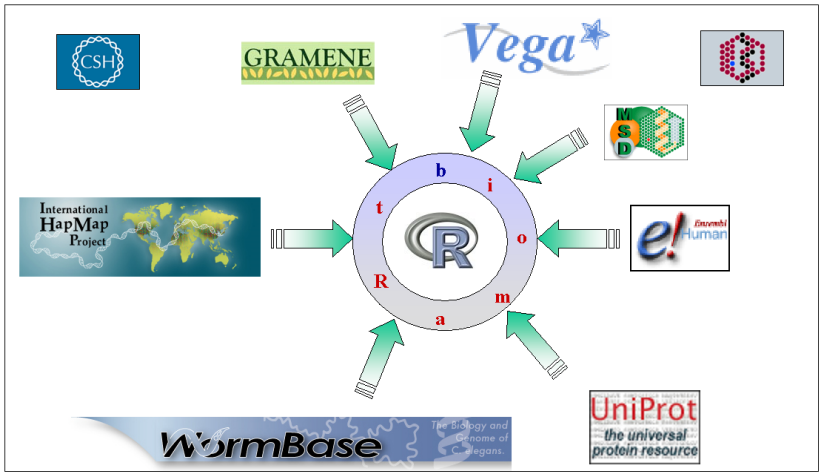


Figure 5.2: Integrating BioMart databases and R. The idea of the biomaRt software package is to efficiently integrate large volumes of biological data from public databases that employ a BioMart data management system in the statistical environment R. Examples of BioMart databases are Ensembl, Uniprot and Wormbase.

Vertebrate Genome Annotation; dbSNP [102], the Single Nucleotide Polymorphism database of NCBI, and sequence mart, containing the Ensembl genome sequences. The package depends on the R package *RMySQL* and has been tested on Windows, Linux and OSX. After loading the library one can connect to either public BioMart databases or local installations of these. biomaRt offers several functions that enable querying these databases. One set of functions can be used to annotate identifiers, such as Affymetrix, RefSeq, or Entrez Gene, with information such as gene symbol, chromosomal coordinates, OMIM, and GO. Alternatively one can use a gene symbol as starting point and query for corresponding Affymetrix identifiers on a given chip. The queries can also have an inter-species nature and one can use an identifier of one type in species *a* to look up identifiers of the same or another type, corresponding to homologs in species *b*. A second set of functions allow sequence related data retrieval. Given a species and chromosome coordinates, one can retrieve genome sequences. This way a user can go directly from a set of differentially expressed genes to the upstream promoter sequences. Similarly SNP information can be retrieved. The SNP information is derived from dbSNP, which is mapped to Ensembl.

biomaRt provides documentation in the form of manual pages for every function and a vignette, which is an interactive document containing executable code chunks giving a more problem-oriented style of help.

5.3.3 Simple biomaRt functions

There are many functions implemented in biomaRt to retrieve standard data that are frequently used in microarray analysis. Instead of going through each function separately, their use is shown with an example. Below is an example script made by one of the biomaRt users (from http://folk.uio.no/mortema/master.R) to show how to use most of the simple biomaRt query functions in an R session.

```
# Very useful Biomart functions
library(biomaRt)
mart <- martConnect( biomarts = c("ensembl","vega"))
#list available AffyArrays at mart
getAffyArrays(mart = mart)
#example using affy id from Ensembl
g = getGene( id = "1939_at", array = "hg_u95av2", mart = mart)

#example using locuslink id from vega
g = getGene( id = 100, type = "entrezgene", species = "hsapiens",
 db = "vega", mart = mart)

#list available species from ensembl & vega
species =getSpecies(mart = mart)
show(species)

#retrieves SNPs
mart <- martConnect(biomarts=c("ensembl","snp"))
snp = getSNP(chromosome = 1, start = 30065068, end = 30104323,
 species ="hsapiens", mart = mart)

#OMIM
omim = getOMIM( id = 672, type = "entrezgene", mart = mart)
omim = getOMIM( id = "1939_at", array = "hg_u95av2",
       mart = mart)

# get Homologs
#HUGO to Entrez Gene
homolog = getHomolog(id = 1:20, from.species = 'hsapiens',
       to.species = 'mmusculus', from.type = 'entrezgene',
       to.type = 'refseq', mart = mart)
show(homolog)
#ensembl to ensembl
homolog = getHomolog( id = "ENSG00000072778",
 from.species = "hsapiens",from.type = "ensembl",
```

```
 to.type="ensembl",to.species="mmusculus", mart = mart)

show(homolog)
#Affy to Affy
homolog = getHomolog( id = "1424184_at", to.array = "hg_u95av2",
     from.array = "mouse430a_2", mart = mart )
show(homolog)
#Affy to ensembl
homolog = getHomolog( id = "1424184_at", to.species = "hsapiens",
   to.type = "ensembl", from.array = "mouse430a_2", mart = mart )
show(homolog)

#example using affy id
go = getGO( id = "1939_at", array = "hg_u95av2", mart = mart)
#example using locuslink id
go = getGO( id = 672, type = "entrezgene", species = "hsapiens",
     mart = mart)

# get FASTA sequences
mart <- martConnect()
martConnect(biomarts='sequence',mart=mart,
host='ensembldb.ensembl.org',user='anonymous',password='')
seq<-getSequence(chromosome=c(2,2),start=c(100000,30000),
end=c(100300,30500),species="hsapiens",mart=mart)
exportFASTA(seq,file="test.fasta")

martDisconnect(mart = mart)
```

This example script first shows how to extract information about a gene starting from Affymetrix identifiers in Ensembl or starting from EntrezGene identifiers in VEGA. Then the script shows how to retrieve SNP information of all SNPs recorded between certain positions on the chromosome from a selected genome. The script continues with showing how to retrieve OMIM information starting from EntrezGene or Affymetrix identifiers. A next section in the script does a variety of homology mappings between mouse and human using different types of identifiers. Then it is shown how to retrieve GO information and at last it is shown how to retrieve genomic sequences using the getSequence function. Another example that could be solved using the simple biomaRt functions is to sort different genes based on their chromosome coordinates, this could be used to investigate if co-localized genes are also co-expressed.

A last example could be a microarray analysis in *Drosophila*, where we want to focus on genes that have human homologs known to be involved in a certain disease. biomaRt enables one to first look up the human homologs, then using these homologs to query for OMIM identifiers. *Drosophila* genes that have

human homologs with an OMIM identifier associated with them, can then be selected for subsequent analysis.

5.3.4 Advanced biomaRt functions and data mining

Beside the simple biomaRt functions described above, biomaRt has a set of advanced functions that allow the users to retrieve any data field from the BioMart databases that are available. These functions are based on the other BioMart APIs, such as *Martshell* (a command line query interface) and work on all BioMart systems that have MySQL access enabled or an active web service. A general theme in these advanced queries is that the query consists of three distinct features (see Figure 5.3):

- Attributes: this is the part of the query that defines the information that the user wants to retrieve. For example we want to retrieve the chromosome and band.
- Filters: this is a set of filters that need to be applied to make the selection. For example we use EntrezGene identifiers as filter.
- Values: these are the allowed values for the filters. For example a list of EntrezGene identifiers when *entrezgene* was set as filter.

A first step in an analysis using the advanced functions is to select a BioMart database to use. The function *listMarts*, returns all available BioMart databases. After selecting a BioMart database, the user has to select a dataset from the selected BioMart using the *useDataset* function. An example of a data set is *hsapiens_gene_ensembl* in Ensembl. Now the *listAttributes* and *listFilters* functions can be used to display the available attributes and filters for the selected data set. For *Homo Sapiens* in Ensembl there are currently more than 100 attributes and filters to choose from. In a last step the actual query is performed using the *getBM* function. Examples of the use of these advanced functions can be found in Appendix C.

5.4 Discussion

The Bioconductor package *biomaRt* enables direct access from Bioconductor to BioMart databases such as Ensembl, creating a strong alliance for data analysis with biological databases. The current annotation packages available from Bioconductor are complementary to our package. They use precompiled

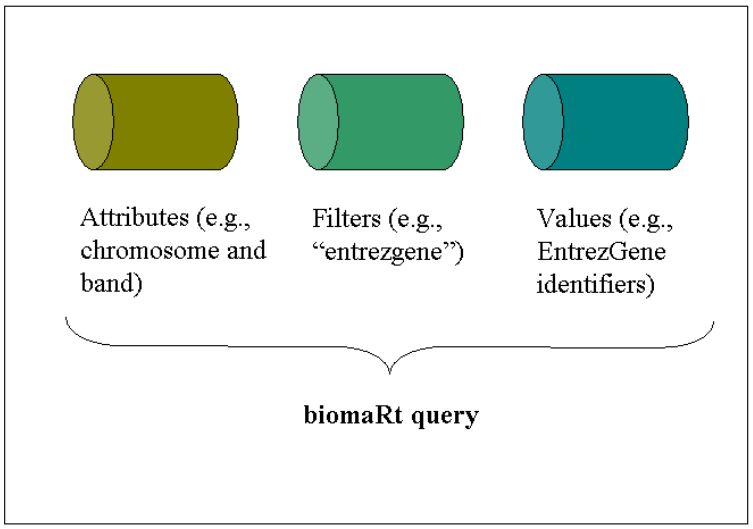


Figure 5.3: Composition of a BioMart query. The query consists of three elemenst: (1) a set of attributes, which are the features that one wants to retrieve for example *chromosme name*, (2) a set of filters which makes the application retrieve the features that pass these filters for example EntrezGene identifiers, (3) a set of values for the selected filter.

annotation tables derived from NCBI and stored as hash tables in R [126]. Precompiled annotation packages are convenient when working with one or a few array types with relative constant designs, however this approach has limitations. When multiple chip designs are used in for example a meta-analysis study, different metadata packages need to be installed and will contain redundant information. Very large gene sets make the metadata packages sizable, while with *biomaRt*, only the annotation of the genes of interest is retrieved. *biomaRt* is more scalable as it gathers up-to-date information from BioMart databases. Fast data retrieval is possible as the *biomaRt* package executes direct SQL queries from R to the BioMart databases. Besides annotation information *biomaRt* also enables mapping of homologs and retrieval of sequence and SNP data that can become part of a microarray data analysis. The biomaRt package will be further developed to include more BioMart databases and allow more complex types of queries. This tight integration of large public databases with data analysis in R provides a powerful platform for biological data mining. This work was published in the journal *Bioinformatics* in 2005 [33].

Chapter 6

Contributions to microarray data analysis tools

There are many software tools available to analyze microarray data. A major web application for microarray data analysis is Expression Profiler developed by the European Bioinformatics Institute. An important aspect of microarray data analysis is having highly interactive and flexible visualizations. We developed new visualization modules for Expression Profiler, work which was published in Nucleic Acids Research [67]. Microarray data analysis using Expression Profiler relates to the CAGE project (described in Chapter 3) as there is a tight link between ArrayExpress, the microarray data repository which holds the CAGE data, and Expression Profiler. The result of a microarray data analysis is typically a list of gene names which is usually too large to biologically validate each gene in the list. Endeavour is a tool developed by ESAT-SCD for gene prioritization, and orders gene lists according to their relevance to process that one is studying by using known biological information, such as the information described in the previous chapter or data from other microarray experiments. A second contribution to software analysis tools was the development of an Arabidopsis specific version of Endeavour, which again links to CAGE as the CAGE microarray data can be included in this tool.

6.1 Expression Profiler: next generation

Expression Profiler (EP) is a web-based platform for microarray gene expression and other functional genomics related data analysis. Expression Profiler: next generation (EP:NG) (http://www.ebi.ac.uk/microarray-srv/EP/cgi-bin/ep_ui.pl) modularizes the original design of EP and allows individual analysis-task-related components to be developed by different groups, work together, and share a common user interface [67]. A typcial workflow in EP:NG is displayed in Figure 6.1. The user either uploads data from an external source or exports the data from ArrayExpress for analysis in EP:NG. The Data Selection component provides a basic statistical overview of the uploaded dataset. The Transformation component can be used to impute missing values in the data and to perform a transformation, such as taking the log. Next, the user can chose a variety of methods for further analysis. For example, the user can explore the structure of the data by using the clustering methods available in the Hierachical and K-groups Clustering components. Alternatively the users who are interested in studying a particular gene or a group of genes can use the Similarity Search and Signature Algorithm [56] components to investigate the data and look for genes that have similar expression profiles over all conditions or are co-expressed in only a subset of conditions.

6.1.1 Data visualization in Expression Profiler

At one point during the development of this modular platform, EP:NG had a new user interface but still relied on the old Expression Profiler for visualizing the analysis results and a project was started to end this dependence. The results web pages produced by the old Expression Profiler were not integrated into the rest of the user interface and allowed limited interactivity. Our aim was to get an integrated highly interactive and flexible analysis results web page as output from every analysis in EP. Different options were considered, such as the proprietary format Flash from Macromedia and a new upcoming W3C Recommendation graphics format: SVG http://www.w3.org/TR/SVG/.

6.1.2 Scalable Vector Graphics

Web tools usually show the analysis results using bitmap graphics, such as JPEG, PNG, and GIF. These bitmaps are in essence two-dimensional arrays of colored pixels. In contrast to this, vector-based graphics usually contain a binary description of how to draw the image. In SVG this description is held in SVG elements, which are XML, the final display in the browser however is

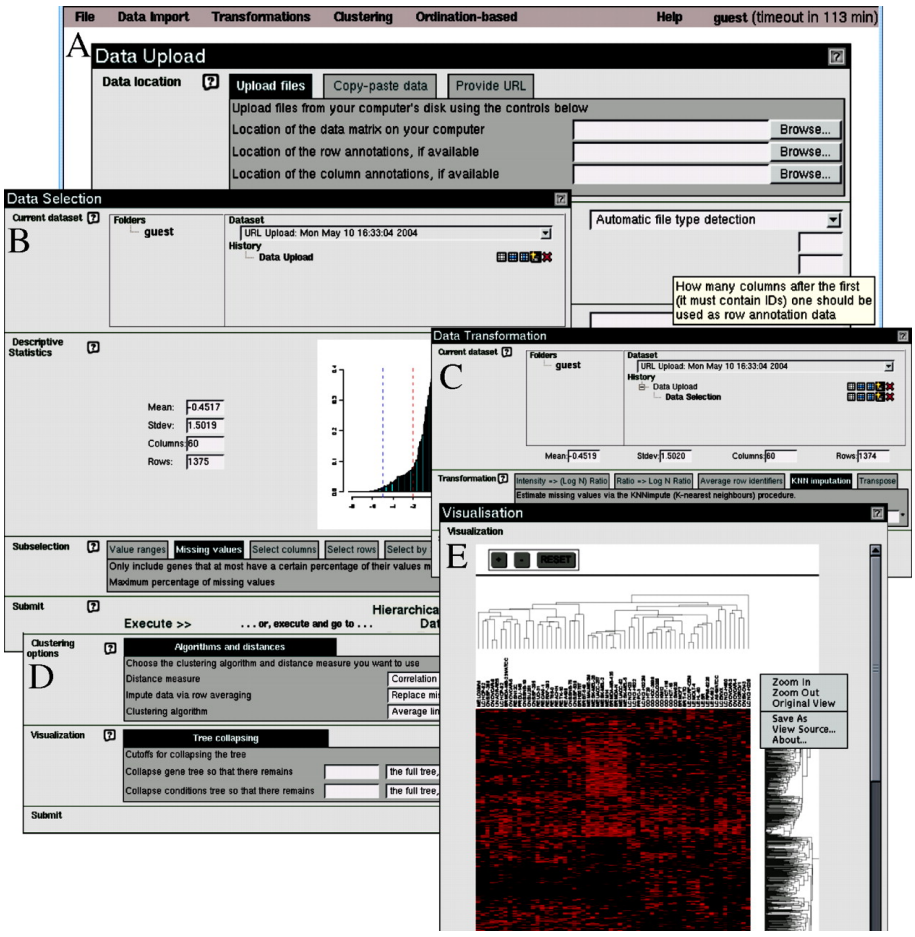


Figure 6.1: Expression Profiler: Next Generation (EP:NG), a web-based platform for microarray gene expression and other functional genomics related data analysis. External data can be uploaded or data selected from the ArrayExpress repository can be exported into EP:NG. After data upload the Data Selection component shows a basic statistical overview of the data. Then the Transformation component can be used to transform the data or impute missing values. After these initial steps, different components provide a variety of data analysis methods such as hierachical clustering, similarity search and the signature algorithm. One of our contributed visualization modules is shown at the top level and contains a hierachical clustering result which includes a gene and sample tree.

rendered to a bitmap. This rendering happens dynamically at the client side, as the browser displays the page. An advantage of this approach is that it is possible to adjust certain parameters of the display to the client's choice. One such parameter important in visualization of data analysis results is zooming. In contrast to ordinary bitmap graphics,where zooming ends in blurry figures, zooming SVG graphics keeps the graphics sharp. Another advantage over bitmap graphics is that SVG is XML, text in graphics remains text and is searchable and selectable. Although SVG graphics was at the time still in full development and SVG support by the main browsers limited, its advantages made us chose to develop new visualization modules that produce SVG images.

6.1.3 Implementing SVG in EP:NG

The EP:NG visualization module was implemented as a Perl module that could be used by the other Perl modules of EP:NG. Functions were written to draw interactive line plots (see Figure 6.2), heatmaps and associated trees of hierarchical clustering of samples and genes. Flexible visualization web pages were created by considering the output as a grid of which each element can contain a visualization or not. A heatmap with a hierarchical clustering of the genes for example consists of a grid with one row and two columns, the first column containing the tree and the second column containing the heatmap.

6.1.4 Current status EP:NG visualization

The EP:NG graphics has been further developed by Misha Kapushesky (EBI) and the interface became highly interactive. By now main browsers such as Internet Explorer and Firefox support the SVG format natively or can make use of plug-ins to enable SVG visualization.

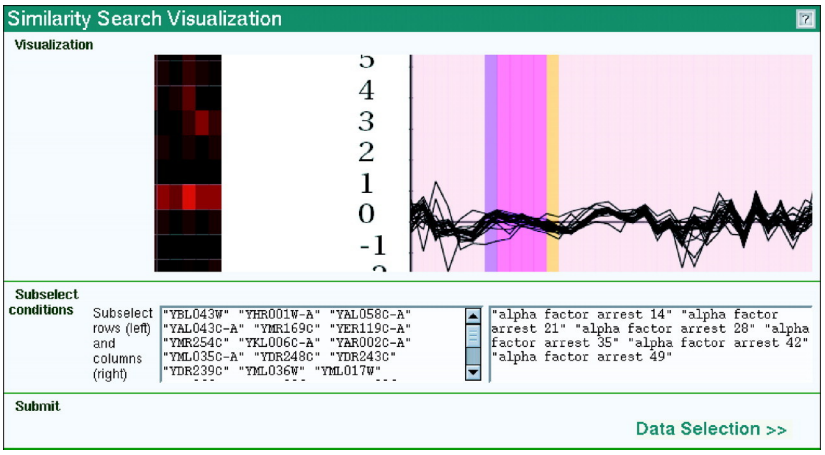


Figure 6.2: SVG: interactive line plots combined with a heatmap. As an example of interactivity, regions of the lineplot can be selected, which changes the color of the plot segment and the corresponding hybridization names appear in the text box.

6.2 At-Endeavour

The result of a microarray data analysis often consists of a list of differentially expressed genes. The number of genes in this list however is most of the times several times larger than the number of genes that can be validated and followed-up by wet-lab experiments. An ordering of the gene lists according to most interesting to least interesting genes to do further analysis on is thus useful. Our group at ESAT-SCD has developed a tool *Endeavour* [1] to prioritize a list of genes based on another set of genes that are known to be involved in the process one is studying.

6.2.1 Endeavour

Endeavour[1] is a software application developed at ESAT-SCD for the computational prioritization of *test* genes, based on a set of *training genes*. Prioritization methods can be regarded as measuring the similarity of a gene with a certain biological feature, such as a disease, or measuring the similarity of a gene with genes involved in this disease. In Endeavour similarity between genes in terms of vector based information sources, such as microarray data are calculated using the Pearson correlation coefficient:

$$r(x,y) = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2 \sum_{i=1}^{n}(y_i - \bar{y})^2}}. \tag{6.1}$$

For other vector based information such as text information the cosine similarity is used:

$$\text{sim}_{\cos}(x,y) = \frac{\sum_{i=1}^{m} x_i y_i}{\sqrt{\sum_{i=1}^{m} x_i^2 \sum_{i=1}^{m} y_i^2}}. \tag{6.2}$$

Similarity based on annotation information from sources, such as GO and InterPro is measured by a p-value, which is calculated using a binomial statistic that represents the over-representation of a certain identifier in the training set. The scoring by each of the submodels results in N different rankings r_1, r_2, ..., r_N and these can be combined into a final ranking using order statistics[1].

6.2.2 At-Endeavour

In this thesis we developed an *Arabidopsis* specific version of Endeavour, called *At-endeavour*. As this was a deliverable of the CAGE project our main focus was to integrate CAGE microarray data and enable annotation of CATMA and AGI identifiers, which are *Arabidopsis* specific. Because of the lack of high quality CAGE data we used data from the AtGenExpress project, which we normalized using GCRMA. The AtGenExpress microarray datasets available in At-endeavour are

- Normal *Arabidopsis* development serie, covering 237 hybridizations
- Root and shoot time-course of oxidative stress response
- Root and shoot time-course of UVB stress response
- Root and shoot time-course of cold stress response
- Root and shoot time-course of drought stress response
- Root and shoot time-course of osmotic stress response
- Root and shoot time-course of salt stress response

Beside these microarray experiments, facilities have been build in to include microarray experiments from CAGE, which can be selected by their experiment identifier.

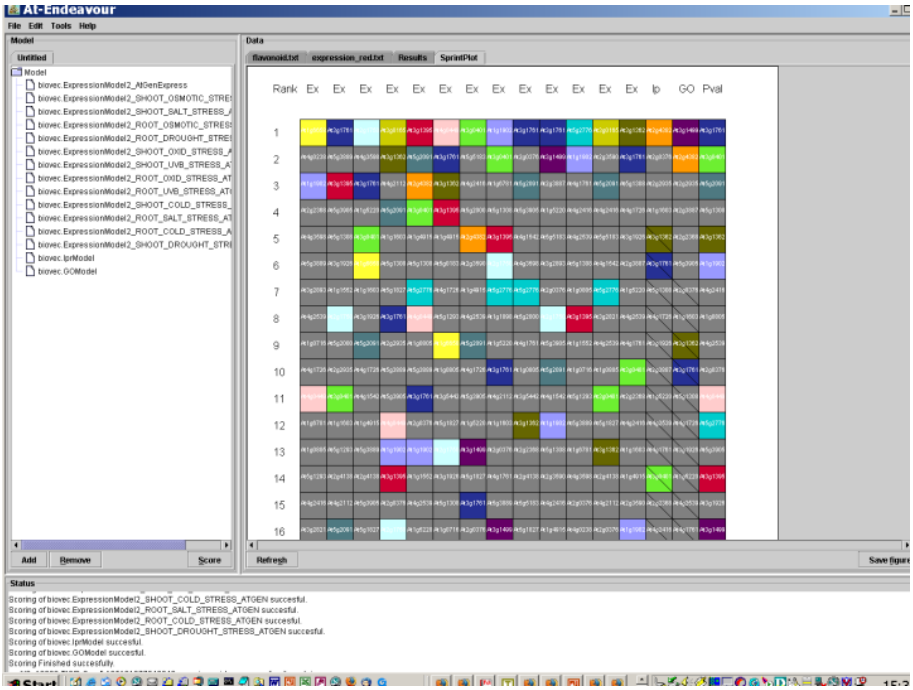


Figure 6.3: At-Endeavour: an *Arabidopsis thaliana* specific version of the gene prioritization tool Endeavour

GO and InterPro models are also available. The annotation data is retrieved from a local Gramene (see Chapter 5) database. Figure 6.3 shows a screenshot of the application displaying the result of a set of test genes based on a set of training genes.

6.3 Discussion

In this chapter we described two contributions to microarray data analysis tools. Expression Profiler Next Generation is a widely used web application for the analysis of microarray data that is tightly integrated with the microarray data sets available from ArrayExpress. We developed a new flexible and interactive visualization module capable of drawing popular graphics used in microarray analysis, such as hierarchical trees, heatmaps, and lineplots. A second contribution was the porting of the Endeavour tool for gene prioritization to *Arabidopsis thaliana*. This was a deliverable of CAGE and can be used by

the *Arabidopsis* community.

Chapter 7

Discussion and conclusion

7.1 Achievements

The achievements of this thesis can be grouped into the following areas: microarray data exchange, high-throughput microarray data preprocessing, integration of biological data in a microarray analysis environment, development of tools for microarray analysis and novel findings in gene expression studies. The preprocessing of microarray data generated by the compendium of *Arabidopsis* gene expression (CAGE) project, was the major project of this Ph.D. and provided the ultimate setting for the development of high-throughput data exchange, preprocessing and analysis tools.

Microarray data exchange

Current microarray studies generate a wealth of data and one microarray analysis usually uses only part of the generated information. Subsequent analysis by researchers other than the data producers becomes common practice. For researchers to use and replicate microarray data generated by others, the MIAME standard was set up to describe the minimal information needed for microarray experiments to be published. Together with this standard, a data exchange format MAGE-ML was developed to efficiently exchange this highly complex data. As the CAGE project wanted to produce a well-annotated MIAME compliant microarray compendium, MAGE-ML was chosen as the data exchange format. However at the start of this thesis, no tools were available to import and export MAGE-ML format microarray data from a microarray data analysis environment. A first contribution of this thesis was the development of an add-on software package (*RMAGEML*) for the statistical programming environment

R, that enabled use of this data format in a popular data analysis environment. This work was done in collaboration with Joke Allemeersch and Dr. V. Carey (Channing lab, Harvard) and published in *Bioinformatics* [32]. *RMAGEML* is also a core component of the CAGE preprocessing pipeline and is now part of Bioconductor, a popular software project for genomic data analysis in R.

CAGE and high-throughput microarray data preprocessing

The Compendium of *Arabidopsis* Gene Expression (CAGE) project was the main project of this Ph.D. The project would produce a microarray data set measuring the expression of about 25,000 genes in 4,000 microarray hybridizations, covering 1,000 different conditions. Our group was responsible for preprocessing the microarray data produced in this project and in a later phase to perform analyses on the compendium data. CAGE makes use of a newly developed microarray platform CATMA. We first compared this novel platform to already existing platforms for gene expression studies in *Arabidopsis thaliana* and found that the platform performed at least as well as the available commercial microarray platforms for *Arabidopsis* [4]. The CAGE project produced microarray data in a high-throuput manner and preprocessing these thousands of hybridisations was a true challenge. Our group developed an automated preprocessing pipeline by integrating different programming environments. The system had to be optimized from time to time, for example, to deal with new array designs and variations in MAGE-ML coding. The preprocessing results and microarray experiment quality assessments were made available through a web application, which was also developed during this thesis. According to the initial planning, CAGE would start to produce data six months after the start of the project but initial data production was slow. In the last few months however, data production finally took off and by now around 1,000 hybridizations have been preprocessed by our automatic pipeline. This result shows that the pipeline is capable of efficiently preprocessing microarray data in a high-throughput manner. However manual intervention was needed from time to time as the microarray data submitted by the partners contained errors (e.g., switched channels), or the MAGE-ML encoding contained errors (e.g., wrong name for the oligo reference sample or quantitation types). It is expected that such errors would be easier to detect at curation time (which takes place before preprocessing), when the MAGE-ML format would be less complex. The data producing partners also delayed submitting new experimental data as they experienced submitting their data to MIAMExpress was laborious especially for large experiments. EBI solved this by providing a batch data upload tool, by which data submission could occur via Excel sheets, for many hybridizations

at once. CAGE data production was hampered by problems at the sample production side (e.g., growth chambers) and array production. The combination of using a new microarray platform and at the same time producing a large compendium with this new platform, has proven to be a daunting task. Though novel, perhaps using an already established microarray platform, as did the AtGenExpress consortium, might have been a more efficient and productive strategy for the partners involved in the provided time.

Integration of biological databases and microarray data analysis

Comprehensive microarray data analysis involves integration of a variety of biological data together with the microarray data. Today many different resources are available each containing a specific set of biological information. Examples of these databases are Ensembl and Wormbase. The information stored in these resources is vast and integrating this biological data into a statistical programming environment was a challenge addressed in this Ph.D. We developed an R package, biomaRt, that enabled integration of this biological data in a data analysis environment R, opening the way to biological data mining. Our package became part of the Bioconductor project and has gained a significant number of users since its introduction. This work was done in collaboration with Dr. A. Kasprzyk, Dr. W. Huber and Dr. A. Brazma from EBI and Dr. S. Davis at NIH and our paper was published in the journal *Bioinformatics* [33].

Contributions to microarray data analysis tools

Preprocessed CAGE microarray data is made available to the community via the ArrayExpress microarray repository. Further data analysis is possible via Expression Profiler, a web based tool for microarray data analysis that is tightly linked to ArrayExpress. Flexible and highly interactive visualizations form an important aspect of data analysis and the development of such software modules was a challenge addressed in this Ph.D. We developed highly interactive and flexible visualization modules for Expression Profiler using SVG graphics. This work was published in *Nucleic Acids Research* [67]. As second contribution to data analysis tools was the porting of the Endeavour tool to *Arabidopsis thaliana* as a CAGE deliverable.

7.2 Future directions

Since the release of MAGE-ML as data exchange format, major microarray groups that make up the MGED society, have been pushing this format forward to become the standard microarray exchange format. The lack of software tools to handle these documents and the fact that of the major microarray repositories only ArrayExpress makes available MAGE-ML documents, shows that this standard is too complex and requires a grand effort to support. The MIAME standards however have been adopted by the major microarray databases (GEO, CIBEX and ArrayExpress). Future research in microarray data exchange should focus on making the MAGE-ML format less complex, so more bioinformaticians will develop tools that can handle the format, which will lead to a wider use and the general acceptance as a microarray exchange standard.

At the current rate of microarray data production, high-throughput preprocessing data and analysis has become a necessity. Robust automatic pipelines can only be built when the data is well annotated in a format that is usable for automation. The MAGE-ML format provided a good basis for this exchange but the documents are quite complex. Because of this complexity not many data analysis applications currently support MAGE-ML and the wealth of information available remains unused. A slimmed down version of MAGE-ML would make development of these applications easier and consolidate the use of MAGE-ML in all microarray repositories. The less complex MAGE model should allow for direct access to sample names and identifiers of the features on the array. There also should be a clear consensus of how MIAME data is coded into MAGE-ML. We experienced the current MAGE-ML too flexible as different encodings are possible so for a software application it is not always straightforward to know where to get the needed information. As a simpler version of MAGE is being developed, our RMAGEML package should follow adaptations to the standard.

Our CAGE pipeline will be used to process all the CAGE data that will still be submitted to MIAMExpress. This pipeline is to our knowledge one of the first that fully supports the import of raw microarray data as MAGE-ML and can update these documents with the normalized data it produced. This might by used as an example for other pipelines that are in development elsewhere.

Integration of other biological data together with a microarray data analysis is necessary to get as much as possible out of the data and to better understand the changes in gene expression. Examples are integration of SNP data and miRNA expression data. BioMart systems are very suitable to store a wide

variety of data and it can be foreseen that many databases will provide a BioMart version of their database in the near future. Our biomaRt package should be general enough to seamlessly integrate these new databases and make their data available for data mining in a powerful data analysis environment.
Both packages (RMAGEML and biomaRt), are part of the Bioconductor project, an open source software project for the analysis and comprehension of genomic data. The package biomaRt, has since its release gained a sizable user group. It can be expected that once the MAGE-ML standard is adopted by other databases such as GEO, that the user group of *RMAGEML* will grow as well.
The Endeavour application described in Chapter 6, uses biological data from public databases such as Ensembl (Endeavour) and Gramene (At-endeavour) to prioritize a list of genes according their similarity to a set of training genes. In the program code, retrieval of this information is currently hard coded and each time a new version of Ensembl is released the code needs to be changed in order to deal with changes in table and field names of the database. Use of the properties of BioMart databases which contain a description of its tables, fields, and data contained within the database, would make the Endeavour applications more robust to new database releases and make inclusion of new species from other BioMart databases feasible. This could be done by implementing the BioMart Java API or a by tighter integration with R and the use of our biomaRt package. Endeavour would also be a useful module in itself to add to Expression Profiler or to be available as a Bioconductor package. In both applications, Endeavour could use biomaRt for its data retrieval. Because of the increasing availability of large biological data sets, the development of software and automated procedures as discussed in this thesis are timely and will remain an exciting area in bioinformatics for years to come.

Nederlandse samenvatting

Microarray compendia en implicaties voor bioinformatica software ontwikkeling

Genexpressie

Het centraal dogma van de biologie stelt dat DNA* door het proces van transcriptie wordt omgezet in messenger RNA† (mRNA) en dat dit messenger RNA door de ribosomen vertaald wordt in eiwitten. De term genexpressie verwijst naar ditzelfde proces, het aanmaken van eiwitten via transcriptie en translatie. De cel heeft verschillende manieren om de expressie van genen te reguleren. De voornaamste manier is regulatie door middel van transcriptie factoren, dat zijn eiwitten die binden op specifieke motieven in de DNA sequentie en de expressie van het gen positief of negatief beïnvloeden. Naast deze transcriptiefactoren zijn er nog tal van andere gekende processen zoals: histone modificatie [73], translationele controle en DNA methylatie [13].
Genexpressie regulatie speelt een grote rol in verscheidene biologische processen zoals ontwikkeling, aanpassing van de cel aan een wijzigende omgeving en in het behouden van een normale celtoestand. Een belangrijke toepassing van genexpressie profielen is bijvoorbeeld in de prognose voor kankerpatiënten [116].

*deoxyribonucleinezuur
†ribonucleinezuur

Microarrays

De microarray technologie is een techniek voor het meten van genexpressie en is in staat dit voor duizenden genen tegelijkertijd te doen [95]. Hiervoor maakt de techniek gebruik van het feit dat complementaire DNA strengen heel specifiek met elkaar gaan binden, dit proces wordt *hybridisatie* genoemd. Door de gelijktijdige meeting van de expressie status van duizenden genen, produceren microarrays een grote hoeveelheid gegevens. Deze moeten genormaliseerd worden om de ware expressie meeting te bepalen door te corrigeren voor ruisfactoren. Initieel bevatten microarray experimenten slechts een kleine hoeveelheid hybridisaties, maar al vrij snel werden de experimenten groter. Op dit moment werken vele groepen op het bepalen van de genexpressiestatus in een groot aantal delen van het organisme [105, 111], in verschillende condities en ontwikkelingstadia [96]. Deze grootschalige metingen worden microarray compendia genoemd [53]. De transitie naar deze grootschalige microarray experimenten bracht nieuwe uitdagingen voor de bioinformatica zoals efficiente gegevensuitwisseling en automatische data verwerking. Deze uitdagingen werden in dit doctoraat behandeld.

Uitwisselen van microarray gegevens

Microarray gegevens zijn complex en enkel zinvol als alle factoren die in het experiment betrokken zijn, gekend zijn. Om gepubliceerde microarray data bruikbaar te maken voor andere onderzoekers en toe te laten dat anderen de resultaten onafhankelijk kunnen bevestigen, werd de MIAME richtlijn opgericht [18, 108]. Deze richtlijn legde een aantal minimumregels op de informatie die gekend moet zijn over een microarray experiment opdat die bruikbaar zou zijn voor anderen. Deze regels zeggen wel niet in welk formaat extra informatie moet uitgewisseld worden. Als reactie hierop ontwikkelde de MGED organisatie (http://www.mged.org) een microarray object model en een uitwisselingsformaat: MAGE.
MAGE bestaat uit een object model, een uitwisselingsformaat (MAGE-ML) en software componenten om met dit data formaat te werken. ArrayExpress [87] is één van de eerste microarray databanken die dit model volledig ondersteunde en is ontwikkeld door het Europees Bioinformatica Instituut (EBI), in Cambridge UK. Ondertussen bevat ArrayExpress duizenden hybridisaties in dit data formaat. Ondanks dit grote aantal was er tot voor kort geen enkel soft-

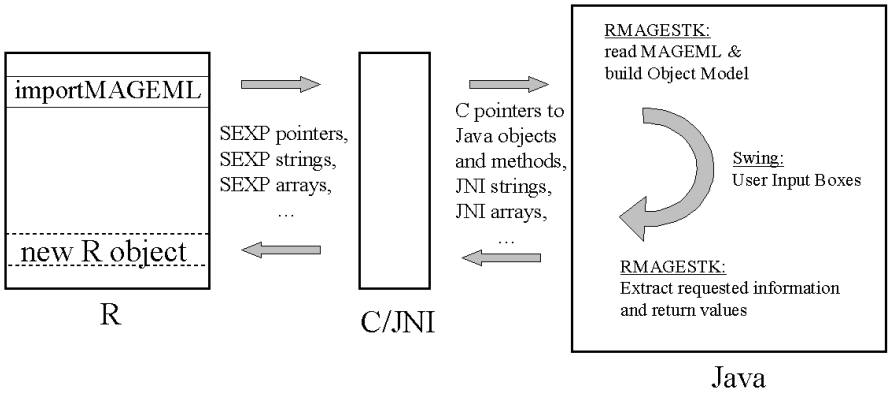


Figure 1: RMAGEML: R - C/JNI - Java interface. Een *light-weight* C/JNI brug maakt communicatie tussen de R programmeer omgeving en de Java Virtuele Machiene mogelijk. Gebruikers kunnen in R *RMAGEML* functies oproepen voor het aanmaken van R objecten met informatie komende van de MAGE-ML documenten. Aan de Java zijde, maken verschillende functies mogelijk om gevens uit de MAGE-ML documenten te halen en gegevens aan deze documenten toe te voegen.

ware pakket beschikbaar dat met deze complexe data structuren kon werken in een analyse omgeving. Dat was de aanleiding voor ons om een software pakket te schrijven dat toelaat met deze data te werken in een voor microarray populaire softwareomgeving: R. Ons pakket (*RMAGEML*) laat toe MAGE-ML bestanden te importeren, en na gegevensverwerking de nieuwe informatie aan deze bestanden toe te voegen en hiervan terug MAGE-ML bestanden te maken. De software maakt gebruik van de door MGED ter beschikking gestelde MAGEstk toolkit. Voor integratie van deze toolkit werken we met een *three-tier* architectuur, welke 3 programmeertalen omvat: R, C++ en Java (zie Figuur 1).

Bij publicatie [32] van ons RMAGEML softwarepakket, werd dit pakket toegevoegd aan het, voor microarray analyse, populaire Bioconductor project [43] (http://www.bioconductor.org).

Compendium van *Arabidopsis thaliana* genexpressie data

Arabidopsis thaliana is een model species voor het plantenrijk. In 2002 startte een samenwerking tussen acht data producerende labo's en twee computationele partners om een compendium van microarray gegevens voor deze plant te maken (CAGE). Deze samenwerking zou over drie jaar lopen en was een EU framework V demonstratie project. Terzelfdertijd als het opstellen van dit compendium zou de bruikbaarheid van een nieuw type microarray (CATMA array [49]) aangetoond worden.
In een benchmark paper door Allemeersch *et al.* [4] werd de performantie van de CATMA array vergeleken met die van commerciële microarrays voor *Arabidiosis* en uit deze studie bleek dat de CATMA array zeker even performant was als de commerciële arrays. CAGE had als doel het bouwen van een compendium met de CATMA array als microarray platform. Onze groep was verantwoordelijk voor de verwerking van deze grote hoeveelheid data en in een latere fase ook voor de initiële analyse ervan.
Voor de aanmaak van het compendium werd besloten met een *reference design* te werken. In deze opzet is één van de stalen het eigenlijke staal waarvan we de expressie willen meten en is het andere staal een referentie die in elke hybridisatie meegenomen wordt. De elementen gespot op de CATMA array hebben de eigenschap allen door één van een set van 16 primers geamplificeerd te zijn. Er werd gekozen voor een gelabelde mix van deze 16 primers als referentie sample (*oligo reference*) .
De verwerking van de bekomen metingen verloopt in verschillende stappen. Vooreerst wordt een filtering van de data toegepast, alle elementen van de array die een voorgrondmeting hebben boven een bepaalde waarde worden als tot expressie komende beschouwd. De minimale voorgrondwaarde moet hiervoor voldoen aan volgende vergelijking:

$$\text{voorgrond} > \text{achtergrond} + 2\sqrt{\frac{\text{var(achtergrond)}}{2} + \frac{\text{var(voorgrond)}}{2}}. \quad (7.1)$$

Gemiddeld hebben ongeveer 50% van de elementen op een microarray een expressiemeting boven deze waarde. Na filtering volgen een aantal normalisatie stappen om te corrigeren voor primer effect en print tip effect. De nood om te corrigeren voor primer effect wordt geïllustreerd in figuur 7.2.

Het ANOVA model dat we gebruiken voor het corrigeren voor de factor primer (P) is gegeven door

$$y_{pj} = \mu + P_p + \epsilon_{pj}, \quad (7.2)$$

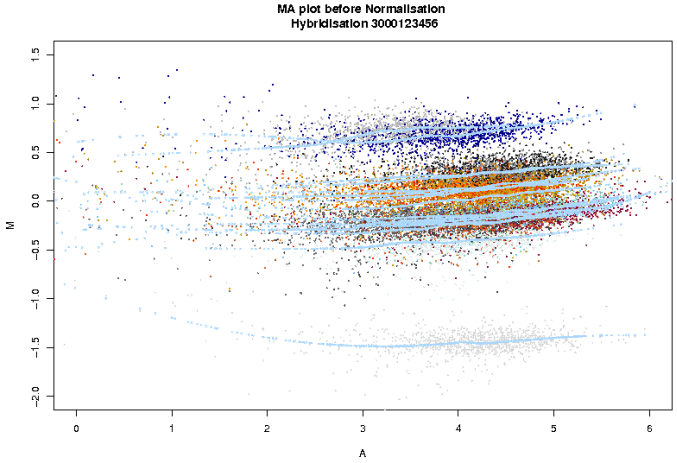


Figure 2: Een MA-plot van een primer versus primer hybridisatie. Elementen zijn gekleurd per primer, de blauwe lijnen zijn lowess lijnen per primer. Deze plot toont duidelijk de aanwezigheid van een primer effect, waarvoor we moeten corrigeren in de normalisatie.

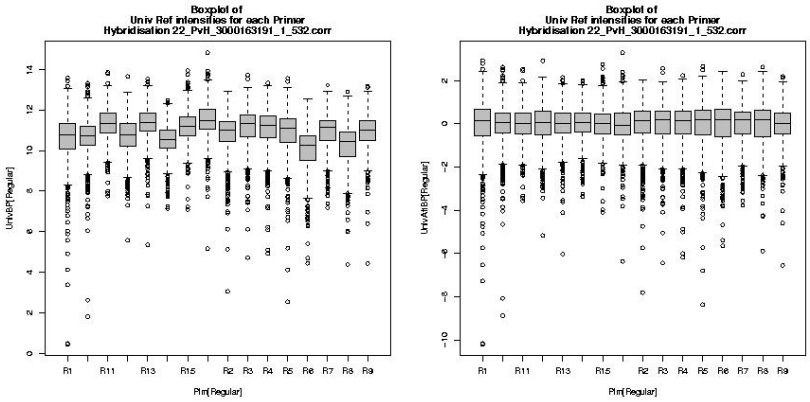
hierbij heeft P 16 niveaus overeenkomstig met de 16 verschillende primers. Deze normalisatiestap wordt enkel toegepast op het *oligo reference* kanaal.

In een tweede ANOVA stap worden beide kanalen gecorrigeerd voor het print tip effect(T) door gebruik te maken van het volgende model

$$y_{tj} = \mu + T_{tj} + \epsilon_{tj}. \tag{7.3}$$

De residu's van dit model zijn onze genormaliseerde expressiewaarden. Figuur 3 toont de intensiteiten gegroepeerd per primer voor en na normalisatie.

Gezien CAGE 4000 hybridisaties zou produceren diende er een automatische pijplijn ontwikkeld te worden voor het efficiënt verwerken van deze grote hoeveelheid gegevens. Het gegevensuitwisselingsformaat in CAGE was de hierboven beschreven MAGE-ML. Het RMAGEML pakket was één van de bouwstenen in de ontwikkeling van de automatische pipelijn. Figuur 4 beschrijft de gegevensuitwisseling tussen de CAGE partners, EBI en ESAT-SCD. Figuur 5 beschrijft de automatische gegevensverwerking pijplijn die we ontwikkelden.



(a) Voor normalisatie (b) Na normalisatie

Figure 3: Box plots van de intensiteiten gegroepeerd per primer, voor en na normalisatie.

Table 1: CAGE data productie en aantal te produceren stalen per partner

Partner	Land	Aantal geplande hybridisaties
PSB	Belgie	800
UNIL	Zwitserland	400
MPI	Duitsland	400
URGV	Frankrijk	400
SLU	Zweden	400
RUU	Nederland	800
CSIC	Spanje	400
HRI	UK	400

Het CAGE project streefde ernaar de eerste compendium data te produceren vanaf de zesde maand na de start van het project en met enkele maanden vertraging zou de data ook publiek toegankelijk worden via ArrayExpress. Tabel 1 toont het aantal hybridisaties dat per partner zou geproduceerd worden. Door problemen zoals o.a. groeikamerproblemen, kwam de echte CAGE data echter pas na 30 maanden toe op ESAT. De pijplijn kon al eerder uitgetest worden doordat er enkele test gegevens al vroeger beschikbaar waren. Figuur 6 toont het aantal door onze pipelijn ontvangen en verwerkte hybridisaties versus het aantal maanden verlopen na de start van het project.

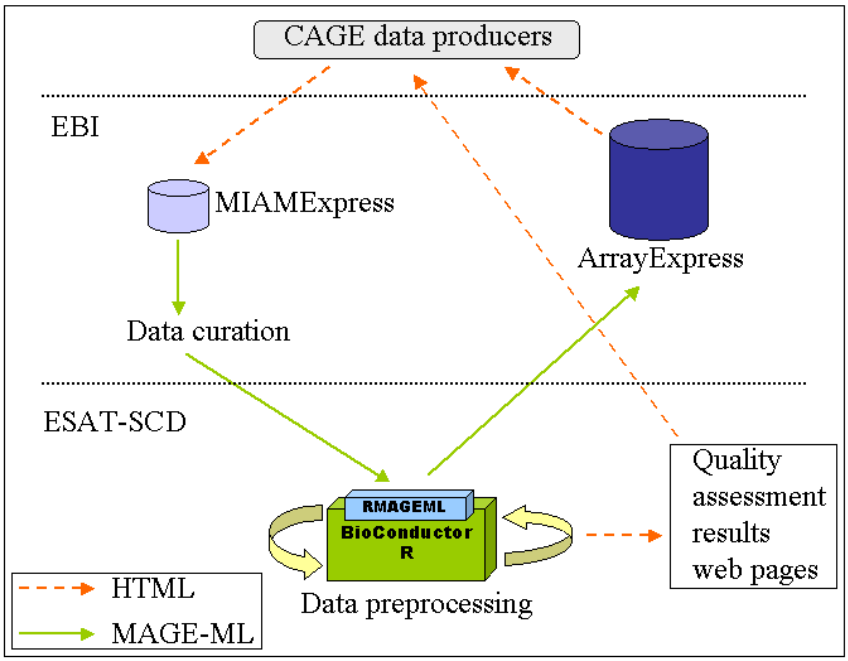


Figure 4: Overzicht van de gegevensuitwisseling in CAGE. CAGE partners geven de gegevens van nieuwe experimenten aan EBI door de webapplicatie MIAMExpress. Na vervollediging van het invoeren van deze informatie maakt EBI MAGE-ML bestanden die alle ingevoerde gegevens bevatten. Curatoren van EBI controleren vervolgens de compleetheid en geldigheid van de ingevoerde data. Onze automatische pijplijn controleert of er nieuwe MAGE-ML gegevens zijn op de EBI site en plaatst een lokale kopij op ESAT-SCD door middel van FTP. Na verwerking van de gegevens op ESAT worden normalisatie gegevens toegevoegd aan de MAGE-ML bestanden en plaatst de pijplijn de data terug op de EBI site via FTP. De pijplijn genereert ook web pagina's die de kwaliteit van de experimenten beschrijven. De genormaliseerde MAGE-ML documenten worden vervolgens op EBI in de ArrayExpress databank geplaatst vanwaar de CAGE partners terug toegang hebben tot hun gegevens.

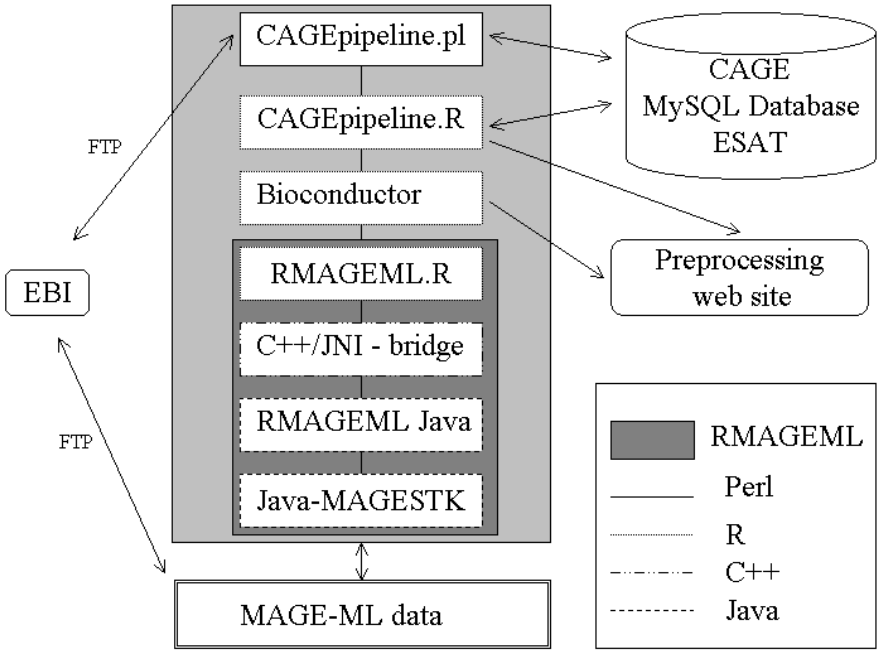


Figure 5: Architectuur van de CAGE gegevensverwerking pijplijn. Een lokale MySQL databank houdt onder andere bij welke experimenten al verwerkt zijn, waar de data files zich bevinden, de namen van de hybridisaties, de aangemaakte HTML pagina's en vooruitgang van verwerking per experiment. Een Perl script is de hoofdcontrole module. Deze module gaat eerst na of er nieuwe CAGE experimenten (in MAGE-ML gegevensformaat) op de EBI FTP site geplaatst zijn. Wanneer nieuwe experimenten gevonden zijn, zal het script deze downloaden en een R script wordt gestart. Dit R script raadpleegt de databank om de locatie en naam van het nieuwe experiment te vinden. Hierna gebruikt het R script het RMAGEML softwarepakket dat de MAGE-ML bestanden leest en bijkomstige informatie over het experiment zoals namen van de stalen en informatie over de gegevensproducent. Het R script voert hierna een ANOVA normalisatie uit zoals hierboven beschreven. Na normalisatie worden de nieuwe genormaliseerde gegevens toegevoegd aan het MAGE-ML bestand door opnieuw gebruik te maken van het RMAGEML pakket. Terzelfdertijd genereert het R script HTML pagina's met de resultaten van de kwaliteitsbepaling, dewelke daarna voor de partners beschikbaar zijn via een webapplicatie. In een laatste stap van de verwerking neemt het Perl script terug over en plaats de nieuwe genormaliseerde bestanden terug op de FTP site van EBI.

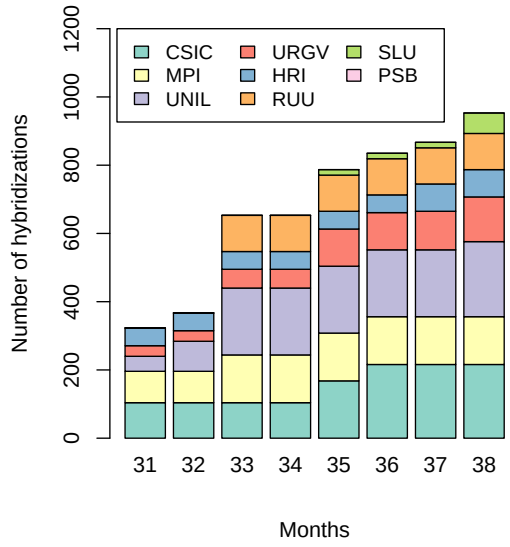


Figure 6: Overzicht van de gegevensproductie door de biologische partners in CAGE in functie van het aantal maanden verlopen na de start van het project.

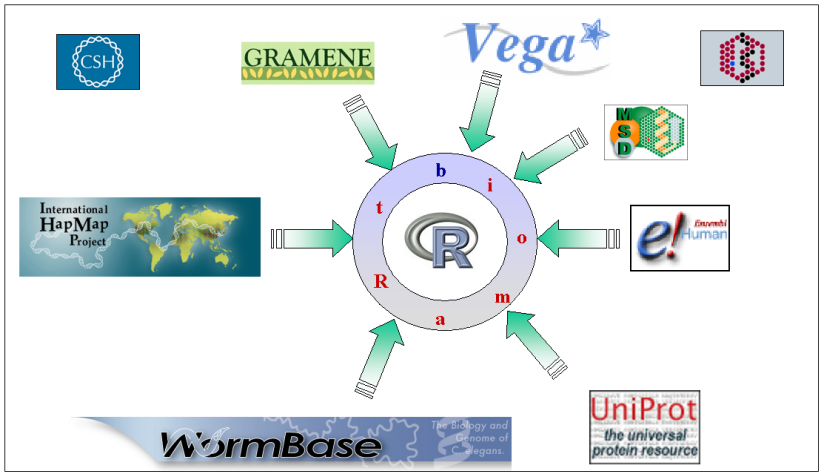


Figure 7: Integratie van BioMart databanken in R. Het doel van het biomaRt softwarepakket is het efficient integreren van grote volumes gegevens beschikbaar via publieke databanken, die een BioMart gegevensbeheer systeem gebruiken. Voorbeelden van zulke databanken zijn Ensembl en Wormbase.

Integratie van biologische gegevensbronnen en gegevens analyse

In de laatste jaren is er een grote hoeveelheid publieke biologische data beschikbaar gekomen. De grote toename in genomen waarvan de sequentie bepaald is, speelde hierbij een grote rol. Integratie en gebruik van deze data in microarray analyse is heel belangrijk. Een recent ontwikkeld gegevensbeheersysteem is BioMart [68]. Sinds de start van het BioMart project hebben al enkele grote databanken een BioMart systeem van hun data ter beschikking gesteld. De bekendste van deze databanken zijn Ensembl [50], Wormbase [99] en UniProt [121]. Deze BioMart systemen zijn voorzien van verscheidene data *retrieval* APIs maar geen van hen is een analyse omgeving. Ons doel was gebruik te maken van deze uniforme systemen en de biologische data die ze bevatten, te integreren in een populaire statistische omgeving: R [55](zie Figuur 7). Deze integratie zou leiden tot een ultiem platform voor biologische data mining. Hiervoor ontwikkelden we biomaRt, een Bioconductor softwarepakket voor integratie van biologische gegevens komende van BioMart systemen.

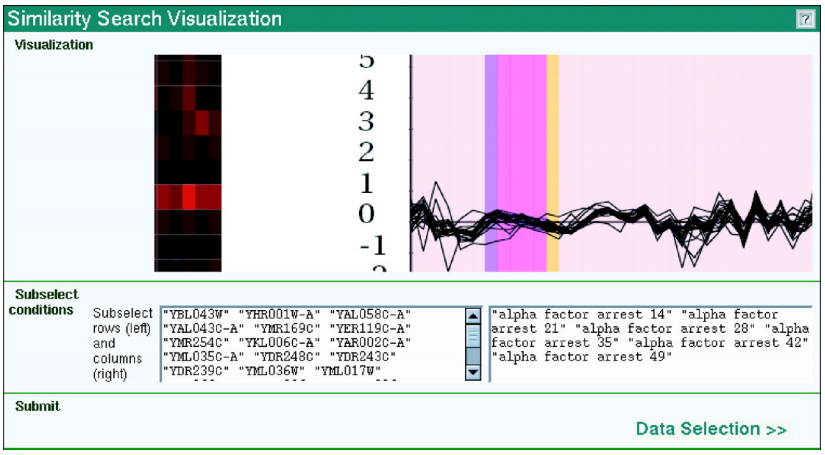


Figure 8: SVG: interactieve plot en heatmap. Als voorbeeld van interactiviteit, kunnen regio's van de lijnplot geselecteerd worden, waardoor deze gebieden van kleur veranderen en de informatie corresponderende met elk geselecteerde segment zichtbaar wordt in het tekstvak.

Bijdrage aan microarray data analyse software

In dit doctoraat werden ook enkele bijdragen geleverd aan microarray analyse software naast R. Vooreerst werd een flexibele en hoog interactieve visualisatie module ontwikkeld voor het web gebaseerde tool: Expression Profiler next generation (EP:NG). De visualisatie module werd ontworpen in Perl en maakt voor visualisatie gebruik van *Scalable Vector Graphics* (SVG). Figuur 8 toont een screenshot van één van de componenten.

In een tweede bijdrage werd een *Arabidopsis* specifieke versie van het Endeavour tool voor gen-priorisatie gemaakt. Dit was een deliverable voor het CAGE project en laat toe de CAGE microarray data te gebruiken in het ordenen van een lijst van genen. Door gebrek aan goede CAGE gegevens, werden in eerste instantie microarray gegevens van AtGenExpress[‡] [96] gebruikt.

[‡]AtGenExpress is net zoals CAGE een project dat een compendium van genexpressie data voor *Arabidopsis* wou maken.

Table 2: Gekende en mogelijke Wnt-doelwit genen.

Gekende Wnt-doelwit genen	mogelijke Wnt-doelwit genen
Dickkopf-1	HoxD1
Sp1-like XSPR-2	Iriquois 3
Axin 2	Brachyury and Tbx related protein (BRAT)
Brachyury	Galectin-3
Cyclin D1	Crab2
BAMBI	EphA2
Msx-2	Insulin-like growth factor
Frizzled-7	Ets-2a
RAR γ	Pax-3
Wnt-1 responsive Cdc42 homolog	Msx-1

Voorbeeld studie, zoeken naar Wnt-doelwit genen

Als voorbeeld werd in deze thesis een microarray studie[§] besproken, met als doel het vinden van nieuwe directe Wnt-doelwitgenen [26]. In deze microarray studie vonden we naast de gekende Wnt-doelwit genen ook een set van nieuwe mogelijke Wnt-doelwitgenen. Tabel 2 toont de reeds gekende en nieuwe doelwitgenen die in deze analyse als significant gevonden waren. Figuur 9 toont dezelfde genen op een M- versus A-plot. De differentiele expressie van de nieuwe kandidaat doelwitgenen werd door middel van qPCR bevestigd.

Conclusie

In deze thesis werd een automatische preprocessing pijplijn ontwikkeld voor het EU CAGE project. Vooreerst werd de performantie van een nieuw type array (CATMA array) uitgetest. De gegevensuitwisseling in dit project was het MAGE-ML formaat. Door de ontwikkeling van het RMAGEML softwarepakket [32] droegen we bij aan het implementeren van deze standaard in een statistische omgeving. De ontwikkeling van een tweede softwarepakket, biomaRt [33], had als doel biologische gegevens, verspreid in verscheidene databanken te integreren in een statistische omgeving en op deze wijze de weg naar

[§]Een samenwerking met Denayer *et al.* 2005 [26]

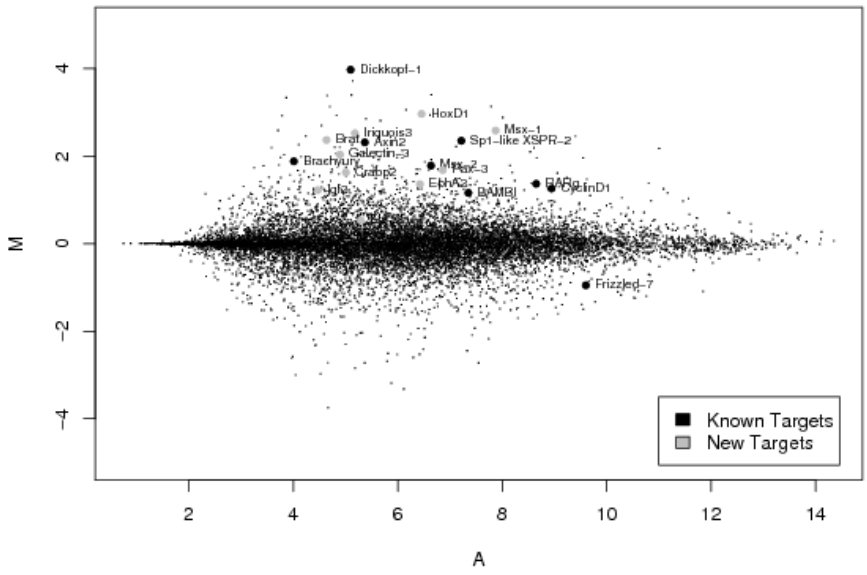


Figure 9: M- versus A-plot van de activator versus repressorvergelijking, met hierop de geselecteerde differentiële genen aangeduid.

biologische data mining te openen. Beide pakketten maken nu deel uit van het welgekende Bioconductor project [43](http://www.bioconductor.org). Dit is een open source softwareproject voor de analyse en het begrijpen van genomische data. Vooral biomaRt kent reeds een aanzienlijke gebruikersgroep. Eens het MAGE-ML data formaat een algemener gebruik kent, kan verwacht worden dat ook het RMAGEML pakket veel gebruikers zal hebben. Verder werd in deze thesis bijgedragen tot analyse van enkele biologische datasets. In deze thesis gaven we als voorbeeld het ontdekken van nieuwe Wnt-doelwit genen met behulp van microarrays. Ook heeft deze thesis bijgedragen aan andere microarray analyse software zoals de ontwikkeling van flexible en interactieve visualisatie modules voor Expression Profiler next generation (EP:NG) [67]. In het kader van CAGE, werd een *Arabidopsis* specifieke versie van het gen priorizatie tool Endeavour ontwikkeld. Door de grote verscheidenheid aan biologische gegevens en de steeds toenemende omvang van experimenten, zal de ontwikkeling van software en automatisering zoals in deze thesis besproken, de komende jaren een belangrijk onderzoeksgebied gebied blijven in de bioinformatica.

Appendix A: case study

Besides providing an automatic preprocessing pipeline, a second CAGE deliverable was the analysis of the compendium data. However due to the large delay in data production and the lack of availability of completed high quality experiments, the analysis of CAGE compendium data was not possible until very recently. Meanwhile several other microarray data analyses were performed during this Ph.D. and this chapter describes one of these studies were potential new target genes for the canonical Wnt pathway were discovered.

Identification of new canonical Wnt target genes

Signal transduction is a process where a cell transfers messages from its environment from the membrane through the cytoplasm and into the nucleus. In the nucleus the signal alters the expression of various genes that contribute to the cell's response to the external stimuli. The mode of signal transduction for pathways triggered by protein ligands is roughly the same regardless of the signal's nature. At the membrane, a signaling protein binds to a receptor that undergoes a configurational change. The receptor is then tagged with a phosphate group by an associated protein kinase enzyme. This receptor phosphorylation enables the receptor to recruit cytoplasmic signaling proteins that then initiate a cascade of events that eventually result in changes of gene expression. The Wnt signal transduction pathway plays an important role in developmental events during embryogenesis and is required for adult tissue maintenance. Perturbation of this signaling cascade promotes human degenerative diseases and cancer [82]. Wnt signals are pleiotropic and have effects in mitogenic stimulation, cell fate specification, and differentiation. Wnt signaling from frizzled receptors to β-catenin is called the canonical Wnt pathway and is evolutionary conserved. Once Wnt proteins bind to the Frizzled receptor

complex at the surface of a cell, the receptors transduce the signal to several intracellular proteins that include Dishevelled (Dsh), glycogen synthase kinase-3β (GSK-3), axin, Adenomatous Polyposis Coli (APC), and β-catenin (see Figure A-1). In normal cells, a complex containing GSK-3, APC and Axin, controls the cytoplasmic β-catenin levels and keeps them low by a continuous proteasome-mediated degradation as discussed in Chapter 2. When cells receive Wnt signals, the degradation pathway is inhibited, and consequently β-catenin accumulates in the cytoplasm and nucleus. Nuclear β-catenin interacts with transcription factors such as lymphoid enhancer-binding factor 1 and T-cell specific transcription factor (LEF/TCF) to alter transcription [82]. A large number of Wnt targets have been identified that includes members of the Wnt signal transduction pathway itself, and provide feedback control during Wnt signaling.

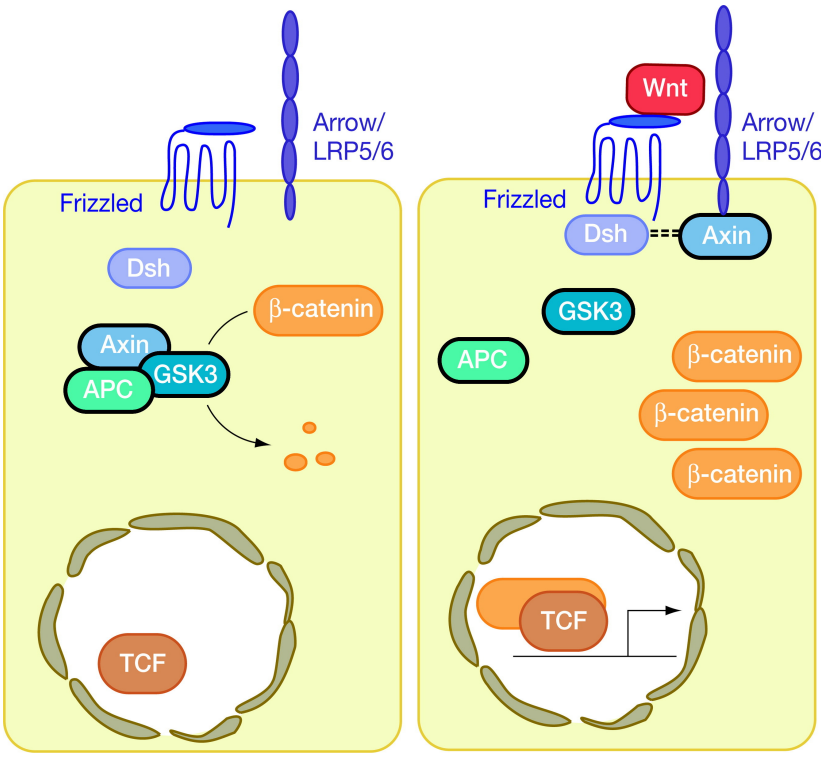


Figure A-1: The canonical Wnt pathway. In normal cells, a complex containing GSK-3, APC and Axin, controls the cytoplasmic β-catenin levels and keeps this level low by a continuous proteasome-mediated degradation. When cells receive Wnt signals, the degradation pathway is inhibited, and consequently β-catenin accumulates in the cytoplasm and nucleus affecting gene transcription. Figure from [82].

Experiment setup

Denayer[¶] *et al*. [26] developed transgenic *Xenopus Leavis* embryos in which the canonical Wnt pathway can be transcriptionally activated (construct containing the transactivation domain of VP16) or repressed (construct containing the EnR transrepression domain of *Drosophila* Engrailed) using a hormone inducible system. These developing embryos were used in an Affymetrix Chip

[¶]In collaboration with Department of Molecular Biomedical Research (RUG): Tinneke Denayer, Tom Deroo, Pieter De Bleser, Dominique Vlieghe, Yves Moreau, Frans Van Roy and Kris Vleminckx

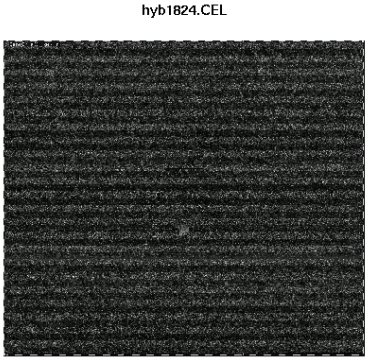


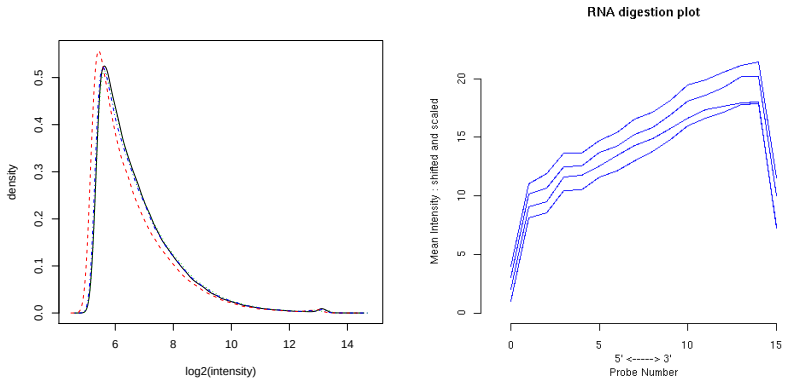
Figure A-2: Image of intensities on one of the slides as a quality assessment to see whether there are scratches or dust on the surface. No scratches are visible here.

screen to identify new Wnt target genes [26]. Four samples were hybridized to four Affymetrix *Xenopus laevis* GeneChips containing probe sets for over 14,400 transcripts. The first sample was taken from a transgenic embryo containing the activating construct and transcription of this construct was activated by Dex hormone treatment. Together with this sample, a control sample, containing an empty vector construct was treated with the same hormone treatment but this should not result in activation of the Wnt pathway. A third sample was taken from an embryo containing the repressor construct after transcription of this repressor was activated by Dex. Similarly as with the activator, a control transgenic embryo containing an empty vector was used as a control sample and received the same Dex treatment.

Quality assessment

As a first quality assessment, images of the hybridized chip were plotted. One of these plots is depicted in Figure A-2. From these images we could conclude that no obvious scratches or dust particles were interfering with the hybridization.

To compare the expression measurements obtained by the different chips in



(a) Density plot (b) RNA degradation plot

Figure A-3: Quality assessment: (a) density plot, showing distribution of signal intensities of all arrays. (b) RNA degradation plot to see if RNA degradation occured, parallel lines indicate no degradation occurred.

this experiment and to see if there are no outlier slides, a density plot was made, with the intensity densities of each array plotted in a different color Figure A-3a. Another quality assessment plot is an RNA degradation plot to assess if any RNA degradation occurred, which could reduce the measured expression levels.

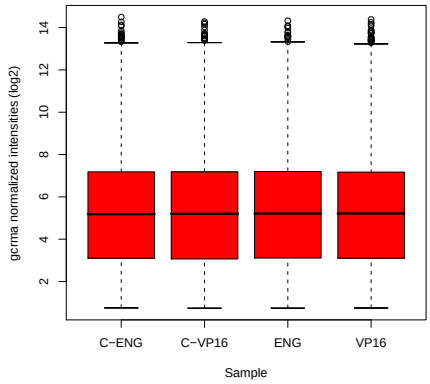


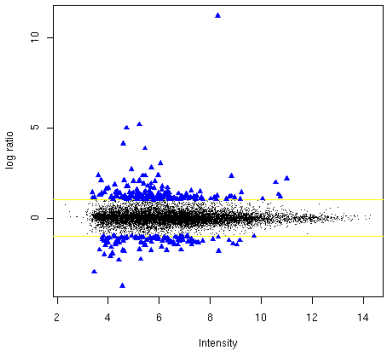
Figure A-4: Boxplot of normalized intensities grouped per array.

Normalization

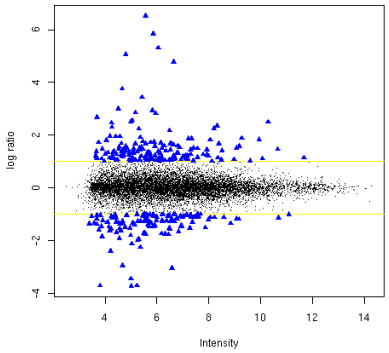
The data was normalized using the GCRMA method as discussed in Chapter 2. Figure A-4 shows boxplots of the intensities grouped per chip after normalization. From this plot we can derive that the normalization worked well.

Determining differentially expressed genes

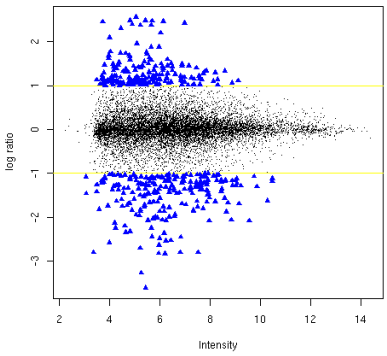
A next step in the analysis was to filter out genes that were not expressed at all. We used a threshold of 3.5 on the average $\log_2(\text{intensities})$ over all arrays. After applying this filter, 10,898 features remained to be analyzed for differential expression. The following sample comparisons were made: Wnt-activating versus its control, Wnt-repressing versus its control, Wnt-activating versus Wnt-repressing and the control of Wnt-activating versus the control of Wnt-repressing. As this is an experiment with no replicates the fold-change was used as a threshold for differential expression. Figure A-5 depicts the M versus A plots of the described comparisons. A threshold of $|M| > 1$ was taken to identify the differentially expressed genes. Table A-1 shows the top 100 most upregulated genes in the Wnt-activating versus Wnt-repressing comparison.



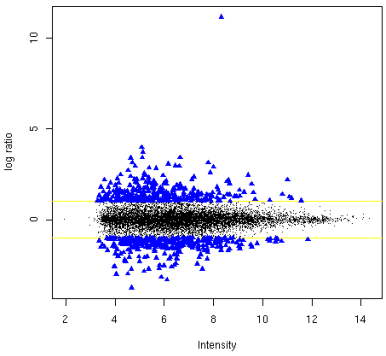
(a) activating/control-activating



(b) repressing/control-repressing



(c) control-activating/control-repressing



(d) activating/repressing

Figure A-5: M versus A plots. The horizontal lines indicate our threshold of $|M| > 1$. The triangles represent differentially expressed features.

Table A-1: Top 100 upregulated genes in activator versus repressor comparison

affyID	log2(ratio)	description
AFFX-Xl-U89963-1_at	11,13	Activation domain of VP16
Xl.251.1.S1_at	3,97	Dickkopf-1
Xl.11060.1.A1_at	3,71	gb:AW764623
Xl.610.1.S1_at	3,4	axin-related protein
Xl.1073.1.S1_at	3,39	maternal protein wnt-11
Xl.12126.1.S1_at	3,38	XHR1-B gene for HES-related 1B
Xl.16177.1.A1_at	3,13	Weakly similar to Axin 1
Xl.7809.1.A1_at	3,13	gb:BG554688
Xl.16120.1.A1_at	3,06	gb:BJ076352
Xl.3370.1.S1_at	2,96	homeobox protein HoxD1
Xl.1358.1.S1_at	2,93	homeobox protein xCAD2
Xl.8458.1.S1_at	2,93	Moderately similar to hypothetical protein FLJ20668
Xl.23739.1.A1_at	2,88	gb:CB564190
Xl.13415.1.A1_at	2,85	gb:BJ092954
Xl.16876.1.S1_at	2,81	gb:BJ048011
Xl.3529.1.A1_at	2,7	gb:BJ056268
Xl.1794.1.A1_at	2,58	Weakly similar to Homeobox protein MSX-1 (Hox-7) (H.sapiens)
Xl.4522.1.S1_at	2,52	homeobox transcription factor iriquois 3
Xl.15051.1.A1_at	2,46	Weakly similar to HeLa cyclin-dependent kinase 2 interacting protein
Xl.4964.1.S1_at	2,45	gb:BF612986
Xl.16120.1.A1_s_at	2,45	gb:BJ076352
Xl.37.1.S1_at	2,41	mRNA for Xom protein.
Xl.1775.1.S1_at	2,37	Brachyury and Tbx related protein Brat
Xl.2755.1.S1_at	2,34	Sp1-like zinc-finger protein XSPR-2
Xl.13925.1.A1_at	2,31	Weakly similar to Axin 2
Xl.11586.1.S1_at	2,26	gb:AW460665
Xl.366.1.S1_x_at	2,26	homeobox protein (Vox-1)
Xl.16144.1.A1_at	2,21	cDNA clone 21G4-1
Xl.5040.1.A1_a_at	2,2	Moderately similar to pre-alpha (globulin) inhibitor, H3 polypeptide
Xl.1975.1.A1_at	2,2	gb:BG439504
Xl.15037.1.S1_at	2,19	RDC1 like protein
Xl.699.1.S1_s_at	2,19	homeobox protein (Vox-15)
Xl.8949.1.S1_at	2,18	ornithine decarboxylase-2
Xl.16733.1.A1_at	2,13	Weakly similar to homeobox protein cdx-4
Xl.15362.1.A1_at	2,1	gb:BI443221
Xl.2755.2.A1_at	2,1	Sp1-like zinc-finger protein XSPR-2
Xl.558.1.S1_at	2,07	homeodomain transcription factor Xvex-1
Xl.79.1.S1_at	2,06	nuclear factor XPA26-T2
Xl.14578.1.A1_at	2,04	gb:BJ082183
Xl.15364.1.S1_at	2,03	galectin family xgalectin-IIIa
Xl.933.1.S1_at	2,03	Brachyury-like T-box transcription factor
Xl.23649.1.S1_at	2	angiotensin receptor related protein

Xl.17764.1.S1_at	1,99	Unknown (protein for MGC:53383)
Xl.26143.1.A1_at	1,98	Weakly similar to WAS protein family, member 2
Xl.1269.1.S1_at	1,97	Unknown (protein for MGC:52559)
Xl.699.1.S1_at	1,96	Vox-15
Xl.3540.1.S1_at	1,96	p75-like transmembrane protein fullback
Xl.2755.1.S1_a_at	1,96	Sp1-like zinc-finger protein XSPR-2
Xl.14499.1.S1_at	1,96	ectoderm neural cortex related-1
Xl.16181.1.A1_at	1,92	Weakly similar to F-box and leucine-rich repeat protein 2
Xl.7556.1.S1_at	1,92	LIM protein prickle
Xl.514.1.S1_at	1,88	brachyury (T)
Xl.5598.1.A1_at	1,87	gb:BF049400
Xl.21634.1.S1_at	1,87	homeobox containing peptide Hhox 7.1
Xl.1273.1.S1_at	1,83	xIRF-6 mRNA, complete cds.
Xl.8351.1.A1_x_at	1,79	gb:AW148150
Xl.2832.1.S1_at	1,79	putative deoxyribonucleoside kinase
Xl.15202.1.A1_at	1,78	Weakly similar to A49068 craniosynotosis-associated homeotic protein MSX2 - human (H.sapiens)
Xl.10269.1.S1_at	1,78	Xcad-3 mRNA
Xl.1529.1.S1_at	1,78	Similar to hypothetical protein FLJ14299, clone IMAGE:5571654, mRNA
Xl.54.1.S1_at	1,78	thylacine 1
Xl.5996.1.S1_at	1,77	gb:AW766768
Xl.608.1.S1_at	1,77	hyaluronan synthase 2
Xl.16656.1.A1_at	1,77	gb:BJ046565
Xl.14379.1.A1_at	1,75	Weakly similar to B44478 probable cell growth or differentiation regulator
Xl.3753.1.A1_at	1,73	gb:BF428388
Xl.4094.1.S1_at	1,72	Weakly similar to TGF(beta)-induced transcription factor 2 (H.sapiens)
Xl.1592.1.A1_at	1,7	gb:BJ052981
Xl.24860.1.A1_at	1,7	gb:BF047438
Xl.5679.1.S1_at	1,7	gb:BE026676
Xl.24766.1.A1_at	1,7	gb:BJ090857
Xl.6034.1.A1_at	1,7	Weakly similar to Fatty acid-binding protein, brain (B-FABP)
Xl.7697.1.S1_at	1,69	Tbx6
Xl.146.1.S1_at	1,68	XMyf-5 mRNA for a myogenic factor
Xl.25932.2.A1_s_at	1,68	Highly similar to Paired box protein Pax-3 (HUP2) (H.sapiens)
Xl.15691.1.A1_at	1,68	Weakly similar to KIAA0644 gene product (H.sapiens)
Xl.25379.1.A1_at	1,67	Weakly similar to hypothetical protein MGC2601 (H.sapiens)
Xl.69.1.S1_at	1,66	homeobox Iro protein
Xl.7720.1.A1_at	1,65	Weakly similar to Retrovirus-related POL polyprotein (Endonuclease) (M.musculus)
Xl.2434.1.A1_at	1,63	Weakly similar to Wnt-1 responsive Cdc42 homolog
Xl.1154.1.S1_at	1,63	xCRABP
Xl.3793.2.S1_x_at	1,63	early endoderm-specific nuclear factor Xenf

Xl.182.1.S1_at	1,62	XFD-13 protein
Xl.802.1.S1_at	1,61	organizer specific homeobox Xcad 1
Xl.26171.1.A1_s_at	1,6	gb:CD362154
Xl.22684.1.A1_at	1,59	Weakly similar to Glia derived nexin precursor (GDN)
Xl.22763.1.A1_at	1,59	gb:BJ081670
Xl.1255.1.S1_at	1,57	NGFI-B mRNA.
Xl.1268.1.S1_at	1,57	mRNA for homeobox protein
Xl.22609.1.S1_at	1,57	gb:BJ085136
Xl.2967.1.A1_at	1,57	gb:BG019611
Xl.23626.1.A1_at	1,55	clone IMAGE:5570434, mRNA.
Xl.13.2.A1_at	1,54	receptor tyrosine kinase (Pag) mRNA, complete cds
Xl.958.1.S2_at	1,54	smad interaction protein 1
Xl.7521.1.A1_at	1,54	Moderately similar to Kruppel-like factor 15
Xl.9842.1.A1_at	1,54	gb:BJ082875
Xl.9144.1.A1_at	1,52	gb:BG347183
Xl.23983.1.A1_at	1,52	gb:BJ083902
Xl.12067.1.S1_at	1,52	enhancer of split related 2

After compiling the differential expression results from all three comparisons and leaving out for example features that were differential in the comparison Wnt-activated versus Wnt-repressed but also in the comparison control-activated versus control-repressed, the remaining differentially expressed features were subselected with an in silico promoter analysis for Wnt responsive promoter sequences by P. De Bleser [26]. And from this subset a final set of genes was chosen for follow-up studies. Interestingly next to new targets, a set of already known Wnt-target genes was found (see Table A-2). Figure A-6 shows an M versus A plot where the selected differential expressed genes are highlighted

Validation of differential expression by qPCR

Quantitative PCR (qPCR) or real-time PCR, enables the quantification of concentrations of DNA, cDNA, or RNA templates. The technique is based on the detection of a fluorescent reporter molecule of whose the number increases as the PCR product accumulates during successive cycles of PCR. qPCR is

Table A-2: Known and putative Wnt-target genes

Known direct Wnt target gene	putative Wnt target gene
Dickkopf-1	HoxD1
Sp1-like XSPR-2	Iriquois 3
Axin 2	Brachyury and Tbx related protein (BRAT)
Brachyury	Galectin-3
Cyclin D1	Crab2
BAMBI	EphA2
Msx-2	Insulin-like growth factor
Frizzled-7	Ets-2a
RAR γ	Pax-3
Wnt-1 responsive Cdc42 homolog	Msx-1

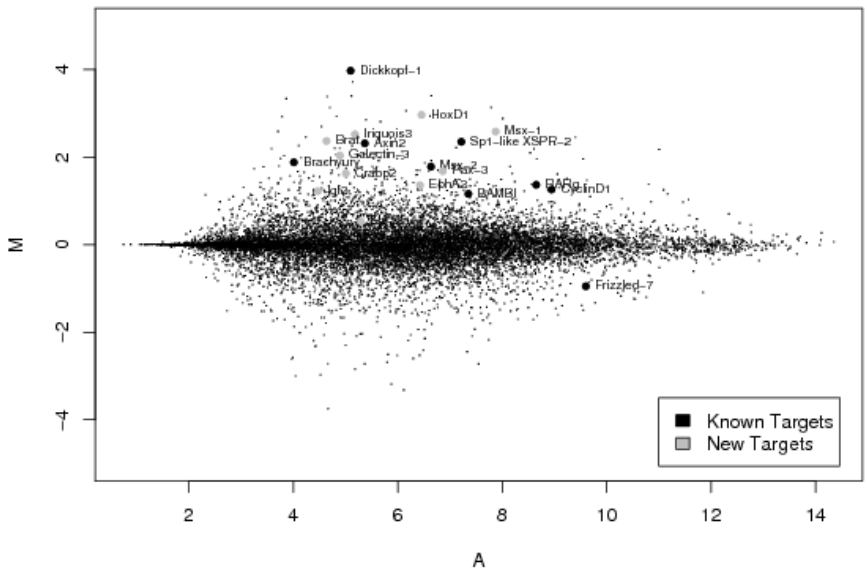


Figure A-6: Selected differential expressed genes plotted on M versus A plot of the activator versus repressor comparison. Highlighted in grey are the new target genes, in black the already known Wnt targets.

a widely used tool to verify results coming from microarray experiments, as it provides an independent measurement of the amount of a specific mRNA present in a sample. To assess if the genes selected are truly differentially expressed, a quantitative PCR was performed on the putative new Wnt-target genes, the known Wnt target BAMBI was taken as a control [26]. From these results we can conclude that in general the microarray experiment results can be confirmed by qPCR.

Discussion

This case study described one of the microarray data analysis performed during this Ph.D. The different steps taken in the analyses were quality assessment, data normalization, detection of differential expression, and experimental validation. From the result it is clear that microarrays can efficiently monitor changes in gene expression, even in situations where no replicates are present (although the use of replicates should be highly recommended). The analysis found known Wnt-target genes together with a set of new putative Wnt-targets. Further experimental validation by qPCR confirmed the results obtained with the microarray experiment. And now subsequent biological experiments are needed to find out more about the relation of these new potential Wnt-targets and the canonical Wnt-pathway.

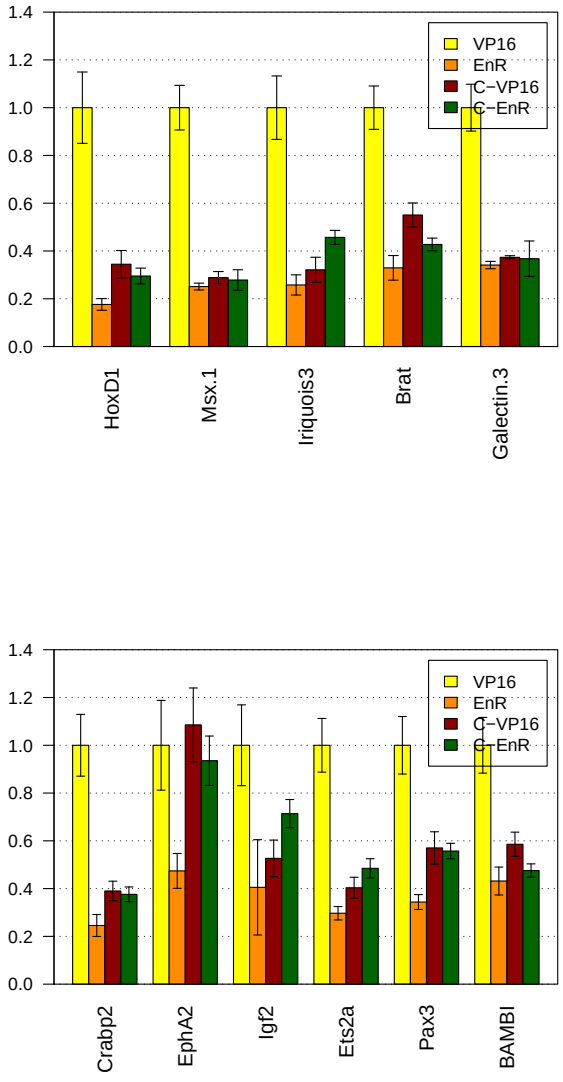


Figure A-7: A frequently used approach to validate the differential expression measured by microarrays is qPCR. The plots above show the candidate Wnt target genes that were found differentially expressed in the microarray analysis. Note that the microarray results are confirmed for most of these genes.

Appendix B: vignette RMAGEML software package

Introduction

MAGE-ML or Microarray Gene Expression Markup Language is a language designed to describe and exchange information about microarray experiments. MAGE-ML is based on XML and can describe microarray designs, microarray experiment setups, gene expression data, and data analysis results.
This package provides the link between MAGE-ML files and BioConductor. It gives the possibility to read in MAGE-ML files that describe cDNA microarray experiments and update those documents with normalized data and information on the normalization procedure. The *RMAGEML* functions convert the MAGE-ML files into the customary BioConductor objects (i.e., `marrayLayout, marrayInfo` and `marrayRaw` objects or limma `RGList` objects).

Here we give a short introduction to the Microarray and GeneExpression Object Model (MAGE-OM) and how we implemented the extraction of information necessary to make BioConductor objects. For a full description of MAGE-OM, we refer to the Gene Expression Specification: http://www.omg.org/cgi-bin/doc?formal/03-02-03.
The main classes of the MAGE object model are BioSequence, Quantitationtype, ArrayDesign, DesignElement, Array, BioMaterial, BioAssay, BioAssayData, Experiment, HigherLevelAnalysis, Protocol, Description, AuditAndSecurity, Measurement, and BioEvent.
In MAGE-ML these translate into packages with the same name. The packages needed for building BioConductor objects are BioAssayData, BioAssay, BioMaterial, BioSequence, ArrayDesign, and DesignElement.

The DesignElement package contains a mapping of *Features*, which are the actual features present on the array, to *Reporters*, the reporter a feature represents. The DesignElement package also provides a mapping from *Reporters* to their corresponding *BioSequence* references. These *BioSequence* objects are characterized by their name and database entries in the BioSequence package. The ArrayDesign package contains information on the layout of the array. From this package, we can derive the position of each *Feature* on the array in terms of *Zone* (block or grid) and row and column within each *Zone*. The BioAssayData package describes the feature references that were assayed and the measured and derived *QuantitationTypes*. The BioAssay package describes the different steps in the microarray experiment. The last package used to make BioConductor objects is the BioMaterial package and describes how a sample is treated to obtain, for example, labeled samples used for hybridization.

Getting started

Installing the package. The package can be installed as a normal R package: download the RMAGEML_2.0.4.tar.gz package and under Unix use the command

`R CMD INSTALL RMAGEML_2.1.0.tar.gz`.

The equivalent command for Windows is

`Rcmd INSTALL RMAGEML_2.1.0.zip`.

The package automatically loads the Biobase and marrayInput packages from BioConductor and the SJava libraries, so these should be installed as well.

Starting R.

Loading the package. You can load the package into R by typing

```
> library(RMAGEML)
Loading required package: marray
```

```
Loading required package: limma
Loading required package: limma
Loading required package: Biobase

Welcome to Bioconductor

Vignettes contain introductory material.
To view, simply type 'openVignette()' or start with
'help(Biobase)'.
For details on reading vignettes, see the openVignette
  help page.
```

Import to marray packages

One step import and creation of an marrayRaw object from MAGE-ML files

In the marray packages of BioConductor the design of an array experiment is typically described by an `marrayLayout` and `marrayInfo` object. The function `importMAGEML` parses all MAGE-ML files present in the directory, which is given as a parameter to the function. From these files it creates an `marrayLayout` object, containing the Layout of one type of microarrays, and an `marrayInfo` object containing the gene names and database entries of the features spotted on the array. The name of the database to which the entries refer, is given in the 'notes' slot of the Gnames object. Next the function will extract the raw data values and output a complete *marrayRaw* object as a result.

The function can be tested on the MEXP-14 dataset. This example is available from ArrayExpress at http://www.ebi.ac.uk/arrayexpress/.
If one knows which *DesignElement Dimension*, *QuantitationType Dimension* and *Quantitation Types* are required, the import function can be used as:

```
> datadir <- system.file("MAGEMLdata", package = "RMAGEML")
> raw <- importMAGEML(directory = datadir, package = "marray",
+     arrayID = "A-MEXP-14", DED = "DED:707", QTD = "QTD:707",
+     name.Rf = "QT:F635 Mean", name.Rb = "QT:B635 Median",
+     name.Gf = "QT:F532 Mean", name.Gb = "QT:B532 Median")
- Java Virtual Machine is running -
parsing MAGEML files
making Layout and Gnames objects
Reading  am2730miame.txt
Reading  am2731miame.txt
Reading  am2732m.txt
```

```
Reading  am2736m.txt
Reading  am2737m.txt
Reading  tm1826m.txt
Reading  tm1827m.txt
Reading  tm1829m.txt
Reading  tm1830m.txt
Reading  tm1831m.txt
> print(raw)

An object of class "marrayRaw"
@maRf
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
[1,] 5841 2030 2968   45 1828 1975 2077 1775 2202   841
[2,] 2002 1312  421   96  399  557  295  748  465    83
[3,] 2254 2057 1097 1163  649  917  755 1276  985   335
[4,] 2212 1492  782  767  709 1114  620 1004  860   488
[5,]   73   76   42   54   45   49   49   46   47    43
955 more rows ...

@maGf
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
[1,]  852  750 1587  135 1625 1183 1598 1108  807  1746
[2,]  652  529  404  162  397  515  285  390  291   190
[3,]  576  615  634  386  734  820  696  573  457   572
[4,]  781  589  733  366  758  848  667  597  559   716
[5,]  157  143  111  124  130  148  135  137  146   131
955 more rows ...

@maRb
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
[1,]   42   42   36   39   41   47   40   45   49    42
[2,]   41   41   35   39   40   45   39   43   42    41
[3,]   41   42   34   41   40   44   40   43   42    42
[4,]   41   40   34   41   40   43   39   43   41    41
[5,]   41   39   34   40   40   42   39   43   41    41
955 more rows ...

@maGb
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
[1,]  150  130   87  120  104  135  117  137  168   127
[2,]  147  128   88  111  105  136  116  131  137   121
[3,]  140  124   85  106  105  133  116  128  135   121
[4,]  138  122   88  108  106  133  116  128  134   120
[5,]  138  122   87  106  104  133  114  128  133   118
955 more rows ...

@maW
<0 x 0 matrix>
```

```
@maLayout
An object of class "marrayLayout"
@maNgr
[1] 4

@maNgc
[1] 4

@maNsr
[1] 10

@maNsc
[1] 6

@maNspots
[1] 960

@maSub
[1] TRUE

@maPlate
factor(0)
Levels:

@maControls
factor(0)
Levels:

@maNotes
[1] ""


@maGnames
An object of class "marrayInfo"
@maLabels
[1] "none" "none" "none" "none" "none"
955 more elements ...

@maInfo
[1] aj508733 VO0618   aj291984 aj306233 aj310439
142 Levels: af025843 af034412 af135499 aj132353  ... y17187
955 more rows ...

@maNotes
[1] "Identifiers refer to database: DB:embl"


@maTargets
An object of class "marrayInfo"
```

```
@maLabels
 [1] "am2730miame.txt" "am2731miame.txt" "am2732m.txt" "am2736m.txt"
 [5] "am2737m.txt"     "tm1826m.txt"     "tm1827m.txt" "tm1829m.txt"
 [9] "tm1830m.txt"     "tm1831m.txt"

@maInfo
       Cy3      Cy5
1  AM-Pool AM2730-I
2  AM-Pool AM2731-1
3  AM-Pool AM2732-I
4  AM-Pool AM2736-I
5  AM-Pool AM2737-I
6  AM-Pool TM1826-I
7  AM-Pool TM1827-I
8  AM-Pool TM1829-I
9  AM-Pool TM1830-I
10 AM-Pool TM1831-I

@maNotes
[1] "Description of the targets"


@maNotes
character(0)
```

If however you do not know which *DesignElement Dimension*, *Quantitation-Type Dimension* and *Quantitation Types* to use, you can call the function as follows:

```
> datadir <- system.file("MAGEMLdata", package = "RMAGEML")
> if (interactive()) {
+     raw <- importMAGEML(directory = datadir, package = "marray")
+ }
```

This will generate a few selection panels which allow selection of the appropriate *DesignElement Dimension, QuantitationType Dimension* and *Quantitation Types*.

Creation of a Gnames marrayInfo object

If one just wants to make an marrayInfo object containing the gene names and database identifiers of the spotted features the function getGnames can be used.

```
> data <- system.file("MAGEMLdata", package = "RMAGEML")
> mageom <- importMAGEOM(directory = data)
parsing MAGEML files
```

```
> getGnames(mageom, arrayID = "A-MEXP-14", DED = "DED:707",
+  package = "marray")

An object of class "marrayInfo"
@maLabels
[1] "none" "none" "none" "none" "none"
955 more elements ...

@maInfo
[1] aj508733 VO0618   aj291984 aj306233 aj310439
142 Levels: af025843 af034412 af135499 aj132353 ... y17187
955 more rows ...

@maNotes
[1] "Identifiers refer to database: DB:embl"
```

Again leaving out the 'DED' parameter will cause selection panels to pop up displaying the available *DesignElement Dimensions*.

Creation of an marrayLayout object

In the marray packages the information on the array layout is stored in an marrayLayout object which can be created by the getArrayLayout function.

```
> data <- system.file("MAGEMLdata", package = "RMAGEML")
> mageom <- importMAGEOM(directory = data)

parsing MAGEML files

> getArrayLayout(mageom, arrayID = "A-MEXP-14", DED = "DED:707")

An object of class "marrayLayout"
@maNgr
[1] 4

@maNgc
[1] 4

@maNsr
[1] 10

@maNsc
[1] 6

@maNspots
[1] 960
```

```
@maSub
[1] TRUE

@maPlate
factor(0)
Levels:

@maControls
factor(0)
Levels:

@maNotes
[1] ""
```

Make an marrayRaw object

The function makeMarrayRaw takes a Gnames and Layout object and parameters corresponding to the *DesignElement Dimension*, *QuantitationType Dimension* and *Quantitation Types* to create an *marrayRaw* object.

```
> data <- system.file("MAGEMLdata", package = "RMAGEML")
> mageom <- importMAGEOM(directory = data)
parsing MAGEML files
> gnames <- getGnames(mageom, arrayID = "A-MEXP-14", DED =
+  "DED:707", package = "marray")
> layout <- getArrayLayout(mageom, arrayID = "A-MEXP-14",
+ DED = "DED:707")
> raw <- makeMarrayRaw(mageOM = mageom, layout = layout,
+  gnames = gnames, directory = data,
+  arrayID = "A-MEXP-14", DED = "DED:707", QTD = "QTD:707",
+  name.Rf = "QT:F635 Mean", name.Rb = "QT:B635 Median",
+  name.Gf = "QT:F532 Mean", name.Gb = "QT:B532 Median")
Reading  am2730miame.txt
Reading  am2731miame.txt
Reading  am2732m.txt
Reading  am2736m.txt
Reading  am2737m.txt
Reading  tm1826m.txt
Reading  tm1827m.txt
Reading  tm1829m.txt
Reading  tm1830m.txt
Reading  tm1831m.txt
```

Import to limma package

One step import and creation of a limma RGList object from MAGE-ML files

In the limma package of BioConductor the raw data is stored in an `RGList` object. The function `importMAGEML` parses all MAGE-ML files present in the directory which is given as a parameter to the function. From these files it creates the `RGList` object, containing the layout, gene names and database entries of the features spotted on the array and the foreground and background intensities for the green and red channels.

The function can be tested on the MEXP-14 dataset. This example is available from ArrayExpress at http://www.ebi.ac.uk/arrayexpress/.
For import to limma the same function as MAGEML import to marray packages can be used, just adapt the name of the package into limma as follows:

```
> datadir <- system.file("MAGEMLdata", package = "RMAGEML")
> raw <- importMAGEML(directory = datadir, package = "limma",
+ arrayID = "A-MEXP-14", DED = "DED:707", QTD = "QTD:707",
+ name.Rf = "QT:F635 Mean", name.Rb = "QT:B635 Median",
+ name.Gf = "QT:F532 Mean", name.Gb = "QT:B532 Median")

parsing MAGEML files
Reading  am2730miame.txt
Reading  am2731miame.txt
Reading  am2732m.txt
Reading  am2736m.txt
Reading  am2737m.txt
Reading  tm1826m.txt
Reading  tm1827m.txt
Reading  tm1829m.txt
Reading  tm1830m.txt
Reading  tm1831m.txt
> print(raw)

An object of class "RGList"
$R
     am2730miame.txt am2731miame.txt am2732m.txt am2736m.txt am2737m.txt
[1,]            5841            2030        2968          45        1828
[2,]            2002            1312         421          96         399
[3,]            2254            2057        1097        1163         649
[4,]            2212            1492         782         767         709
[5,]              73              76          42          54          45
     tm1826m.txt tm1827m.txt tm1829m.txt tm1830m.txt tm1831m.txt
[1,]        1975        2077        1775        2202         841
[2,]         557         295         748         465          83
[3,]         917         755        1276         985         335
[4,]        1114         620        1004         860         488
[5,]          49          49          46          47          43
955 more rows ...
```

```
$G
     am2730miame.txt am2731miame.txt am2732m.txt am2736m.txt am2737m.txt
[1,]             852             750        1587         135        1625
[2,]             652             529         404         162         397
[3,]             576             615         634         386         734
[4,]             781             589         733         366         758
[5,]             157             143         111         124         130
     tm1826m.txt tm1827m.txt tm1829m.txt tm1830m.txt tm1831m.txt
[1,]        1183        1598        1108         807        1746
[2,]         515         285         390         291         190
[3,]         820         696         573         457         572
[4,]         848         667         597         559         716
[5,]         148         135         137         146         131
955 more rows ...

$Rb
     am2730miame.txt am2731miame.txt am2732m.txt am2736m.txt am2737m.txt
[1,]              42              42          36          39          41
[2,]              41              41          35          39          40
[3,]              41              42          34          41          40
[4,]              41              40          34          41          40
[5,]              41              39          34          40          40
     tm1826m.txt tm1827m.txt tm1829m.txt tm1830m.txt tm1831m.txt
[1,]          47          40          45          49          42
[2,]          45          39          43          42          41
[3,]          44          40          43          42          42
[4,]          43          39          43          41          41
[5,]          42          39          43          41          41
955 more rows ...

$Gb
     am2730miame.txt am2731miame.txt am2732m.txt am2736m.txt am2737m.txt
[1,]             150             130          87         120         104
[2,]             147             128          88         111         105
[3,]             140             124          85         106         105
[4,]             138             122          88         108         106
[5,]             138             122          87         106         104
     tm1826m.txt tm1827m.txt tm1829m.txt tm1830m.txt tm1831m.txt
[1,]         135         117         137         168         127
[2,]         136         116         131         137         121
[3,]         133         116         128         135         121
[4,]         133         116         128         134         120
[5,]         133         114         128         133         118
955 more rows ...

$genes
  Block Row Column       ID Name
1     1   1      1 aj508733 none
2     1   1      2   VO0618 none
```

```
3     1   1      3 aj291984 none
4     1   1      4 aj306233 none
5     1   1      5 aj310439 none
955 more rows ...
```

Similarly if one only specifies the 'directory' and the 'package', selection panels will pop up to select the *DesignElement Dimension, QuantitationType Dimension* and *Quantitation Types*.

Creating the genes dataframe of an RGList object

In limma the gene names, gene identifiers and layout information is stored in a dataframe which can be created by the getArrayLayoutLimma function.

```
> data <- system.file("MAGEMLdata", package = "RMAGEML")
> mageom <- importMAGEOM(directory = data)
parsing MAGEML files
> genes <- getArrayLayoutLimma(mageom, arrayID = "A-MEXP-14",
+ DED = "DED:707")
> print(genes[1:10, ])

   Block Row Column       ID Name
1      1   1      1 aj508733 none
2      1   1      2   VO0618 none
3      1   1      3 aj291984 none
4      1   1      4 aj306233 none
5      1   1      5 aj310439 none
6      1   1      6 aj409363 none
7      1   1      7 aj310516 none
8      1   1      8 aj306230 none
9      1   1      9 aj310436 none
10     1   1     10 aj291834 none
```

Make an RGList object

The function makeRG takes a genes dataframe (containing the layout, gene identifiers and gene names), and parameters corresponding to *DesignElement Dimension, QuantitationType Dimension* and *Quantitation Types* to create a limma `RGList` object.

```
> data <- system.file("MAGEMLdata", package = "RMAGEML")
> mageom <- importMAGEOM(directory = data)
parsing MAGEML files
> genes <- getArrayLayoutLimma(mageom, arrayID = "A-MEXP-14",
+ DED = "DED:707")
> raw = makeRG(mageOM = mageom, genes = genes, directory = data,
```

```
+     arrayID = "A-MEXP-14", DED = "DED:707", QTD = "QTD:707",
+     name.Rf = "QT:F635 Mean", name.Rb = "QT:B635 Median",
+     name.Gf = "QT:F532 Mean", name.Gb = "QT:B532 Median")
Reading  am2730miame.txt
Reading  am2731miame.txt
Reading  am2732m.txt
Reading  am2736m.txt
Reading  am2737m.txt
Reading  tm1826m.txt
Reading  tm1827m.txt
Reading  tm1829m.txt
Reading  tm1830m.txt
Reading  tm1831m.txt
```

Update MAGE-ML documents with normalized microarray data

The following code section describes how to update an existing MAGE-ML document with normalized microarray data.

```
#First import the MAGE-ML file containing the raw data
>raw <- importMAGEML(directory = "/home/steffen/data/MEXP-14",
+ name.Gf = "QT:F635 Mean", name.Gb = "QT:B635 Median",
+ name.Rf = "QT:F532 Mean", name.Rb = "QT:B532 Median",
+ package="marray")

#perform a loess normalization
>norm<-maNorm(raw)

#Read in the object model to update it with the normalized data
>mageom <- importMAGEOM(directory = "/home/steffen/data/E-MEXP-14")
>outputDirectory <- "/home/steffen/XMLout"

#input the normalization information such as protocol,
# filenames and quantitation types
>magemlFile <- "RMAGEMLtest2.xml"
>rawDataFiles <- raw@maTargets@maLabels
>externalDataFiles <- c("deriv_test1.txt","deriv_test2.txt",
"deriv_test3.txt","deriv_test4.txt","deriv_test5.txt",
"deriv_test6.txt","deriv_test7.txt","deriv_test8.txt",
"deriv_test9.txt","deriv_test10.txt")
>protocolID <- "P-maNorm-test"
>protocol <- "Applied maNorm to the raw signal intensities"
>qtID <- c("esat.kuleuven.ac.be:maNorm")
>qtName <- c("applied marrayNorm")
>qtDescription <- c("some description")
>qtScale <- c("linear")
>qtDataType <- c("scalar")
>qtDimID <- "esat.kuleuven.ac.be:QTD-test"
>date <- "testdate"
```

```
>tfmID <- "TFM-testID"
>BADIDs <- c("esat.kuleuven.ac.be:BAD-test1",
"esat.kuleuven.ac.be:BAD-test2",
"esat.kuleuven.ac.be:BAD-test3","esat.kuleuven.ac.be:BAD-test4",
"esat.kuleuven.ac.be:BAD-test5","esat.kuleuven.ac.be:BAD-test6",
"esat.kuleuven.ac.be:BAD-test7","esat.kuleuven.ac.be:BAD-test8",
"esat.kuleuven.ac.be:BAD-test9","esat.kuleuven.ac.be:BAD-test10")
>derivedBioAssayIDs <- c("esat.kuleuven.ac.be:DBA-test1",
"esat.kuleuven.ac.be:DBA-test2","esat.kuleuven.ac.be:DBA-test3",
"esat.kuleuven.ac.be:DBA-test4","esat.kuleuven.ac.be:DBA-test5",
"esat.kuleuven.ac.be:DBA-test6","esat.kuleuven.ac.be:DBA-test7",
"esat.kuleuven.ac.be:DBA-test8","esat.kuleuven.ac.be:DBA-test9",
"esat.kuleuven.ac.be:DBA-test10")
>derivedBioAssayDataIDs <- c("esat.kuleuven.ac.be:DBD-test1",
"esat.kuleuven.ac.be:DBD-test2","esat.kuleuven.ac.be:DBD-test3",
"esat.kuleuven.ac.be:DBD-test4","esat.kuleuven.ac.be:DBD-test5",
"esat.kuleuven.ac.be:DBD-test6","esat.kuleuven.ac.be:DBD-test7",
"esat.kuleuven.ac.be:DBD-test8","esat.kuleuven.ac.be:DBD-test9",
"esat.kuleuven.ac.be:DBD-test10")

#now update the MAGE-OM
>addNormToMAGEML(mageOM = mageom, norm = norm, outputDirectory =
outputDirectory, externalDataFiles = externalDataFiles,
protocolID = protocolID, protocol = protocol, date=date,
qtID = qtID, qtName = qtName,  qtDescription = qtDescription,
qtScale = qtScale, qtDataType = qtDataType, qtDimID = qtDimID,
transformationID = tfmID, arrayID="A-MEXP-14",DED = "none",
BADIDs = BADIDs, derivedBioAssayIDs = derivedBioAssayIDs,
derivedBioAssayDataIDs = derivedBioAssayDataIDs,
 rawDataFiles=rawDataFiles)

#write out the MAGE-ML document
>writeMAGEML(mageOM = mageom, directory = outputDirectory,
 file = magemlFile)
```

Appendix C: vignette of biomaRt software package

Introduction

The BioConductor *biomaRt* package provides and API in R to query BioMart databases such as Ensembl (http://www.ensembl.org), a software system which produces and maintains automatic annotation on metazoan genomes. Two sets of functions are currently implemented.
A first set of functions is tailored towards Ensembl and are a set of commonly used queries in microarray data analysis. A second set of functions aims to mimic functionality of other BioMart APIs such as Martshell, Martview, etc. (see http://www.biomart.org for more information). These functions are very general, and can be used with any BioMart system. They allow retrieval of all information that other BioMart APIs provide. With these two sets of functions, one can for example annotate the features on your array with the latest annotations starting from identifiers such as Affymetrix, RefSeq, Entrez Gene,etc. Annotation includes gene names, GO, OMIM annotation, etc. On top of this, biomaRt enables you to retrieve any type of information available from the BioMart databases from R.

Objects

Mart-class

An object of the `Mart` class stores connections to BioMart databases and aditional information about the BioMarts. It has the following slots:

- `mysql`: Logical indicating if access to BioMart database should use MySQL or use the BioMart webservice over HTTP (default)
- `connections`: Stores the MySQLConnections
- `mysqldriver`: Stores the MySQL driver
- `mainTables`: List of the main tables in the BioMart database
- `biomart`: Name of the BioMart database
- `host`: Hostname of the BioMart database
- `dataset`: Name of the dataset that is in use
- `filters`: Environment that stores information on BioMart filters
- `attributes`: Environment that stores information on BioMart attributes

Simple biomaRt functions for frequently used queries to Ensembl

In this section we describe a set of simple functions which are frequently used in the microarray community. More powerful functions and data retrieval from all BioMart databases is described in the next section ”Advanced data retrieval with BioMart API functions”.

Selecting a BioMart database to use

A first step in using the biomaRt package is to connect to a BioMart database. The function useMart establishes a connection with the BioMart databases Default this function will connect to public BioMart databases.If you want to use local BioMart install you have to set the local argument to TRUE and specify host, password and user details in the corresponding arguments.

```
> library(biomaRt)
Loading required package: XML
Loading required package: RCurl
> mart <- useMart("ensembl")
```

You can only hold a limited number of connections with different BioMarts when using biomaRt in MySQL mode. The function martDisconnect can be used to close a mart connection.

```
> martDisconnect(mart)
```

Annotating identifiers with gene information

The function `getGene` uses a query id to look up the name, description and chromosomal information of the corresponding gene. Currently the `getGene` function takes identifiers from entrezgene, ensembl, refseq, affy, hugo and embl.
The *id* argument is either a vector of identifiers or a single identifier to be annotated.
The *array* argument takes affy array identifiers as values. A list of possible identifiers supported by the package can be obtained by executing the function `getAffyArrays`.
The *mart* argument is a mart connection, which was obtained using the method `martConnect`.
The *type* takes the values of 'entrezgene','refseq','hugo', 'ensembl' and 'embl' to clarify which type of identifier is specified in the id argument.

First we select the BioMart databases and the dataset we want to use.

```
> mart <- useMart("ensembl", dataset = "hsapiens_gene_ensembl")
Checking attributes and filters ... ok
```

Then we check which affy arrays are available:

```
> getAffyArrays(mart)
 [1] "affy_hg_focus"
 [2] "affy_hg_u133a"
 [3] "affy_hg_u133a_2"
 [4] "affy_hg_u133b"
 [5] "affy_hg_u133_plus_2"
 [6] "affy_hg_u95av2"
 [7] "affy_hg_u95b"
 [8] "affy_hg_u95c"
```

```
 [9] "affy_hg_u95d"
[10] "affy_hg_u95e"
[11] "affy_u133_x3p"
```

Assume now that we have some upregulated features that we want to annotate. To get the gene information on a certain affy array do:

```
> upregulated <- c("210708_x_at", "202763_at", "211464_x_at")
> gene <- getGene(id = upregulated, array = "affy_hg_u133_plus_2",
+     mart = mart)
> gene
  HUGO symbol
1       CASP3
2       CASP3
3
4
5      CASP10
6
7
8       CASP6
9

1 Caspase-3 precursor (CASP-3) (Apopain) (Cysteine protease CPP32)
2 Caspase-3 precursor (CASP-3) (Apopain) (Cysteine protease CPP32)
3 Caspase-10 precursor (CASP-10) (ICE-like apoptotic protease 4)
4 Caspase-10 precursor (CASP-10) (ICE-like apoptotic protease 4)
5 Caspase-10 precursor (CASP-10) (ICE-like apoptotic protease 4)
6 Caspase-10 precursor (CASP-10) (ICE-like apoptotic protease 4)
7 Caspase-10 precursor (CASP-10) (ICE-like apoptotic protease 4)
8 Caspase-6 precursor (CASP-6) (Apoptotic protease Mch-2)
9 Caspase-6 precursor (CASP-6) (Apoptotic protease Mch-2)
  chromosome  band ensembl_gene_id ensembl_transcript_id
1          4 q35.1 ENSG00000164305       ENST00000308394
2          4 q35.1 ENSG00000164305       ENST00000352650
3          2 q33.1 ENSG00000003400       ENST00000360132
4          2 q33.1 ENSG00000003400       ENST00000346817
5          2 q33.1 ENSG00000003400       ENST00000272879
6          2 q33.1 ENSG00000003400       ENST00000286186
7          2 q33.1 ENSG00000003400       ENST00000313728
8          4   q25 ENSG00000138794       ENST00000265164
9          4   q25 ENSG00000138794       ENST00000352981
```

When using other identifiers we have to specify the `type` and `species`, use the function `getSpecies` to find valid species names.

```
> getGene(id = 100, type = "entrezgene", mart = mart)
  HUGO symbol
1         ADA
```

```
                                                 description
1 Adenosine deaminase (EC 3.5.4.4) (Adenosine aminohydrolase).
  chromosome   band ensembl_gene_id ensembl_transcript_id
1         20 q13.12 ENSG00000196839       ENST00000359372
```

GO annotation

Gene Onotology annotation can be retrieved with the function `getGO`. The arguments are the same as the function `getGene`.

```
> go <- getGO(id = upregulated[1], array = "affy_hg_u133_plus_2",
+     mart = mart)
> go
        go_id                   go_description code
1  GO:0005515                  protein binding IEA
2  GO:0008234 cysteine-type peptidase activity IEA
3  GO:0030693                caspase activity IEA
4  GO:0006508     proteolysis and peptidolysis IEA
5  GO:0042981          regulation of apoptosis IEA
6  GO:0005634                       nucleus IDA
7  GO:0005634                       nucleus IEA
8  GO:0005737                     cytoplasm IDA
9  GO:0005737                     cytoplasm IEA
10 GO:0030225       macrophage differentiation IMP
11 GO:0030225       macrophage differentiation IEA
12 GO:0030690              Noc1p-Noc2p complex IMP
13 GO:0030690              Noc1p-Noc2p complex IEA
14 GO:0005515                  protein binding IEA
15 GO:0030693                caspase activity IEA
16 GO:0006508     proteolysis and peptidolysis IEA
17 GO:0042981          regulation of apoptosis IEA
18 GO:0008234 cysteine-type peptidase activity IEA
19 GO:0030693                caspase activity TAS
20 GO:0042802             protein self binding IPI
21 GO:0006508     proteolysis and peptidolysis IEA
22 GO:0006917           induction of apoptosis TAS
23 GO:0042981          regulation of apoptosis IEA
24 GO:0005515                  protein binding IEA
25 GO:0008234 cysteine-type peptidase activity IEA
26 GO:0030693                caspase activity IEA
27 GO:0006508     proteolysis and peptidolysis IEA
28 GO:0042981          regulation of apoptosis IEA

   ensembl_gene_id ensembl_transcript_id
1  ENSG00000003400       ENST00000360132
2  ENSG00000003400       ENST00000360132
3  ENSG00000003400       ENST00000360132
4  ENSG00000003400       ENST00000360132
5  ENSG00000003400       ENST00000360132
```

```
6  ENSG00000003400       ENST00000360132
7  ENSG00000003400       ENST00000360132
8  ENSG00000003400       ENST00000360132
9  ENSG00000003400       ENST00000360132
10 ENSG00000003400       ENST00000360132
11 ENSG00000003400       ENST00000360132
12 ENSG00000003400       ENST00000360132
13 ENSG00000003400       ENST00000360132
14 ENSG00000003400       ENST00000346817
15 ENSG00000003400       ENST00000346817
16 ENSG00000003400       ENST00000346817
17 ENSG00000003400       ENST00000346817
18 ENSG00000003400       ENST00000272879
19 ENSG00000003400       ENST00000272879
20 ENSG00000003400       ENST00000272879
21 ENSG00000003400       ENST00000272879
22 ENSG00000003400       ENST00000272879
23 ENSG00000003400       ENST00000272879
24 ENSG00000003400        ENST00000286186
25 ENSG00000003400       ENST00000286186
26 ENSG00000003400       ENST00000286186
27 ENSG00000003400       ENST00000286186
28 ENSG00000003400       ENST00000286186
```

OMIM annotation

OMIM annotation can be retrieved with the function `getOMIM`. The arguments are the same as the function `getGene`.

```
> omim <- getOMIM(id = "203140_at", array =
+  "affy_hg_u133_plus_2", mart = mart)
> omim
  omim_id                       description ensembl_gene_id
1  109565             Lymphoma, B-cell (2) ENSG00000113916
2  109565 Lymphoma, diffuse large cell (3) ENSG00000113916
3  109565             Lymphoma, B-cell (2) ENSG00000113916
4  109565 Lymphoma, diffuse large cell (3) ENSG00000113916
  ensembl_transcript_id
1       ENST00000355918
2       ENST00000355918
3       ENST00000232014
4       ENST00000232014
```

INTERPRO protein domains

INTERPRO protein domains of the corresponding proteins can be searched with the function `getINTERPRO`. Again the arguments are the same as the function `getGene`.

```
> getINTERPRO(id = "1939_at", array = "affy_hg_u95av2",
+ mart = mart)
  interpro_id                          description
1   IPR002117                     p53 tumor antigen
2   IPR011615                      p53, DNA-binding
3   IPR010991                  p53, tetramerisation
4   IPR001472 Bipartite nuclear localization signal
  ensembl_gene_id    ensembl_transcript_id
1 ENSG00000141510          ENST00000269305
2 ENSG00000141510          ENST00000269305
3 ENSG00000141510          ENST00000269305
4 ENSG00000141510          ENST00000269305
```

Homology mapping

This function maps homologs of genes of one species to another species. To use the function one needs two instances of a mart object where two different datasets are selected e.g. hsapiens_gene_ensembl and mmusculus_gene_ensembl if you want to map homologues between these two species. Now we can look for homologs:

```
> from.mart = useMart("ensembl",
+ dataset = "hsapiens_gene_ensembl")
Checking attributes and filters ... ok
> to.mart = useMart("ensembl",
+ dataset = "mmusculus_gene_ensembl")
Checking attributes and filters ... ok
> getHomolog(id = 2, from.type = "entrezgene", to.type =
+ "refseq", from.mart = from.mart, to.mart = to.mart)

                   V1                 V2           V3
1  ENSMUSG00000030111 ENSMUST00000032203    NM_175628
2  ENSMUSG00000030111 ENSMUST00000078399
3  ENSMUSG00000059908 ENSMUST00000032228    NM_008645
4  ENSMUSG00000030131 ENSMUST00000076175 NM_001013775
5  ENSMUSG00000030131 ENSMUST00000081777    NM_008646
6  ENSMUSG00000030131 ENSMUST00000078431
7  ENSMUSG00000067797 ENSMUST00000032227
8  ENSMUSG00000030113 ENSMUST00000032206
9  ENSMUSG00000030359 ENSMUST00000032510    NM_007376
10 ENSMUSG00000030359 ENSMUST00000088106
```

Identify subsets of genes for further analysis with the `getFeature` function

The function `getFeature` is a general function to look up identifiers which pass a certain filter. A first such a filter is to look for identifiers that correspond

to genes with a given symbol. If the array argument is given then affy identifiers from that array will be returned. For retrieving other identifiers one has to specify the species and the type of identifier to retrieve.

```
> getFeature(symbol = "BRCA2",
+ array="affy_hg_u133_plus_2",mart = mart)

  hgnc_symbol affy_hg_u133_plus_2
1       BRCA2         208368_s_at
```

A second possible filter is to look for identifiers which have a certain OMIM disease term attached to them (this only works for hsapiens).

Similarly one can look for id's that have a certain GO annotation e.g. retrieve all affy id's on the hgu133plus2 array which have protein-tyrosine kinase activity.

An other filter uses the position of genes on the genome. One can query for all genes on a certain chromosome:

```
> ychrom <- getFeature(chromosome = "Y",
+  type = "entrezgene", mart = mart)
> ychrom[1:10, ]

   transcript_stable_id chr_name entrezgene
1       ENST00000331098        Y      55344
2       ENST00000326153        Y       8225
3       ENST00000328520        Y         NA
4       ENST00000300846        Y      28227
5       ENST00000361450        Y      28227
6       ENST00000334060        Y       6473
7       ENST00000344378        Y       6473
8       ENST00000329757        Y         NA
9       ENST00000361536        Y       1438
10      ENST00000359095        Y       1438
```

Or query for genes that lay in a particular region:

```
> feat = getFeature(chromosome = 1, start = 3e+05, end = 3500000,
```

```
+ type = "entrezgene", mart = mart)
> feat[1:3,]
    transcript_stable_id chr_name chr_start chr_end entrezgene
1        ENST00000327169        1    407522  408460      81399
2        ENST00000327169        1    407522  408460     135896
3        ENST00000327169        1    407522  408460      26683
```

Sequence information

The function `getSequence` retrieves the sequence given it's chromosome, start and end position.

```
> getSequence(id = "BRCA1", type = "hugo",
+  seqType = "peptide", mart = mart)

               V1 V2             V3
1 ENSG00000012048 17 protein_coding

1 MDLSALRVEEVQNVINAMQKILECPICLELIKEPVSTKCDHIFCKFCMLKLLNQKKGPSQ
CPLCKNDITKRSLQESTRFSQLVEELLKIICAFQLDTGLEYANSYNFAKKENNSPEHLKDEV
SIIQSMGYRNRAKRLLQSEPENPSLQETSLSVQLSNLGTVRTLRTKQRIQPQKTSVYIELGS
DSSEDTVNKATYCSVGDQELLQITPQGTRDEISLDSAKKAACEFSETDVTNTEHHQPSNNDL
NTTEKRAAERHPEKYQGSSVSNLHVEPCGTNTHASSLQHENSSLLLTKDRMNVEKAEFCNKS
KQPGLARSQHNRWAGSKETCNDRRTPSTEKKVDLNADPLCERKEWNKQKLPCSENPRDTEDV
PWITLNSSIQKVNEWFSRSDELLGSDDSHDGESESNAKVADVLDVLNEVDEYSGSSEKIDLL
ASDPHEALICKSERVHSKSVESNIEDKIFGKTYRKKASLPNLSHVTENLIIGAFVTEPQIIQ
ERPLTNKLKRKRRPTSGLHPEDFIKKADLAVQKTPEMINQGTNQTEQNGQVMNITNSGHENK
TKGDSIQNEKNPNPIESLEKESAFKTKAEPISSSISNMELELNIHNSKAPKKNRLRRKSSTR
HIHALELVVSRNLSPPNCTELQIDSCSSSEEIKKKKYNQMPVRHSRNLQLMEGKEPATGAKK
SNKPNEQTSKRHDSDTFPELKLTNAPGSFTKCSNTSELKEFVNPSLPREEKEEKLETVKVSN
NAEDPKDLMLSGERVLQTERSVESSSISLVPGTDYGTQESISLLEVSTLGKAKTEPNKCVSQ
CAAFENPKGLIHGCSKDNRNDTEGFKYPLGHEVNHSRETSIEMEESELDAQYLQNTFKVSKR
QSFAPFSNPGNAEEECATFSAHSGSLKKQSPKVTFECEQKEENQGKNESNIKPVQTVNITAG
FPVVGQKDKPVDNAKCSIKGGSRFCLSSQFRGNETGLITPNKHGLLQNPYRIPPLFPIKSFV
KTKCKKNLLEENFEEHSMSPEREMGNENIPSTVSTISRNNIRENVFKEASSSNINEVGSSTN
EVGSSINEIGSSDENIQAELGRNRGPKLNAMLRLGVLQPEVYKQSLPGSNCKHPEIKKQEYE
EVVQTVNTDFSPYLISDNLEQPMGSSHASQVCSETPDDLLDDGEIKEDTSFAENDIKESSAV
FSKSVQKGELSRSPSPFTHTHLAQGYRRGAKKLESSEENLSSEDEELPCFQHLLFGKVNNIP
SQSTRHSTVATECLSKNTEENLLSLKNSLNDCSNQVILAKASQEHHLSEETKCSASLFSSQC
SELEDLTANTNTQDPFLIGSSKQMRHQSESQGVGLSDKELVSDDEERGTGLEENNQEEQSMD
SNLGEAASGCESETSVSEDCSGLSSQSDILTTQQRDTMQHNLIKLQQEMAELEAVLEQHGSQ
PSNSYPSIISDSSALEDLRNPEQSTSEKAVLTSQKSSEYPISQNPEGLSADKFEVSADSSTS
KNKEPGVERSSPSKCPSLDDRWYMHSCSGSLQNRNYPSQEELIKVVDVEEQQLEESGPHDLT
ETSYLPRQDLEGTPYLESGISLFSDDPESDPSEDRAPESARVGNIPSSTSALKVPQLKVAES
AQSPAAAHTTDTAGYNAMEESVSREKPELTASTERVNKRMSMVVSGLTPEEFMLVYKFARKH
HITLTNLITEETTHVVMKTDAEFVCERTLKYFLGIAGGKWVVSYFWVTQSIKERKMLNEHDF
EVRGDVVNGRNHQGPKRARESQDRKIFRGLEICCYGPFTNMPTDQLEWMVQLCGASVVKELS
SFTLGTGVHPIVVVQPDAWTEDNGFHAIGQMCEAPVVTREWVLDSVALYQCQELDTYLIPQI
PHSHY*
```

Single Nucleotide Polymorphisms

The function `getSNP` retrieves all SNP's between a given a start and end position on a gives chromosome.. Note: make sure you have a Mart object with connections to ensembl and snp

```
> mart = useMart("snp", dataset = "hsapiens_snp")
Checking attributes and filters ... ok
> getSNP(chromosome = 8, start = 148350,
+ end = 148612, mart = mart)

        tscid  refsnp_id allele    start strand
1  TSC1723456  rs3969741    C/A   148394      1
2  TSC1421398  rs4046274    C/A   148394      1
3  TSC1421399  rs4046275    A/G   148411      1
4                rs13291    C/T   148462      1
5  TSC1421400  rs4046276    C/T   148462      1
6               rs4483971    C/T   148462      1
7              rs17355217    C/T   148462      1
8              rs12019378    T/G   148471      1
9  TSC1421401  rs4046277    G/A   148499      1
10             rs11136408    G/A   148525      1
11 TSC1421402  rs4046278    G/A   148533      1
12             rs17419210    C/T   148533     -1
13             rs28735600    G/A   148533      1
14 TSC1737607  rs3965587    C/T   148535      1
15              rs4378731    G/A   148601      1
```

Advanced data retrieval with BioMart API functions

In this section we'll discuss functions that resemble other BioMart APIs such as Martshell (see: http://www.biomart.org for more info). These functions are very general and can be used on all BioMart databases. The order in which the functions are discussed is the usual order of how you should use them.

listMarts

The `listMarts` lists the possible BioMarts where we can connect to.

```
> library(biomaRt)
```

```
> marts <- listMarts()
> marts

$biomart
[1] "wormbase" "uniprot" "msd" "snp" "vega" "ensembl"

$version
[1] "WORMBASE CURRENT (CSHL)" "UNIPROT 4-5 (EBI)"
[3] "MSD 4 (EBI)"             "SNP 37 (SANGER)"
[5] "VEGA 37 (SANGER)"        "ENSEMBL 37 (SANGER)"
```

useMart

Here we select from the list of possible BioMart databases, a BioMart that we want to use. You have to specify the full name of the BioMart e.g. "ensembl_mart_32".

```
> mart <- useMart("ensembl")
```

listDatasets

Next we want to select a specific dataset of the selected BioMart. To see which dataset is available we use the function `listDatasets`.

```
> listDatasets(mart)

                      dataset    version
1    rnorvegicus_gene_ensembl    RGSC3.4
2    scerevisiae_gene_ensembl       SGD1
3       celegans_gene_ensembl     CEL150
4  cintestinalis_gene_ensembl       JGI2
5   ptroglodytes_gene_ensembl    CHIMP1A
6      frubripes_gene_ensembl      FUGU4
7       agambiae_gene_ensembl     AgamP3
8       hsapiens_gene_ensembl     NCBI35
9        ggallus_gene_ensembl    WASHUC1
10   xtropicalis_gene_ensembl       JGI4
11        drerio_gene_ensembl     ZFISH5
12 tnigroviridis_gene_ensembl TETRAODON7
13      mmulatta_gene_ensembl   MMUL_0_1
14    mdomestica_gene_ensembl    BROADO2
15    amellifera_gene_ensembl    AMEL2.0
```

```
16 dmelanogaster_gene_ensembl      BDGP4
17     mmusculus_gene_ensembl    NCBIM34
18       btaurus_gene_ensembl   Btau_2.0
19   cfamiliaris_gene_ensembl    BROADD1
```

useDataset

To actually use a dataset we use the function `useDataset` to update our Mart object so it contains the configuration information of the dataset of interest.

```
> mart <- useDataset(dataset =
+ "hsapiens_gene_ensembl", mart = mart)
Checking attributes and filters ... ok
```

Filter, Values and Attributes

In BioMart, a filter is used to search a set of attributes that have a specified value for that filter. To explain this better lets consider the following use case. We want to get the gene symbol, chromosome name and band of the following features on the affy hgu95av2 chip: 1939_at,2082_s_at and 1454_at. In this case the attributes are gene symbol, chromosome name and band, they are the information we want to retrieve. The filter is the hgu95av2 chip and as values for this filter we use the affy identifiers we want to retrieve the information from. In BioMart a list of possible attributes that we can query for can be retrieved by using the function `listAttributes`

```
> attributes <- listAttributes(mart)
> attributes[1:10]

 [1] "aa_position"   "adf_embl"   "adf_entrezgene" "adf_go"
 [5] "adf_interpro"  "adf_omim"   "adf_pdb"        "adf_refseq"
 [9] "adf_swall"     "adf_swissprot"
```

Similarly a list of possible filters can be obtained with the function `listFilters`.

```
> filters <- listFilters(mart)
> filters[1:10]

 [1] "3downstream"             "3utr"
 [3] "5upstream"               "5utr"
 [5] "affy_hc_g110_bool"       "affy_hg_focus"
 [7] "affy_hg_focus_boolean"   "affy_hg_u133a"
 [9] "affy_hg_u133a_2"         "affy_hg_u133a_2_boolean"
```

To get the information from our example we can use the function `getBM`, using valid attributes and filter.

```
> getBM(attributes = "hgnc_symbol", filter = "affy_hg_u95av2",
+     values = c("1939_at", "2082_s_at", "1454_at", "1000_at"),
+     mart = mart)

  hgnc_symbol
1       MAPK3
2       SMAD3
3        TP53
4
5        CDH1

> getBM(attributes = c("chr_name", "band"),
+     filter = "affy_hg_u95av2", values = c("1939_at",
+     "2082_s_at", "1454_at", "1000_at"), mart = mart)

  chr_name   band
1       16  p11.2
2       15 q22.33
3       17  p13.1
4       16  q22.1
5       16  q22.1
```

As you see multiple attributes can be retrieved at once but in the current version of biomaRt there is the restriction that the attributes which are queried together, should somehow be of a similar type, e.g. chromosome band and chromosome name or e.g. allele, SNP, and frequency of snp.

Local BioMart databases

The biomaRt package can be used with a local install of a public BioMart database or a locally developed BioMart database. In order for biomaRt to recognize the database as a BioMart, make sure that the local database you create has a name conform with

```
database_mart_version
```

where database is the name of the database and version is a version number. No more underscores than the ones showed should be present in this name. A possible name is for example

```
ensemblLocal_mart_36
```

. For more information on how to install a public BioMart database see: http://www.biomart.org/install.html and follow link databases.

Bibliography

[1] S. Aerts, D. Lambrechts, S. Maity, P. Van Loo, B. Coessens, F. De Smet, L. Tranchevent, B. De Moor, P. Marynen, B. Hassan, P. Carmeliet, and Y. Moreau. Gene priorization via genomic data fusion. *Nat Biotech*, in press.

[2] Affymetrix. Statistical Algorithms Description Document, 2002.

[3] B. Alberts, D. Bray, J. Lewis, M. Raff, K. Roberts, and J.D. Watson. *Molecular Biology of the Cell*. Garland Science Publishing, 1994.

[4] J. Allemeersch, S. Durinck, R. Vanderhaeghen, P. Alard, R. Maes, K. Seeuws, T. Bogaert, K. Coddens, K. Deschouwer, P. Van Hummelen, M. Vuylsteke, Y. Moreau, J. Kwekkeboom, A.H. Wijfjes, S. May, J. Beynon, P. Hilson, and M.T. Kuiper. Benchmarking the catma microarray. a novel tool for arabidopsis transcriptome analysis. *Plant Phys*, 137:588–601, 2005.

[5] S.F. Altschul, T.L. Madden, A.A. Schaffer, J. Zhang, Z. Zhang, W. Miller, and D.J. Lipman. Gapped blast and psi-blast: A new generation of protein database search programs. *Nucleic Acids Res*, 25:3389–3402, 1997.

[6] J. L. Ashurst, C. K. Chen, J. G. Gilbert, et al. The vertebrate genome annotation (vega) database. *Nucleic Acids Res*, 33:D459–D465, 2005.

[7] C.W.B. Bachem, R.S. Van der Hoeven, S.M. De Bruijn, D. Vreugdenhil, M. Zabeau, and R.G.F. Visser. Visualization of differential gene expression using a novel method of RNA fingerprinting based on aflp: Analysis of gene expression during potato tuber development. *The plant journal*, 9:745–753, 1996.

[8] A. Barczak, M.W. Rodriguez, K. Haspers, L.L. Koth, Y.C. Tai, B. M. Bolstad, T.P. Speed, and D.J. Erle. Spotted long oligonucleotide arrays for human gene expression analysis. *Genome Res*, 13:1775–1785, 2003.

[9] D.P. Bartel. MicroRNAs: Genomics, biogenisis, mechanism, and function. *Cell*, 116:281–297, 2004.

[10] A. C. Bell, A.G. West, and G. Felsenfeld. Insulators and boundaries: Versatile regulatory elements in the eukaryotic genome. *Science*, 291:447–450, 2001.

[11] T.Z. Berardini, S. Mundodi, L. Reiser, E. Huala, M. Garcia-Hernandez, P. Zhang, L.A. Mueller, J. Yoon, A. Doyle, G. Lander, N. Moseyko, D. Yoo, I. Xu, B. Zoeckler, M. Montoya, N. Miller, D. Weems, and S.Y. Rhee. Functional annotation of the arabidopsis genome using controlled vocabularies. *Plant Phys*, 135:745–755, 2004.

[12] M. Birbeck, J. Duckett, O.G. Gudmundson, P. Kobak, E. Lenz, S. Livingstone, D. Marcus, S. Mohr, J. Pinnock, K. Visco, A. Watt, K. Williams, Z. Zaev, and N. Ozu. *Professional XML 2nd edition*. Wrox, UK, 2001.

[13] A. Bird. DNA methylation patterns and epigenetic memory. *Genes Dev*, 16(6-21):185–193, 2002.

[14] E. Birney, D. Andrews, M. Caccamo, Y. Chen, L. Clarke, G. Coates, T. Cox, F. Cunningham, V. Curwen, T. Cutts, T. Down, R. Durbin, X.M. Fernandez-Suarez, P. Flicek, S. Graf, M. Hammond, J. Herrero, K. Howe, V. Iyer, K. Jekosch, A. Kahari, A. Kasprzyk, D. Keefe, F. Kokocinski, E. Kulesha, D. London, I. Longden, C. Melsopp, P. Meidl, B. Overduin, A. Parker, G. Proctor, A. Prlic, M. Rae, D. Rios, S. Redmond, M. Schuster, I. Sealy, S. Searle, J. Severin, G. Slater, D. Smedley, J. Smith, A. Stabenau, J. Stalker, S. Trevanion, A. Ureta-Vidal, J. Vogel, S. White, C. Woodwark, and T.J. Hubbard. Ensembl 2006. *Nucleic Acids Res*, 34:D556–D561, 2006.

[15] M. Boehm and F. Slack. A developmental timing microrna and its target regulate life span in c. elegans. *Science*, 310:1911–1913, 2005.

[16] B.M. Bolstad, R.A. Irizarry, M. Astrand, and T.P. Speed. A comparison of normalized methods for high density oligonucleotide array data based on variance and bias. *Bioinformatics*, 19(2):185–193, 2003.

[17] D.C. Boyes, R. Zayed, A.M. andAscenzi, McCaskill A.J., N.E. Hoffman, K.R. Davis, and J. Gorlach. Growth stage-based phenotypic analysis of arabidopsis: a model for high throughput functional genomics in plants. *Plant Cell*, 13:1499–1510, 2001.

[18] A. Brazma, P. Hingamp, J. Quackenbush, G. Sherlock, P. Spellman, C. Stoeckert, J. Aach, W. Ansorge, C.A. Ball, H. Causton, T. Gaasterland, P. Glenisson, F.C. Holstege, I.F. Kim, V. Markowitz, J.C. Matese, H. Parkinson, A. Robinson, U. Sarkans, S. Schulze-Kremer, J. Stewart, R. Taylor, J. Vilo, and M. Vingron. Minimum information about a microarray experiment (MIAME)-toward standards for microarray data. *Nat Genet*, 29(4):365–371, Dec 2001.

[19] A. Brazma, H. Parkinson, U. Sarkans, M. Shojatalab, J. Vilo, N. Abeygunawardena, E. Holloway, M. Kapushesky, P. Kemmeren, G.G. Lara, A. Oezcimen, P. Rocca-Serra, and S.A. Sansone. Arrayexpress–a public repository for microarray gene expression data at the EBI. *Nucleic Acids Res*, 31(1):68–71, Jan 2003.

[20] E. Camon, M. Magrane, D. Barrell, V. Lee, E. Dimmer, J. Maslen, D. Binns, N. Harte, R. Lopez, and R. Apweiler. The gene ontology annotation (goa) database: sharing knowledge in uniprot with gene ontology. *Nucleic Acids Res*, 32:D262–266, 2004.

[21] J.F. Chen, E.M. Mandel, J.M. Thomson, Q. Wu, T.E. Callis, S.M. Hammond, F.L. Conlon, and D.Z. Wang. The role of microrna-1 and microrna-133 in skeletal muscle proliferation and differentiation. *Nat Genet*, epub:1–1, 2005.

[22] X. Chen. A microRNA as a translational repressor of apetala2 in arabidopsis flower development. *Science*, 303:2022–2025, 2004.

[23] Gene Ontology Consortium. The gene ontology (go) database and informatics resource. *Nucleic Acids Res*, 32:258–261, 2004.

[24] L. Cope, R Irizarry, H. Jaffee, Z. Wu, and T.P. Speed. A benchmark for affymetrix genechip expression measures. *Bioinformatics*, 20(3):323–331, 2004.

[25] L. DeFrancesco. Journal trio embraces MIAME. *The Scientist*, Oct 2002.

[26] T. Denayer. Functional study of the canonical wnt pathway using transgenic tools in *Xenopus laevis*. *PhD thesis, University of Ghent*, 2005.

[27] J. DeRisi, L. Penland, P.O. Brown, M.L Bittner, P.S. Meltzer, M. Ray, Y. Chen, Y.A. Su, and J.M. Trent. Use of a cdna microarray to analyse gene expression patterns in human cancer. *Nat Genet*, 14:457–460, 1996.

[28] A.M. Dudley, J. Aach, M.A. Steffen, and G.M. Church. Measuring absolute expression with microarrays with a calibrated reference sample and an extended signal intensity range. *Proc Natl Acad Sci USA*, 99(11):7554–7559, 2002.

[29] S. Dudoit, J.P. Schaffer, and J.C. Boldrick. Multiple hypothesis testing in microarray experiments. *Statistical Science*, 18:71–103, 2003.

[30] S. Dudoit and J. H. H. Yang. R packages for the analysis of cDNA microarray data. In G. Parmigiani, E. Garrett, R. Irizarry, and S. Zeger, editors, *The analysis of gene expression data: methods and software*, pages 313–341. Springer, NY, 2003.

[31] D.J. Duggan, M. Bittner, Y. Chen, P. Meltzer, and J. Trent. Expression profiling using cDNA microarrays. *Nat Genet*, 21(1Suppl):10–14, 1999.

[32] S. Durinck, A. Allemeersch, V.J. Carey, Y. Moreau, and B De Moor. Importing mage-ml format microarray data into bioconductor. *Bioinformatics*, 20(18):3641–3642, 2004.

[33] S. Durinck, Y. Moreau, A. Kasprzyk, S. Davis, B. De Moor, A. Brazma, and W. Huber. Biomart and bioconductor: a powerful link between biological databases and microarray data analysis. *Bioinformatics*, 21:3439–3440, 2005.

[34] C.E. Eckfeldt, E.M. Mendenhall, and C.M. Verfaillie. The molecular repertoire of the 'almighty' stem cell. *Nat Rev Mol Cell Biol*, 6(9):726–737, 2005.

[35] R. Edgar, M. Domrachev, and A.E. Lash. Gene expression omnibus: NCBI gene expression and hybridization array data repository. *Nucleic Acids Res*, 30:207–210, 2002.

[36] M.B. Eisen, P.T. Spellman, P.O. Brown, and D. Botstein. Cluster analysis and display of genome-wide expression patterns. *Proc Natl Acad Sci USA*, 95(25):14863–14868, 1998.

[37] H.G. Frohnhofer and C. Nusslein-Volhard. Organization of anterior pattern in the drosophila embryo by the maternal gene bicoid. *Nature*, 324:120–125, 1986.

[38] A.P. Gasch, P.T. Spellman, C.M. Kao, O. Carmel-Harel, M.B. Eisen, G. Storz, D. Botstein, and P.O. Brown. Genomic expression programs in the response of yeast cells to environmental changes. *Mol Biol Cell*, 11(12):4241–4257, 2000.

[39] L. Gautier, L. Cope, B. Bolstad, and R. Irizarry. affy-analysis of affymetrix genechip data at the probe level. *Bioinformatics*, 20(3):307–315, 2004.

[40] E.R. Gavin and R. Lehman. Localization of nanos rna controls embryonic polarity. *Cell*, 71(2):301–313, 1992.

[41] R. Gentleman, V. Carey, W. Huber, R. Irizarry, and S. Dudoit. *Bioinformatics and Computational Biology Solutions Using R and Bioconductor*. Springer, NY, 2005.

[42] R. Gentleman and V. J. Carey. Visualization and annotation of genomic experiments. In G. Parmigiani, E. Garrett, R. Irizarry, and S. Zeger, editors, *The analysis of gene expression data: methods and software*, pages 313–341. Springer, NY, 2003.

[43] R.C. Gentleman, V.J. Carey, D.M. Bates, B. Bolstad, M. Dettling, S. Dudoit, B. Ellis, L. Gautier, Y. Ge, J. Gentry, K. Hornik, T. Hothorn, W. Huber, S. Iacus, R. Irizarry, F. Leisch, C. Li, M. Maechler, A. Rosinni, G. Sawitzki, C. Smith, G. Smyth, L. Tierney, J.Y.H. Yang, and J. Zhang. Bioconductor: open software development for computational biology and bioinformatics. *Genome Biol*, 5(10):R80, 2004.

[44] T.R. Golub, D.K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J.P. Mesirov, H. Coller, M.L. Loh, J.R. Downing, M.A. Caligiuri, C.D. Bloomfield, and E.S. Lander. Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science*, 286(5439):531–537, 1999.

[45] S.C. Hammond, E. Bernstein, D. Beach, and G.J. Hannon. An RNA-directed nuclease mediates posttranscriptional gene silencing in drosophila cells. *Nature*, 404:293–296, 2000.

[46] D. Hancock, M. Wilson, G. Velarde, N. Morrison, A. Hayes, H. Hulme, A.J. Wood, K. Nashar, D.B. Kell, and Brass A. maxdload2 and maxdbrowse: standards-compliant tools for microarray experimental annotation, data management and dissemination. *BMC Bioinformatics*, 6:264, 2005.

[47] J.G. Herman and S.B. Baylin. Gene silencing in cancer in association with promoter hypermethylation. *N Eng J Med*, 349:2042–2054, 2003.

[48] H. Heyman, P. Kauraniemi, S. Hautaniemi, M. Wolf, S. Mousses, E. Rozenblum, M. Ringner, G. Sauter, O. Monni, A. Elkahloun, O. Kallioniemi, and A. Kallioniemi. Impact of dna amplification on gene expression patterns in breast cancer. *Cancer Res*, 62:6240–6245, 2002.

[49] P. Hilson, J. Allemeersch, T. Altmann, S. Aubourg, A. Avon, J. Beynon, R.P. Bhalerao, F. Bitton, M. Caboche, B. Cannoot, V. Chardakov, C. Cognet-Holliger, V. Colot, M. Crowe, C. Darimont, S. Durinck, H. Eickhoff, A.F. de Longevialle, E.E. Farmer, M. Grant, M.T. Kuiper, H. Lehrach, C. Leon, A. Leyva, J. Lundeberg, C. Lurin, Y. Moreau, W. Nietfeld, J. Paz-Ares, P. Reymond, P. Rouze, G. Sandberg, M.D. Segura, C. Serizet, A. Tabrett, L. Taconnat, V. Thareau, P. Van Hummelen, S. Vercruysse, M. Vuylsteke, M. Weingartner, P.J. Weisbeek, V. Wirta, F.R. Wittink, M. Zabeau, and I. Small. Versatile gene-specific sequence tags for arabidopsis functional genomics: transcript profiling and reverse genetics applications. *Genome Res*, 14:2176–2189, 2004.

[50] T. Hubbard, D. Andrews, M. Caccamo, G. Cameron, Y. Chen, M. Clamp, L. Clarke, G. Coates, T. Cox, F. Cunningham, V. Curwen, T. Cutts, T. Down, R. Durbin, X.M. Fernandez-Suarez, J. Gilbert, M. Hammond, J. Herrero, H. Hotz, K. Howe, V. Iyer, K. Jekosch, A. Kahari, A. Kasprzyk, D. Keefe, S. Keenan, F. Kokocinsci, D. London, I. Longden, G. McVicker, C. Melsopp, P. Meidl, S. Potter, G. Proctor, M. Rae, D. Rios, M. Schuster, S. Searle, J. Severin, G. Slater, D. Smedley, J. Smith, W. Spooner, A. Stabenau, J. Stalker, R. Storey, S. Trevanion, A. Ureta-Vidal, J. Vogel, S. White, C. Woodwark, and E. Birney. Ensembl 2005. *Nucleic Acids Res*, 33:D447–D453, 2005.

[51] W. Huber, A. Heydebreck, H. Sueltmann, A. Poutska, and M. Vingron. Variance stabilization applied to microarray data calibration and to the quantification of differential expression. *Bioinformatics*, 18(Supplement1):S96–S104, 2002.

[52] T.R. Hughes, M. Mao, A.R. Jones, J. Burchard, M.J. Marton, K.W. Shannon, S.M. Lefkowitz, M. Ziman, J.M. Schelter, M.R. Meyer, Kobayashi S, Davis C., H. Dai, Y.D. He, S.B. Stephaniants, G. Cavet, W.L. Walker, A. West, E. Coffey, D.D. Shoemaker, R. Stoughton, A.P. Blanchard, S.H. Friend, and P.S. Linsley. Expression profiling using microarrays fabricated by an ink-jet oligonucleotide synthesizer. *Nat Biotechnol*, 19:342–347, 2001.

[53] T.R. Hughes, M.J. Marton, A.R. Jones, C.J. Roberts, R. Stoughton, C.D. Armour, H.A. Bennett, E. Coffey, H. Dai, Y.D. He, M.J. Kidd, A.M. King, M.R. Meyer, D. Slade, P.Y. Lum, S.B. Stepaniants, D.D. Shoemaker, D. Gachotte, K. Chakraburtty, J. Simon, M. Bard, and S.H. Friend. Functional discovery via a compendium of expression profiles. *Cell*, 102:109–126, 2000.

[54] M. Hulskamp, C. Pfeifle, and D. Tautz. A morphogenetic gradient of hunchback protein organizes the expression of the gap genes kruppel and knirps in the early drosophila embryo. *Nature*, 346(6284):577–580, 1990.

[55] R. Ihaka and R. Gentleman. R: A language for data analysis and graphics. *Journal of Computational and Graphical Statistics*, 5(3):299–314, 1996.

[56] J. Ihmels, G. Friedlander, S. Bergmann, O. Sarig, Y. Ziv, and N. Barkai. Revealing modular organization in the yeast transcriptional network. *Nat Genet*, 31:370–377, 2002.

[57] K. Ikeo, J. Ishi-i, T. Tamura, T. Gojobori, and Y. Tateno. CIBEX: center for information biology gene expression database. *Curr Biol*, 326:1079–1082, 2003.

[58] Arabidopsis Genome Initiative. Analysis of the genome sequence of the flowering plant arabidopsis thaliana. *Nature*, 408:796–815, 2000.

[59] The international HapMap consortium. The international hapmap project. *Nature*, 426:789–796, 2003.

[60] The international hapmap consortium. A haplotype map of the human genome. *Nature*, 437:1299–1320, 2005.

[61] R. Irizarry, L. Gautier, and E. Cope. An R package for oligonucleotide array analysis. In G. Parmigiani, E. Garrett, R. Irizarry, and S. Zeger,

editors, *The analysis of gene expression data: methods and software*, pages 313–341. Springer, NY, 2003.

[62] R.A. Irizarry, B. Hobbs, F. Collin, Y.D. Beazer-Barclay, K.J. Antonellis, U. Scherf, and T.P. Speed. Exploration, normalization, and summaries of high-density oligonucleotide array probe level data. *Biostatistics*, 4:249–264, 2003.

[63] R.A. Irizarry, Z. Wu, and H.A. Jaffee. Comparison of affymetrix genechip expression measures. *Bioinformatics*, Epub ahead of print, 2006.

[64] R.J. Johnston and O. Hobert. A microrna controlling left/right neuronal asymmetry in ceanorhabditis elegans. *Nature*, 426:845–849, 2003.

[65] M.D. Kane, T.A. Jatkoe, C.R. Stumpf, J. Lu, J.D. Thomas, and S.J. Madore. Assessment of the sensitivity and specificity of oligonucleotide (50mer) microarrays. *Nucleic Acids Res*, 28:4552–4557, 2000.

[66] C.D. Kaplan, L. Laprade, and F. Winston. Transcription elongation factors repress transcription initiation from cryptic sites. *Science*, 301:1096–1099, 2003.

[67] M. Kapushesky, P. Kemmeren, A.C. Culhane, S. Durinck, J. Ihmels, C. Korner, M. Kull, A. Torrente, U. Sarkans, J. Vilo, and A. Brazma. Expression profiler: next generation-an online platform for analysis of microarray data. *Nucleic Acids Res*, 32:W465–W470, 2004.

[68] A. Kasprzyk, D. Keefe, D. Smedley, D. London, W. Spooner, C. Melsopp, M. Hammond, P. Rocca-Serra, T. Cox, and E. Birney. Ensmart: A generic system for fast and flexible access to biological data. *Genome Res*, 14:160–169, 2004.

[69] K. Kerr, M. Martin, and G. Churchill. Analysis of variance for gene expression microarray data. *J Comput Biol*, 7:819–837, 2000.

[70] P.J. Killion, G. Sherlock, and V.R. Iyer. The longhorn array database (LAD): an open-source, MIAME compliant implementation of the stanford microarray database (SMD). *BMC Bioinformatics*, 4(32), 2003.

[71] R.D. Komerg and Y. Lorch. Twenty-five years of the nucleosome, fundamental particle of the eukaryote chromosome. *Cell*, 98:285–294, 1999.

[72] S. Kuersten and E.B. Goodwin. The power of the 3' UTR: translational control and development. *Nat Genet*, 4:626–637, 2003.

[73] S.K. Kurdistani, S. Tavazoie, and M. Grunstein. Mapping global histone acetylation patterns to gene expression. *Cell*, 117(6):721–733, 2004.

[74] M. Lagos-Quintana, R. Rauhut, J. Meyer, A. Brokhardt, and T. Tuschl. New microRNAs from mouse and human. *RNA*, 9:175–179, 2003.

[75] R.C. Lee, R.L. Feinbaum, and V. Ambros. The c. elegans heterochronic gene lin-4 encodes small rnas with antisense complementarity to lin-14. *Cell*, 75:843–854, 1993.

[76] B. Lemon and R. Tjian. Orchestrated response: a symphony of transcription factors for gene control. *Genes Dev*, 14:2551–2569, 2000.

[77] C. Li and W.H. Wong. Model-based analysis of oligonucleotide arrays: Expression index computation and outlier detection. *Proc Natl Acad Sci USA*, 98(1):31–36, 2001.

[78] P. Liang and A.B. Pardee. Differential display of eukayotic messenger RNA by means of the polymerase chain reaction. *Science*, 257:967–970, 1992.

[79] J.D. Lieb and Clarke N.D. Control of transcription through intergenic patterns of nucleosome composition. *Cell*, 123:1187–1190, 2005.

[80] R.J. Lipshutz, P.A. Stephen, T. Gingeras, and D.J. Lockhart. High density synthetic oligonucleotide arrays. *Nat Genet*, 21 (1 Suppl):20–24, 1999.

[81] D.J. Lockhart, H. Dong, M.C. Byrne, M.T. Follettie, M.V. Gallo, M.S. Chee, M. Mittmann, C. Wang, M. Kobayashi, H. Horton, and E.L. Brown. Expression monitoring by hybridization to high-density oligonucleotide arrays. *Nat Biotechnol*, 14:1675–1680, 1996.

[82] C. Y. Logan and R. Nusse. The Wnt signaling pathway in development and disease. *Annu Rev Cell Dev Biol*, 20:781–810, 2004.

[83] M. Maurer, R. Molidor, A. Sturn, J. Hartler, H. Hackl, G. Stocker, A. Prokesch, M. Scheideler, and Z. Trajanoski. MARS: microarray analysis, retrieval, and storage system. *BMC Bioinformatics*, 6:101, 2005.

[84] J Mellor. The dynamics of chromatin remodeling at promoters. *Mol Cell*, 19(2):147–157, 2005.

[85] G. Mukherjee, N. Abeygunawardena, H. Parkinson, S. Contrino, S. Durinck, A. Farne, E. Holloway, P. Lilja, Y. Moreau, A. Oezcimen, T. Rayner, A. Sharma, A. Brazma, U. Sarkans, and M. Shojatalab. Plant-based microarray data at the european bioinformatics institute. introducing atmiamexpress, a submission tool for arabidopsis gene expression data to arrayexpress. *Plant Phys*, 139(2):632–636, 2005.

[86] M. Navarange, L. Game, D. Fowler, V. Wadekar, H. Banks, N. Cooley, F. Rahman, J. Hinshelwood, P. Broderick, and H.C. Causton. Mimir: a comprehensive solution for storage, annotation and exchange of microarray data. *BMC Bioinformatics*, 6:268, 2005.

[87] H. Parkinson, U. Sarkans, M. Shojatalab, N. Abeygunawardena, S. Contrino, R. Coulson, A. Farne, G.G. Lara, E. Holloway, M. Kapushesky, P. Lilja, G. Mukherjee, A. Oezcimen, T. Rayner, P. Rocca-Serra, A. Sharma, S. Sansone, and A. Brazma. Arrayexpress–a public repository for microarray gene expression data at the ebi. *Nucleic Acids Res*, 33 (Database issue):D553–555, 2005.

[88] C.M. Perou, T. Sorlie, M. Eisen, M. van de Rijn, S.S. Jeffrey, C.A. Rees, J.R. Pollack, D.T. Ross, H. Johnson, L.A. Akslen, O. Fluge, A. Pergamenschikov, C. Williams, S.X. Zhu, P.E. Lonning, A. Borresen-Dale, P.O. Brown, and D. Botstein. Molecular portraits of human breast tumours. *Nature*, 406(6797):747–752, 2000.

[89] D.R. Rhodes, T.R. Barrette, M.A. Rubin, D. Ghosh, and A.M. Chinnaiyan. Meta-analysis of microarrays: Interstudy validation of gene expression profiles reveals pathway dysregulation in prostate cancer. *Cancer Research*, 62:4427–4433, 2002.

[90] D.R. Rhodes, J. Yu, K. Shanker, N. Deshpande, R. Varambally, D. Ghosh, D. Barrette, A. Pandey, and A.M. Chinnaiyan. Large-scale meta-analysis of cancer microarray data identifies common transcriptional profiles of neaplastic transformation and progression. *Proc Natl Acad Sci USA*, 101(25):9309–9314, 2004.

[91] P. Roepman, L.F. Wessels, N. Kettelarij, P. Kemmeren, A.J. Miles, P. Lijnzaad, M.G. Tilanus, R. Koole, G.J. Hordijk, P.C. van der Vliet, M.J. Reinders, P.J. Slootweg, and F.C. Holstege. An expression profile for

diagnosis of lymph node metastases from primary head and neck squamous cell carcinomas. *Nat Genet*, 37(2):182–186, 2005.

[92] S. Rozen and H. Skaletsky. *Primer3 on the WWW for general users and for biologists programmers*, volume 32. Humana Press, Totowa, NJ, 2000. In *Bioinformatics Methods and protocols*.

[93] L.H. Saal, C. Troein, J. Vallon-Christersson, S. Gruvberger, A Borg, and C. Peterson. Bioarray software environment: A platform for comprehensive management and analysis of microarray data. *Genome Biol*, 3(8):software0003.1–0003.6, 2002.

[94] M. Schena, D. Shalon, R.W. Davis, and P.O. Brown. Quantitative monitoring of gene expression patterns with a complementary DNA microarray. *Science*, 270:467–470, 1995.

[95] M. Schena, D. Shalon, R. Heller, A. Chai, P.O. Brown, and R.W. Davis. Parallel human genome analysis: Microarray-based expression monitoring of 1000 genes. *Proc Natl Acad Sci USA*, 93:10614–10619, 1996.

[96] M. Schmid, T.S. Davison, S.R. Henz, U.J. Pape, M. Demar, M. Vingron, B. Scholkopf, D. Weigel, and J.U. Lohmann. A gene expression map of *Arabidopsis thaliana* development. *Nat Genet*, 37(5):501–506, 2005.

[97] G.M. Schratt, F. Tuebing, E.A. Nigh, C.G. Kane, M.E. Sabatini, M. Kiebler, and E. Greenberg. A brain-specific microRNA regulates dendritic spine development. *Nature*, 439:283–289, 2006.

[98] M.A. Schwabish and K. Struhl. Evidence for eviction and rapid deposition of histones upon transcriptional elongation by RNA polymerase II. *Mol Cell Biol*, 24:10111–10117, 2004.

[99] E.M. Schwarz, I. Antoshechkin, C. Bastiani, T. Bieri, D. Blasiar, P. Canaran, J. Chan, N. Chen, W.J. Chen, P. Davis, T.J. Fiedler, L. Girard, T.W. Harris, E.E. Kenny, R. Kishore, D. Lawson, R. Lee, H.M. Muller, C. Nakamura, P. Ozersky, A. Petcherski, A. Rogers, W. Spooner, M.A. Tuli, K. Van Auken, D. Wang, R. Durbin, J. Spieth, L.D. Stein, and P.W. Sternberg. Wormbase: better software, richer content. *Nucleic Acids Res*, 34(Database issue):D475–D478, 2006.

[100] J. Sebat, B. Lakshmi, J. Troge, J. Alexander, J. Young, P. Lundin, S. Maner, H. Massa, M. Walker, M. Chi, N. Navin, R. Lucito, J. Healy, J. Hicks, K. Ye, A. Reiner, T.C. Gilliam, B. Trask, N. Patterson,

A. Zetterberg, and M. Wigler. Large-scale copy number polymorphism in the human genome. *Science*, 305(5683):525–528, 2004.

[101] K. Shedden, W. Chen, R. Kuick, D. Ghosh, J. Macdonald, K.R. Cho, T.J. Giordano, S.B. Gruber, E.R. Fearon, J. M. G. Taylor, and S. Hanash. Comparison of seven methods for producing affymetrix expression scores based on false discovery rates in disease profiling data. *BMC Bioinformatics*, 6(1):26, 2005.

[102] S. T. Sherry, M. H. Ward, M. Kholodov, J. Baker, L. Phan, E.M. Smigielski, and K. Sirotkin. dbSNP:the NCBI database of genetic variation. *Nucleic Acids Res*, 29(1):308–311, 2001.

[103] S. Singh-Gasson, R.D. Green, Y. Yue, C. Nelson, F. Blattner, M.R. Sussman, and F. Cerrina. Maskless fabrication of light-directed oligonucleotide microarrays using a digital micromirror array. *Nat Biotechnol*, 17:974–978, 1999.

[104] Smyth2004. Linear models and emperical bayes methods for assessing differential expression in microarray experiments. *Statistical Applications in Genetics and Molecular Biology*, 3:Article 3, 2004.

[105] C. G. Son, S. Bilke, S. Davis, B.T. Greer, J.S. Wei, C.C. Whiteford, Q.R. Chen, N. Cenacchi, and J. Khan. Database of mRNA gene experssion profiles of multiple human organs. *Genome Res*, 15:443–450, 2005.

[106] P. T. Spellman, M. Miller, J. Stewart, C. Troup, U. Sarkans, S. Chervitz, D. Bernhart, G. Sherlock, C. Ball, M. Lepage, M. Swiatek, W.L. Marks, J. Goncalves, S. Markel, D. Iordan, M. Shojatalab, A. Pizarro, J. White, R. Hubley, E. Deutsch, M. Senger, B.J. Aronow, A. Robinson, D. Bassett, C.J. Jr Stoeckert, and Brazma A. Design and implementation of microarray gene expression markup language (MAGE-ML). *Genome Biol*, 3(9):RESEARCH0046, 2002.

[107] P.T. Spellman, G. Sherlock, M.O. Zhang, V.R. Iyer, K. Anders, M.B. Eisen, P.O. Brown, D. Botstein, and B. Futcher. Comprehensive identification of cell cycle-regulated genes of the yeast saccharomyces cerevisiae by microarray hybridization. *Mol Biol Cell*, 9(12):3273–3297, 1998.

[108] C.J. Stoeckert, H. Causton, and A.B. Ball. Microarray databases: standards and ontologies. *Nat Genet*, 32 (Suppl 1):469–473, 2002.

[109] J.D. Storey and R. Tibshirani. Statistical significance for genomewide studies. *Proc Natl Acad Sci USA*, 100(16):9440–9445, 2003.

[110] B.E. Stranger, M.S. Forrest, A.G. Clark, M.J. Minichiello, S. Deutsch, R. Lyle, S. Hunt, B. Kahl, S.E. Antonarakis, S. Tavare, P. Deloukas, and E.T. Dermitzakis. Genome-wide associations of gene expression variation in humans. *PLoS Genet*, 1(6):e78, 2005.

[111] A.I. Su, T. Wiltshire, S. Batalov, H. Lapp, K.A. Ching, D. Block, J. Zhang, R. Soden, M. Hayakawa, G. Kreiman, M.P. Cooke, J.R. Walker, and J.B. Hogenesch. A gene atlas of the mouse and human protein-encoding transcriptomes. *Proc Natl Acad Sci USA*, 101:6062–6067, 2004.

[112] J.Q. Svejstrup. Transcription. histones face the FACT. *Science*, 301:1053–1055, 2003.

[113] V. Thareau, P. Dehais, C. Serizet, P. Hilson, P. Rouze, and S. Aubourg. Automatic design of gene-specific sequence tags for genome-wide functional studies. *Bioinformatics*, 19:2191–2198, 2003.

[114] V. Tusher, R. Tibshirani, and G. Chu. Significance analysis of microarrays applied to the ionizing radioation response. *Proc Natl Acad Sci USA*, 98:5116–5121, 2001.

[115] J. Van de Peppel, P. Kemmeren, H. van Bakel, M. Radonjic, D. van Leenen, and F. Holstege. Monitoring global messenger RNA changes in externally controlled microarray experiments. *EMBO reports*, 4(4):387–393, 2003.

[116] L. van't Veer, H. Dai, M.J. van de Vijver, Y.D. He, A.A. Hart, M. Mao, H.L. Peterse, K. van der Kooy, M.J. Marton, A.T. Witteveen, G.J. Schreiber, R.M. Kerkhoven, C. Roberts, P.S. Linsley, R. Bernards, and S.H. Friend. Gene expression profiling predicts clinical outcome of breast cancer. *Nature*, 415(6871):530–536, 2002.

[117] V.E. Velculescu, L. Zhang, B. Vogelstein, and K.W. Kinzler. Serial analysis of gene expression. *Science*, 270:484–487, 1995.

[118] D.H. Ware, P. Jaiswal, J. Ni, I.V. Yap, X. Pan, K.Y. Clark, L. Teytelman, S.C. Schmidt, W. Zhao, K. Chang, S. Cartinhour, L.D. Stein, and S.R. McCouch. Gramene, a tool for grass genomics. *Plant Phys*, 130(4):1606–1613, 2002.

[119] J.D. Watson and F.H. Crick. A structure for deoxyribose nucleic acid. *Nature*, 171:737–738, 1953.

[120] C. Workman, L.J. Jensen, H. Jarmer, R. Berka, L. Gautier, H.B. Nielsen, H. Saxild, C. Nielsen, S. Brunak, and S. Knudson. A new non-linear normailzation method for reducing variability in dna microarray experiments. *Genome Biol*, 3(9):research0048, 2002.

[121] C.H. Wu, R. Apweiler, A. Bairoch, D.A. Natale, W.C. Barker, B. Boeckmann, S. Ferro, E. Gasteiger, H. Huang, R. Lopez, M. Magrane, M.J. Martin, R. Mazumder, C. O'Donovan, N. Redaschi, and B. Suzek. The universal protein resource (uniprot): an expanding universe of protein information. *Nucleic Acids Res*, 34(Database issue):D187–191, 2006.

[122] Z. Wu, R. Irizarry, R. Gentleman, F. M. Murillo, and Spencer. F. A model based background adjustment for oligonucleotide expression arrays. *Journal of the American Statistical Association*, 99(468):909–917, 2004.

[123] Y.H. Yang, S. Dudoit, P. Luu, D.M. Lin, V. Peng, J. Ngai, and T.P. Speed. Normalization for cDNA microarray data: a robust composite method addressing single and multiple slide systematic variation. *Nucleic Acids Res*, 30(4):e15, 2002.

[124] Y.H. Yang and T.P. Speed. Design issues for cDNA microarray experiments. *Nat Rev Genet*, 3(8):579–588, 2002.

[125] S. O. Zakharin, K. Kim, T. Mehta, L. Chen, S. Barnes, K.E. Scheirer, R.S. Parrish, D.B. Allison, and G.P. Page. Sources of variation in affymetrix microarray experiments. *BMC Bioinformatics*, 6:214–225, 2005.

[126] J. Zhang, V. Carey, and R. Gentleman. An extensible application for assembling annotation for genomic data. *Bioinformatics*, 19(1):155–156, 2003.

[127] P. Zimmermann, M. Hirsch-Hoffmann, L. Henning, and W. Gruisem. GENEVESTIGATOR. Arabidopsis Microarray Database and Analysis Toolbox. *Plant Phys*, 136:2621–2632, 2004.

Index

Arabidopsis thaliana, 60

ArrayExpress, 49
At-endeavour, 100
AtGenExpress, 77

BioMart, 85
biomaRt, 87
Boyes growth stages, 66

CAGE, 59
CATMA array, 62
central dogma of biology, 14

dbSNP, 84
DNA, 12

Endeavour, 99
Ensembl, 82
EP, 96
EP:NG, 96
expression compendia, 2

fold-change, 33
functional genomics, 2

GCRMA, 32
gene expression, 14
GO, 83
Gramene, 85
GST, 62

HapMap, 84

JNI, 55

MA-plot, 34
MAGE-ML, 47
MAGE-OM, 40
MAS5.0, 29
MBEI, 30
MGED, 38
MIAME, 38
MIAMExpress, 49
miRNA, 17
mRNA, 12

OMIM, 83

PO, 64
promoter, 14
proteins, 12
PTGS, 17

qPCR, A-10
quelling, 17

reference design, 67
RMA, 31
RMAGEML, 52
RNAi, 17

SAM, 34
SNP, 84
SVG, 96

transcription, 12
transcription factor, 14
transcriptome, 2

translation, 14

UML, 40
UniProt, 84

VEGA, 84
volcano plot, 34
VSN, 32

Wnt, A-1

XML, 45
    DTD, 47

Curriculum vitae

Steffen Durinck was born in Ghent, Belgium, on February 8th 1977. He obtained a Master of Science in Engineering in Cellular and Gene Biotechnology (Bio-ingenieur Cel- en Genbiotechnologie) with magna cum laude distinction at the University of Ghent in 2000. His master thesis 'The tumor suppressor gene E-cadherin: study of LOH and E-cadherin mediated differential gene expression' was performed at the department of Molecular Cell Biology under supervision of Dr. G. Berx and Prof. Dr. F. van Roy. After graduating he started a Ph.D. study at the department of Molecular Cell Biology but after one year of research, his scientific interests changed towards the emerging field of bioinformatics. He terminated his Ph.D. post and in 2002 obtained a Master of Science in Bioinformatics at the K.U.Leuven with magna cum laude distinction. In 2002 he joined ESAT-SCD and started his Ph.D. study in the field of bioinformatics under supervision of Prof. Dr. Bart De Moor and Prof. Dr. Yves Moreau. During his Ph.D. study, he obtained a Marie-Curie fellowship to do a four month internship at the European Bioinformatics Institute in Hinxton, Cambridge, UK. Later during his Ph.D. he returned for a second one year internship at EBI were he joined the group of Dr. Alvis Brazma. In 2005 he was an invited speaker at the Bioconductor meeting in Seattle, USA, to give a practical on the biomaRt software package.