Analysis and Visualization of Gene Expression Data

Dissertation
der Mathematisch-Naturwissenschaftlichen Fakultät
der Eberhard Karls Universität Tübingen
zur Erlangung des Grades eines
Doktors der Naturwissenschaften
(Dr. rer. nat.)

vorgelegt von
Stephan Hermann Georg Symons
aus Duisburg

Tübingen
2011

Tag der mündlichen Qualifikation: 30. 11. 2011
Dekan: Prof. Dr. Wolfgang Rosenstiel
1. Berichterstatter: Dr. Kay Nieselt
2. Berichterstatter: Prof. Dr. Michael Kaufmann

Contents

Table of Contents i

List of Figures vi

Zusammenfassung vii

Abstract ix

List of Abbreviations xiii

1 Introduction 1
- 1.1 Outline . . . 3

2 Analysis of Gene Expression 5
- 2.1 Analysis of Gene Expression . . . 5
  - 2.1.1 Gene Expression . . . 5
  - 2.1.2 Regulation of Gene Expression . . . 7
  - 2.1.3 Measuring Gene Expression . . . 8
- 2.2 The Microarray Technology . . . 9
  - 2.2.1 Principle . . . 9
  - 2.2.2 Technologies . . . 10
  - 2.2.3 Development and Standards . . . 11
  - 2.2.4 Experimental Workflow . . . 12
  - 2.2.5 Microarrays for Genomic Analysis . . . 12
- 2.3 RNA-Seq . . . 13
  - 2.3.1 Next-Generation Sequencing . . . 13
  - 2.3.2 Normalization and Analysis . . . 14
- 2.4 Metabolomics . . . 15
- 2.5 Gene Expression Data Analysis . . . 16
  - 2.5.1 Microarray Data Normalization . . . 16
  - 2.5.2 Statistical Analysis . . . 19
  - 2.5.3 Cluster Analysis . . . 20
  - 2.5.4 Higher Level Analysis . . . 21

3 Data Visualization 23

3.1 Data Visualization 23
  3.1.1 General Concepts 26
  3.1.2 Exploratory Data Analysis 28
  3.1.3 Visual Analytics 28
3.2 Visualization Plots 29
  3.2.1 Scatterplots 30
  3.2.2 Parallel Coordinates 31
  3.2.3 Heat Maps 32
  3.2.4 Box Plots 33
  3.2.5 Bar Plots and Histograms 34
  3.2.6 Sequence Logos 35
  3.2.7 Radial Plots 36
3.3 Graphs and Networks 37
  3.3.1 Selected Problems and Algoritms 37
  3.3.2 Graph Drawing 39
3.4 Visualization of Biological Pathways 40
  3.4.1 Data Formats 41
  3.4.2 SBGN 43
3.5 Mayday 45
  3.5.1 Data Model 45
  3.5.2 Extensions to Mayday 46
  3.5.3 Visualization Framework 47

4 Analysis and Visualization of Resequencing Microarrays 49
4.1 Analysis of Resequencing Data 49
  4.1.1 Base Calling 50
4.2 ResqMi 53
  4.2.1 Application: Comparison of Base Calling Algorithms 58
4.3 Discussion and Outlook 60
  4.3.1 Possible New Applications 61

5 Context based Visualization of Gene Expression Data 63
5.1 Meta Information 63
5.2 Profile Logo 65
  5.2.1 Discretization of numerical data 66
  5.2.2 Implementation in Mayday 67
  5.2.3 How to Interpret Profile Logos 68
5.3 Chromogram and Tag Cloud 71
  5.3.1 Chromograms 72
  5.3.2 The Chromogram Plugin for Mayday 73
  5.3.3 Tag Clouds 74
  5.3.4 The Tag Cloud Plugin for Mayday 76
  5.3.5 Interpretation of Chromograms and Tag Clouds 77
5.4 Term Pyramid 79

5.4.1 The Term Pyramid Plugin for Mayday 80
5.4.2 Interpretation of Term Pyramids 81
5.5 Probe Rank Plot 82
5.5.1 The Probe Rank Plot for Mayday 82
5.5.2 Interpretation of Probe Rank Plots 84
5.6 Summary and Discussion 86

6 Graph Based Visual Exploration of Gene Expression Data 89
6.1 Exploratory Analysis of Biological Networks 89
6.2 Related Work 92
6.3 The Mayday Graph Framework 95
6.3.1 Overall Design 95
6.3.2 Graph Visualization in Mayday 96
6.4 Graph-based visualization of expression data 98
6.4.1 Options for visualizing nodes 99
6.4.2 Adding Meta Information 102
6.4.3 Visualizing edges 104
6.4.4 Graph Layout 105
6.5 Pathway Viewers 106
6.5.1 The KEGG Pathway Viewer 108
6.5.2 The SBGN Pathway Viewer 111
6.6 MGV - The Mayday Graph Viewer 114
6.6.1 Implementation 114
6.6.2 Node Groups 116
6.6.3 Working with Graphs in MGV 118
6.6.4 Data Integration 120
6.7 Cross Dataset analysis 130
6.7.1 Cross-Dataset Visualization and Statistics 130
6.7.2 Example 132
6.8 Summary and Discussion 133

7 Discussion and Outlook 137
7.1 Discussion 137
7.2 Outlook 140

A MGV Key Shortcuts 143

B Publications 145
B.1 Articles 145
B.2 Posters & Talks 146

List of Figures

2.1 Processes involved in gene expression 6
2.2 Complexity of gene expression 7
2.3 Key regulation points in gene expression 8
2.4 Probe designs for gene expression and tiling microarrays 10
2.5 Several microarrrays of different platforms 11
2.6 Background and foreground intensities of a cDNA microarray 18
2.7 Effect of loess normalization 18

3.1 Anscombe's Quartet 24
3.2 Classical examples of data visualization 25
3.3 Parallel coordinate plot 32
3.4 Clustered heatmap 33
3.5 Box plot 34
3.6 Bar plot and histogram 35
3.7 Sequence Logo 36
3.8 Overview of the SBGN process description language 43
3.9 First steps of the glycolysis in the SBGN process description language . . 44
3.10 Main window of Mayday 46
3.11 Overview of the architecture of Mayday 47

4.1 Intensity charts of a resequencing microarray 50
4.2 Probes on a resequencing microarray 51
4.3 Overview of Clark's base calling algorithm 52
4.4 Intensity and sequence visualization in ResqMi 55
4.5 Example of the Sequence Heat Map 56
4.6 Resequencing window of ResqMi 57
4.7 Potential quality issue in the Mito dataset 59
4.8 No-call and non-reference call ratios for the CFTR and Mito dataset. . . . 60

5.1 Profile logo 68
5.2 Profile logo of antibiotics cluster 70
5.3 Profile logo of a mixture of clusters 71
5.4 Chromogram color transformation 72
5.5 Chromogram example 73
5.6 Tag cloud example 75
5.7 Tag cloud of annotations variant genes in *Steptomyces coelicolor* 76

5.8 Chromogram and tag clouds and of clustered genes 78
5.9 Interactions between tag clounds and chromograms 79
5.10 Stacked chromogram 80
5.11 Term pyramid 81
5.12 Term pyramids examples comparing clusters on Gene Ontology terms . . 83
5.13 Probe rank plot 84
5.14 Probe rank examples 85
5.15 UML diagram of the five plots 88

6.1 Visualization of biological pathways in R and Cytoscape 93
6.2 UML diagram of core graph classes of 97
6.3 Overview of the graph visualization classes in Mayday 98
6.4 Data preparation for node rendering 99
6.5 Basic node renderers 100
6.6 Single probe renderers 101
6.7 Multiple probe renderers 101
6.8 Renderers for additional information 102
6.9 Renderer decorator concept 103
6.10 Meta information decorators 104
6.11 Auxiliary items 104
6.12 Edge rendering options 105
6.13 Group based layout of the neural network of *C. elegans* 107
6.14 Group based layout of transcription factors and controlled genes in *Saccharomyces cerevisiae* 107
6.15 Overview of a KEGG pathway 110
6.16 Detail of a KEGG pathway 111
6.17 SBGN pathway viewer showing the phenylalanine biosynthesis in *P. aeruginosa*. 112
6.18 SBGN pathway viewer showing the histidine degradation in *P. aeruginosa*.113
6.19 Scope of MGV 114
6.20 Grouped nodes in MGV 117
6.21 Abstracts and interaction partners of SCO5088 119
6.22 Methods of joining nodes in MGV 120
6.23 Network of biochemical pathways in *S. coelicolor* 123
6.24 Examples for data integration in MGV 124
6.25 Comparison of clusterings of genes between conditions in *Saccharomyces cerevisiae*. 126
6.26 Differential splicing and gene expression 129
6.27 Annotation graph for SCO1942 131
6.28 Cross dataset statistic options 132
6.29 Cross dataset visualization 134

Zusammenfassung

Die heutigen Methoden der Genexpressionsanalyse erlauben die Datenerfassung mit zunehmender Geschwindigkeit und Qualität. Hochdurchsatzverfahren wie DNA-Microarrays und Sequenzierungsverfahren der zweiten Generation haben zahlreiche neue Entdeckungen ermöglicht. Zusammen mit Ergebnissen aus Verfahren der Proteom- und Metabolomanalyse stehen große Datenmengen zur Verfügung, zusätzlich ergänzt durch viele Annotationen und Metadaten. Im Rahmen von Genexpressionsstudien müssen diese Daten oft mit visuellen Methoden untersucht und analysiert werden. In dieser Arbeit werden dazu neue Methoden und Konzepte für die Visualisierung von Genexpressionsdaten im Kontext von Metainformationen und Ergebnissen anderer Technologien vorgestellt.

Zunächst werden Standardvisualisierungsmethoden für Resequenzierungs-Microarrays diskutiert. Zum Zwecke der Anwendung von generischen und angepassten Visualisierungsmethoden auf entsprechende Daten wird das Programm ResqMi ("Resequencing using Microarrays") vorgestellt. ResqMi bietet neue Möglichkeiten für die Qualitätskontrolle, Analyse und Nachbearbeitung der Daten.

Der Fokus dieser Arbeit liegt auf der Visualisierung von Genexpressionsdaten. Zunächst werden einige Visualisierungsmethoden für Genexpressionsdaten im Kontext von Metainformationen aus Prozessierungsergebnissen und externen Quellen - etwa Funktionsannotationen - vorgestellt. Zur Darstellung von geclusterten Genexpressionsprofilen werden Profillogos verwendet, die das Konzept der Sequenzlogos für die Darstellung von Expressionsdaten erweitern. Chromogramme und Tag Clouds, Visualisierungstools für verschiedene Aspekte von nominalen Daten, werden hier kombiniert genutzt, um Muster in Annotationen von Genexpressionsdaten zu finden. Außerdem werden zur Visualisierung von nominalen und ordinalen Metainformationen geeignete erweiterte tabellenartige Ansichten verwendet: die Term Pyramid bzw. Probe Rank Plots.

Die graphenbasierte Visualisierung von Genexpressionsdaten ist generischer und bietet viele zusätzliche Möglichkeiten und wird im Weiteren näher verfolgt. Viele Anwendungen für die Visualisierung biologischer Pathways nutzen nicht alle Möglichkeiten für die Darstellung von Genexpressions- und Metadaten. Hier werden verschiedene Möglichkeiten zur Einbeziehung von Genexpressionsdaten in die Darstellung biologischer Daten untersucht. Sowohl spezielle Anwendungen für die Darstellung biologischer Pathways (in KEGG- und BioPax-Format) als auch MGV ("Mayday Graph Viewer") werden vorgestellt. MGV ist ein generisches Tool, das die Visualisierung vieler verschiedener biologischer Netzwerke mit vielen Optionen innerhalb einer mächtigen Oberfläche ermöglicht. Verschiedene Strategien für die Erzeugung, Strukturierung und Analyse innerhalb dieses Programmes werden vorgestellt. Außerdem wird die Integration von Daten aus unterschiedlichen Studien und Technologien in MGV untersucht. Dynamische Gruppen von Knoten, die mit Daten aus verschiedenen Quellen angereichert sind, sind die Ausgangsbasis für datensatzübergreifende Analysen. Weitere Anwendungen umfassen unter anderem die Analyse von Metabolomik-Daten, der Vergleich von Clusterings und die Visualisierung von Genmodellen.

Abstract

Today, gene expression data is acquired with increasing speed with increasing quality and depth. High throughput technologies like DNA microarrays and next generation sequencing technologies have led to a rising pace of new discoveries in the biomedical field. These technologies are complemented by high throughput pipelines for proteomics and metabolomics profiling. Altogether, vast amounts of primary measured data, complementary data from other omics and meta information from many sources is available for researchers. This data needs to be jointly analyzed and visualized in context of external data and meta information. In this thesis, new tools and concepts are introduced for the purpose of visualizing gene expression data in the context of meta information and complementary data from other "omics" experiments.

First, the application of generic visualization tools to resequencing microarrays, which are used for finding mutations in single genes is discussed. For this final step of gene expression analysis, an application called ResqMi, ("Resequencing using Microarrays") is presented that allows to use generic and adapted visualization tools on resequencing microarrays, in order to improve quality control, data analysis and revision of problematic base calls.

The focus of this work is on the visualization of gene expression data. Here, new tools for the visualization of gene expression data in the context of meta information from processing results and external sources, like functional annotations are introduced. For the visualization of clustered gene expression data, profile logos extend the concept of sequence logos to expression data. Chromograms and tag clouds, tools for visualizing different properties of collections of nominal data are applied in combination in order to explore temporal, spatial and other patters in annotations of gene expression data. Furthermore, enhanced tabular views of summarized gene annotations and genes ranked by statistical values are discussed for comparative visualization of textual and numeric meta data.

Graph based visualizations of gene expression and meta data are more generic and investigated in greater detail. Most tools for visualizing biological pathways do not make full use of gene expression or meta information data. Here, a variety of ways to include gene expression data into biological network visualizations is investigated and implemented, based both on the node rendering and the layout of the graph. This allows dense, high dimensional visualizations. Specialized tools that are optimized for working with pathways in KEGG and BioPax formats, are presented as well as MGV (the Mayday Graph Viewer), a general tool for visualizing a wide range of biological networks that offers a full range of options within a rich, extensible user interface. Options for integration and creation of network data, data organization and analysis within the graph framework are investigated. MGV furthermore incorporates tools for integrating data from several datasets, which allows to combine multiple “omics” data in one visualization. With dynamic groups that can contain nodes with data from all sources, cross dataset analyses can be performed. Further applications include the integration of metabolomics data, clustering comparisons and the visualization of gene models.

Acknowledgments

I owe my deepest gratitude to Dr. Kay Nieselt, for providing me the opportunity to work on this interesting topic and for comprehensive support, much freedom to complete this thesis and interesting discussions. I am grateful towards Prof. Dr. Michael Kaufmann for supervising my thesis and providing important support at the end of my work. Furthermore, I would like to thank Prof. Dr. Oliver Kohlbacher for making available his support in a number of ways.

I am very grateful towards Florian Battke, his formidable technical and theoretical knowledge was a great help during fruitful discussions and collaborative efforts on several projects.

I am thankful towards my colleagues at the Center for Bioinformatics, especially towards Janko Dietzsch, Alexander Herbig, Aydın Can Polatkan and Günter Jäger, who created a great creative and productive working atmosphere in our group and with whom I had many productive (and funny) discussions. Florian and Alexander are furthermore to be acknowledged for proofreading my manuscript, which was a great help for me.

For the successful work with the Mayday team over the years, a number of people are to be especially acknowledged: Nils Gehlenborg for initiating the project and Janko Dietzsch for mentoring it. I also want to thank several students who worked on this and other projects and made Claudia Broelemann, Katharina Buck, Stephan Gade, Roland Keller, Bettina Knapp, Matthias Munz, Michael Piechotta, Stefan Raue, Christian Sieber, Lucia Spangenberg, Nastasja Trunk Karin Zimmermann, and Christian Zipplies. They and many others helped to made the Integrative Transcriptomics Group a wonderful place to work.

I am grateful towards Kirstin Weber and Michael Bonin for the successful collaboration on ResqMi, as well as to Gunnar Rätsch for advice on resequencing microarrays.

It was a honor to work with many collaborators on several interesting and successful projects. These are especially Prof. Dr. Christina Pfannenberg, Prof. Dr. Hinnak Northoff, Markus Löffler, Derek Zieker and Ingmar Königsrainer from the Universitätsklinik Tübingen and Prof. Dr. Peter Gebicke-Haerter, Ulrich Sommer and Markus Heck from the ZI, Mannheim. During these projects I have learned a lot.

Whenever I felt I was “doomed”, my friends with much optimism and a dash of sarcasm were a great support for me, letting me soon find out that in most cases, I was not doomed. This inspiration and support during my studies was very important to me. I want to express my gratitude towards Lisa Schuster, her support and patience during my work on this thesis helped me a lot, as well as I am especially grateful to my parents and family for their wonderful support during my studies and PhD work and my whole life.

List of Abbreviations

BFS	Breadth First Search
DNA	Deoxyribonucleic acid
DFS	Depth First Search
EDA	Exploratory Data Analysis
FDR	False Discovery Rate
FWER	Family-Wise Error Rate
GO	Gene Ontology
HSB	Hue Saturation Brightness color model.
HTML	Hypertext Markup Language
KEGG	Kyoto Encyclopedia of Genes and Genomes
KGML	KEGG Markup Language
MAGE-ML	Microarray Gene Expression Markup Language
MIAME	Minimum Information About a Microarray Experiment
MIO	Meta Information Object
PCA	Principal Component Analysis
OWL	Web Ontology Language
ResqMi	Resequencing Using Microarrays
RNA	Ribonucleic acid
SAX	Symbolic Aggregate ApproXimation
SBH	Sequencing by Hybridization
SBGN	Systems Biology Graphical Notation
SBML	Systems Biology Markup Language
SBO	Systems Biology Ontology
SI	International System of Units
SNP	Single Nucleotide Polymorphism
SOM	Self Organizing Map
SVG	Scalable Vector Graphics
TSD	Transitional State Discrimination
UML	Unified Modeling Language
XML	Extensible Markup Language

1. Introduction

This thesis is concerned with the analysis and visualization of gene expression data. The analysis of gene expression is motivated by the numerous findings expected from comprehensive knowledge of the pattens of gene expression in an organism. Gene expression is involved in nearly all biological processes. Patterns of gene expression control cell cycle, embryonic development, aging, diseases including cancer, or adaptation to environmental changes. There are numerous regulation methods on multiple levels, that control gene expression [111]. These mechanisms make the analysis of gene expression data a complicated endeavor, even for single genes. However, the focus of biomedical research recently shifted from the reductionist view of single aspects to a holistic view of entire systems. Ultimately, systems biology intends to comprehensively model entire biological systems with formal methods. To this end, high quality data measuring the activity and abundance of ideally all relevant components is necessary. For the analysis of gene expression, high-throughput methods are available on all levels, that allow the quantitative analysis of thousands of components in parallel. These systems encompass microarrays and RNA-seq for transcriptomics and mass spectrometry methods, combined with gas or liquid chromatography for proteomics and metabolomics.

Comprehensive gene expression analysis gives rise to large, multilevel datasets, where each level introduces new potential insights, but also new potential problems. Besides statistical modeling and analysis, an important tool gene for expression data analysis is visualization. It is common to visualize the data at multiple points during the analysis process. Usually, this is done before normalization steps to decide on methods and parameters and after data processing steps to assess the result. Furthermore visualization is used before and after confirmatory analysis to generate and illustrate hypotheses. There are no defined steps for this process, and a large variety of methods is used [47].

Visualization is the process of creating graphical representations for data that facilitate interpretation and identification of patterns using human perception [27]. Visualization has been established as an important analysis tool, especially for exploratory purposes. The importance of visualization is classically shown by Anscombe's quartet [5], that highlights how important it is to visualize the primary data. Common tools encompass parallel coordinates and heat maps, which allow large scale visualization of high dimensional data. Furthermore, good visualization require careful data processing and transformation [58, 205]. Modern analysis tools allow to create multiple connected and coordinated views of the data, that even in a static form would have been impossible to create by hand. Challenges encompass the structuring and preparation of the data. The concept of visual analytics introduces summaries and overviews, combined with statistical analysis and optimized perception, also in dealing with large and complex datasets.

An important aspect of modern gene expression data analysis is that there is already a vast knowledge about each of the measured components. Annotations, sequences, structures, scientific articles and functional categories as well as activity measurements in other conditions are available for many genes, proteins and metabolites. This information is important to use during analysis and visualization.

Integrating meta information into visualizations of gene expression data is therefore a main topic in this thesis. For this purpose, first a set of context-based visualization tools for gene expression data is presented. Partially these tools are based on well-established visualizations applied in other areas, like sequence logos [159] for multiple sequence alignments and tag clouds [198] for word counts. The context-based visualization tools allow to interactively visualize textual and numerical meta information combined with gene expression data, either directly or via connected views. The tools presented are of exploratory nature and are intended to find patterns both in gene expression data and annotations.

Often, meta information is structured and describes not a single object, but a biological process. The common representation of a biological process is a so-called pathway. Pathways are conceptual representations of biological processes and can be seen of the modules of which life consists. It is important to keep in mind that biological pathways are not isolated processes, but instead are highly connected and interfere with each other. Biology is therefore also referred to as a network science [10]. Networks of various types and densities can be found in biology. Besides pathways and their interactions, molecular level interactions and regulatory networks are investigated.

The natural representation for a biological network is a graph. Based on this powerful theoretical foundation, visualization and analysis of biological pathways is done. While this is already an important part of high level meta information, in this thesis a thorough visualization of graph-based meta information and textual as well as numerical meta information along with gene expression data is presented. For this purpose, the various options for visualizing gene expression data and meta information in graphs are investigated. This is also part of an exploratory strategy [97]. Implementations of this concept are presented, in which users can dynamically modify generic or specialized graphs enriched with gene expression and meta information data. Visually analyzing the graphs is possible using various tools for summary, details and dynamic extensions and reductions of the graphs. Due to the fast development in the biomedical field, extensibility is important. In order to prove this extensibility, the analysis and visualization of several problems in this field are investigated, as well as the problem of handling multiple datasets. Handling of multiple datasets that describe the complementary data, is often cumbersome, even for basic visual comparison and analyses. The framework presented herein partly addresses this problem.

The analysis of gene expression cannot fully answer the question of whether and how a gene is active. Multiple mechanisms for modifications before and after transcription might distort the gene product. Additionally, due to mutations, the protein resulting from gene expression might not be active at all. Causes are often single nucleotide polymorphisms (SNPs), i.e. single (non-synonymous) mutations in the genomic sequence of the gene. These mutations are best observable on the genomic level. Therefore, genomic analysis can be seen the final step of gene expression analysis. A proven tool for identifying mutations are resequencing microarrays [69]. Resequencing microarrays are based on the same principle as gene expression microarrays, however they are used with genomic DNA and yield not expression values, but the sequence of one or a few single genes. Base calling algorithms [41] transform the measured intensities into base sequences. For the analysis of resequencing microarray data, the tool ResqMi is presented in this thesis. It addresses the necessity of quality control and visualization features, as well as of fast and interpretable base calling methods. Quality control and visualization methods designed for resequencing arrays might also prove to be useful for new and emerging resequencing technologies.

This thesis contributes new methods for visual analysis and exploration of a variety of data, with the emphasis on high-throughput life science data. Based on the current state of methods for gene expression data analysis and visualization, an inventory of possible methods for visualizing annotations in context is provided. The context-based tools presented here are adjusted and applied to gene expression data. Graph-based visualization of gene expression data, in the context of biological models is a topic of active research. The tools provided in this work extend current approaches by added visualization plots and the option to include meta information and annotations in a modular way. Implementations of all these concepts are available. Facile integration of complementary expression data further adds to the use of the presented tools. Complement to this, the analysis of resequencing microarrays is enhanced by ResqMi with visualization tools for quality control, data inspection and summarization.

1.1. Outline

This thesis has three major parts: (1) The biological and theoretical background for this work is given in chapters 2 and 3. More precisely, chapter 2 gives the biological background on gene expression, the measurement thereof, especially on the transcriptional level. Further techniques of acquiring systems biology data are briefly discussed. The analysis of gene expression data is also summarized in this chapter. The following chapter 3 summarizes the development and current state of data visualization, with a focus on visualization of gene expression data. Elements of graph theory and graph drawing are then discussed. Special attention is given to the visualization of biological pathways. The chapter is concluded by a brief overview of the analysis tools used and extended in this thesis.

(2) The main work presented in this thesis is subject of chapters 4, 5 and 6. The first chapter deals with the analysis and visualization of resequencing microarray data. Proposals for the further use of the techniques are given at the end of this chapter. Chapter 5 gives an overview of visualizing gene expression data in the context of meta information. In this chapter, several context-based visualization tools are introduced and applied. Graph-based visualization of biological data is the topic of chapter 6. This chapter describes the details of graph-based visual data exploration and presents imple-

mentations of this concept: a specialized variant for biochemical pathways, and a generic graph viewer for biological data, both in concept and implementation.

(3) This thesis is completed in chapter 7 with a discussion of the work presented and an outlook on possible further directions.

2. Analysis of Gene Expression

Understanding the patterns of gene expression is one of the major topics in biological and medical research. This is in fact one of the core questions of biology: How do organisms function at the most basic level of control? Regulation of gene expression concerns many parts of biology and medicine: developmental genetics, physiology, signal transduction, infection biology, oncology and many others [111]. While the process of gene expression in general is fairly well understood, the regulation of gene expression is not trivial even in simple organisms, such as viruses and bacteriophages. In this chapter, an overview of the process of gene expression, as well as of the most important regulation mechanisms is given (section 2.1). There is a large variety of tools for the analysis of gene expression. This thesis focuses on high-throughput techniques, such as microarrays and RNA sequencing (RNA-Seq). They are briefly introduced in sections 2.2 and 2.3. These techniques are used to study gene expression at transcription level, which is the first step of the process. Regulation and variation of gene products are however not limited to transcription. Therefore measuring the protein concentration, as the result of the translation step is also important. Furthermore, the final effects of gene expression is measurable on the metabolic level. Techniques for the analysis of metabolomes are discussed in section 2.4.

2.1. Analysis of Gene Expression

The analysis of gene expression is crucial for current biological research. Gene expression is controlling a vast number of biological processes, including cell differentiation, stress response, embryonal development or cancer. Cells of the same organism, which have the same genome, can differ greatly in function. These differences are due to differential gene expression. Important insights about these and other processes are to be gained by analysing the modal, temporal and spacial patterns of gene expression.

2.1.1. Gene Expression

Gene expression can be considered the fundamental activity of all organisms. Most activity of an organism, including gene expression itself, is controlled and performed by proteins, which are (together with noncoding RNAs) the products of gene expression. The basic principle of gene expression, on a high-level view, consists of two steps: *transcription* and *translation* (see figure 2.1). Origin of all gene expression is the genomic DNA of a cell, that contains genes, i.e. transcribed loci. During transcription, the sequence of a gene is transcribed into a complement RNA molecule. For protein coding genes, the RNA is called a messenger RNA (mRNA), and it serves as a template for protein biosynthesis

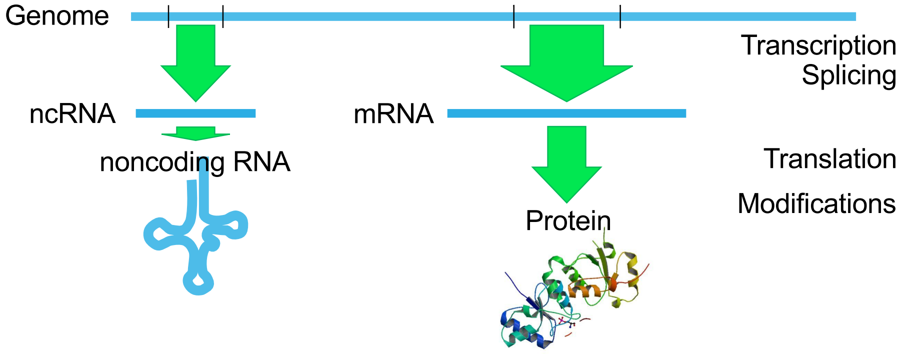
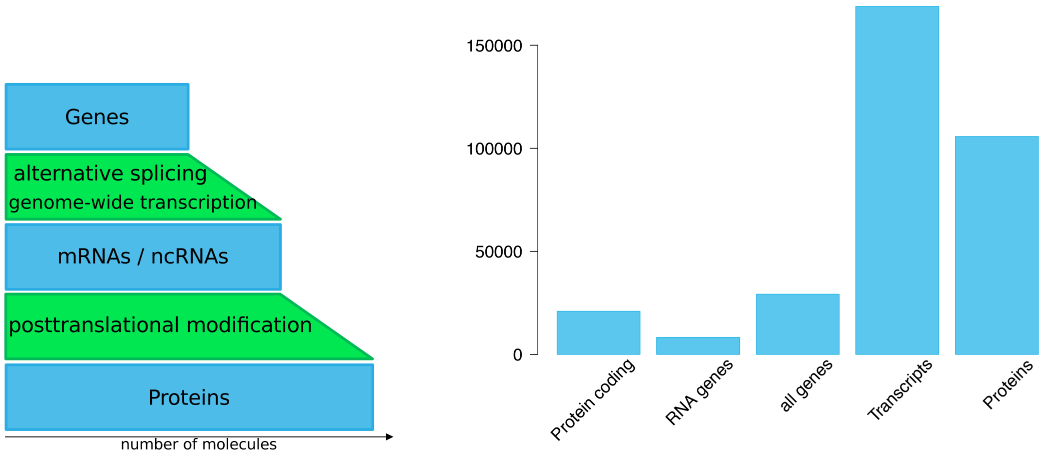


Figure 2.1.: Coarse overview of the processes involved in gene expression. The figure omits transport processes necessary in eukaryotes and genomewide transcription processes for brevity. Protein structure taken from PDB, accession id: `3pxe`

during the translation step. Classically, the information flow from DNA to RNA to proteins is called the *central dogma* of molecular biology. Today, several exceptions and extensions of this concept are known, e.g. retroviruses, which have RNA as genetic material. The *transcriptome*, the assembly of all transcripts in an organism, is by no means limited to mRNAs. Non-coding RNAs, including ribosomal RNAs and transfer RNAs (tRNAs) and several different types of regulatory RNAs are a sizeable part of the transcriptome. The qualitative and quantitative analysis of the transcriptome is called *transcriptomics* and has lead to important insights into many biological questions

The biological details of gene expression are non-trivial and are not discussed here. However, it is important to note that prokaryotes and eukaryotes differ greatly in genome organization, which gives rise to important differences in the process and complexity of gene expression. Prokaryotes produce so-called polycistronic mRNAs that span several genes. If genes structured in this way are commonly regulated, they form a so-called *operon*. Operons often contain genes for a common biological process. The *lac* operon [84], which contains genes for lactose metabolism in *Escherichia coli* is a classical example.

Eukaryotes have a nucleus, a separate compartment in which the DNA is located, and in which transcription takes place. Furthermore, especially in higher eukaryotes, genes consist of *exons* and *introns*, of which only the former are translated. The process of *splicing*, which removes the introns, may give rise to different transcripts from the same genomic locus, introducing a further layer of complexity into the eukaryotic transcriptome. Both retention of introns and skipping of exons are common events of differential splicing. The patterns of differential splicing are an important subject of research. In the latest release of the *Homo sapiens* genome, 20,930 protein coding genes and 8,271 RNA genes are annotated [55, 53]. These give rise to 168,848 transcripts, which consist of 615,132 exons. Furthermore, transcripts may occur from not annotated regions. This *genome wide transcription* [92] further increases the complexity of the transcriptome.



(a) Potential of gene expression

(b) Annotated objects in *Homo sapiens*

Figure 2.2.: Complexity of gene expression. Part (a) shows the potential of gene expression, induced by the possible RNA and Protein processing steps. Part (b) shows the currently annotated numbers of genes (from Ensembl [53]) and proteins (from UniProt [194]) in *Homo sapiens* as of April 2011.

Additional complexity can be observed on the level of the proteome (the whole ensemble of proteins in an organism). *Posttranslational modifications* are observed in all organisms and alter proteins to differ from their primary sequence defined by the mRNA. Common modifications are acetylations, methylatons or phosphorylations, and a large variety of other modifications has been identified [126]. Potentially, the number of distinct proteins in an organism far extends the number of genes and transcripts (see figure 2.2a). Currently, fewer proteins than transcripts are annotated (based on UniProt [184, 194], see figure 2.2b). This is due to the facts that not every transcript is translated and that protein identification is non-trivial.

In summary, studying gene expression is an endeavor to be taken on multiple levels. From a limited number of genes a much larger number of different transcripts and proteins may be produced. The main complexity of higher eukaryotes does not lie, as once thought, in the number of genes, but in the variations and regulation of gene expression.

2.1.2. Regulation of Gene Expression

The *regulation* of gene expression is important for cells and organisms. Unnecessary gene products are a waste of resources, but may also harm the cell structure or even the entire organism. Processes as apoptosis or cell division must be controlled carefully for normal development and to prevent cancer. This is achieved by a multitude of regulatory processes, of which the most important ones are shown in figure 2.3.

On the level of transcription, gene expression is controlled on DNA level. The process of transcription is performed by the RNA polymerase that must bind to a transcription

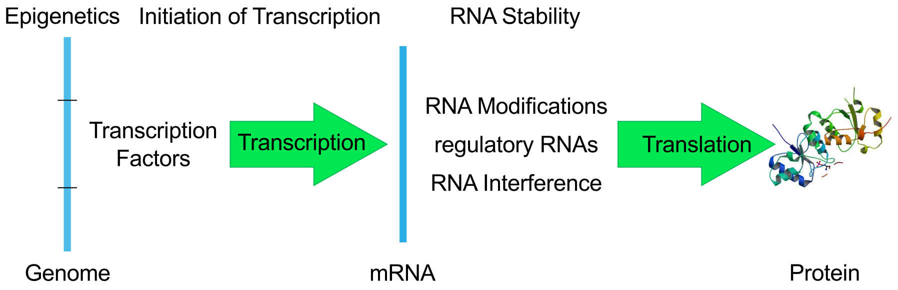


Figure 2.3.: This figure shows the key regulation points in gene expression: Epigenetics which controls the overall access to the DNA, initiation of transcription, which is controlled by transcription factors and RNA stability. Protein structure taken from PDB, accession id: `3pxe`

start site on the genomic DNA. The binding of the RNA polymerase and the efficiency of transcription is controlled by various transcription factors. The access for the RNA polymerase can be enhanced or restricted by enhancer or repressor proteins, that bind to specific regions, often, but not necessarily near the transcription start side of a gene [111].

Due to genomic *imprinting*, a set of epigenetic modifications, the access to the gene locus may not be possible at all. DNA methylation or the modifications to the histone molecules that structure the genomic DNA prevent the unfolding of the DNA. In consequence, the RNA polymerase and transcription factors cannot bind to the DNA and no transcription can occur.

After transcription, the stability and processing of pre-mRNA are targets of regulation. The stability of RNA is enhanced by the 5'-cap-structure and the polyadenylation of the 3' end of the RNA, which is subject to rapid decay by various enzymes without these structures. The mRNA needs to be single stranded in order to be used in translation. Antisense RNAs, such as miRNAs and siRNAs specifically target mRNAs which are then inactivated and decayed rapidly. This process is called RNA interference (RNAi) and leads to the silencing of expressed genes.

The multitude of regulation mechanisms make the analysis of gene expression difficult, since the effects of changes (i.e. resulting from mutations, the environment, and other causes) are not necessarily linear: The control of transcription in an organism is exercised by a complex and non-linear network that can contain thousands of components.

2.1.3. Measuring Gene Expression

The question of whether and in which quantity a gene is expressed is common in biological research. For this purpose, many different methods have been developed to measure gene expression on all levels. On the transcription level, the earliest method for qualitative detection of mRNAs is the northern blot [4]. A quantitative measurement can be achieved with quantitative real time PCR (qRT-PCR). *In-situ* hybridization allows to find spatial patterns in transcript abundance. While these methods focus on a single, or a few genes, there are also high-throughput methods, that allow to measure whole genome transcription [111]. *SAGE*, serial analysis of gene expression [197] works by identifying short sequence tags. *DNA microarrays* (see section 2.2) use nucleic acid hybridization to estimate transcription levels. Recently, *high throughput sequencing* of RNA (RNA-Seq) allows a more direct measurement of transcripts. This technique is beginning to replace microarrays for most applications [163]. The latter two methods are described in detail in the next two sections.

On the translation level, the presence of proteins can be detected via *western blot*, and, for a spatial pattern, using *immunoflourescence* methods [111]. Towards a whole proteome analysis, *2D polyacrylamide gel electrophoresis* (2D-PAGE) allows the separation of several thousand proteins, and could also be used in comparative studies. *Liquid chromatography followed by mass spectrometry* (LC-MS) allows to quantify, via peptides, a large number of proteins from complex samples.

In addition to measuring the intermediate and final products of gene expression, it is often useful to quantify the effects of gene activity, especially in the case of enzymes. Metabolite concentrations can be measured by various means of chemical analytics. In order to quantify a high number of metabolites, *gas chromatography* followed by mass spectrometry (GC-MS) is used (see section 2.4).

2.2. The Microarray Technology

2.2.1. Principle

The until now most frequently used tool for the whole-genome analysis of gene expression are *DNA microarrays*, or microarrays for short. The underlying idea of microarray-based gene expression analysis is the assumption, that the concentration of mRNA is proportional to the activity of the related gene. The general principle of measuring RNA concentrations using microarrays is nucleic acid *hybridization*: A labeled sample of DNA or RNA, extracted from a biological source, e.g. a certain tissue or cell culture, hybridizes to a DNA probe. The probe is fixed at a defined position (called spot) within a grid on a solid surface, called the (micro)array or slide. Hybridization fixates the labeled sample to the probe. The label, usually a fluorescent dye, allows then to estimate the concentration of the different DNA features in the sample. Fluorescence intensity is assumed to be proportional to the concentration of the labeled sample. This leads to a relative, dimensionless measurement of gene expression, which usually requires quality control and normalization in order to be interpretable. It is possible to use either one or two samples, using two different fluorescent dyes. Thus, on one array, two hybridizations are possible, allowing two measurements to be made. For this purpose, the dyes *Cy5* (red under fluorescent light) and *Cy3* (green under fluorescent light) are commonly used. More than two samples on the same array would be possible in theory, however, no such established platforms exist.

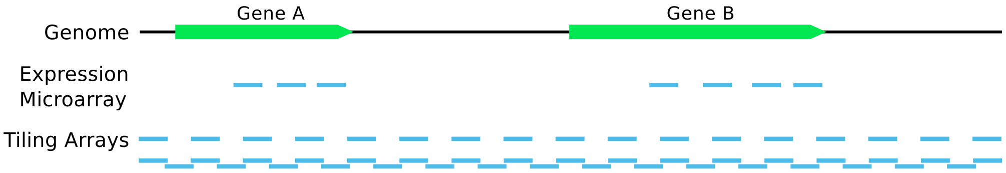


Figure 2.4.: Possible outcome of a probe design for a gene expression microarray and for two tiling arrays with different resolution. Genes A and B are denoted in 5'-3' direction. Probes in the respective array designs are displayed as blue lines.

2.2.2. Technologies

Several technologies of microarrays are available (see figure 2.5 for examples). In general, the production method of the physical arrays and the origin of the DNA probe characterize a microarray platform. Two strategies of producing microarrays are used:

- *Spotted* microarrays are produced using a spotting robot that prints the probe material onto the substrate. The probes can either be cDNA or oligonucleotides. In the former case, probes can be up to several hundred bases long, in the latter up to 60 bases is common. The main limitation of the spotting approach is a potential lack of quality: uneven spot sizes, shapes and non-uniform grids are common. However, costs and the effort of building a new design are low. Spotted microarrays are frequently used for two-channel experiments.

- *In-situ* microarrays are produced by in place synthesis of the probe [170]. This is done in an iterative process: In each cycle, a protective group is removed from the open end of the oligonucleotide using ultraviolet light which is directed to the relevant positions, often using a mask. Then, the next nucleotide is concatenated to the growing oligonucleotide. Each step requires a different mask, which is expensive and limits the probe length; to address this, maskless production processes are available (e.g. from Agilent Technologies). This process however leads to high quality arras with uniform grids and evenly sized square spots. Affymetrix GeneChip®s are produced in this way. Like the Affymetrix arrays, most *in-situ* arrays are used for single channel studies.

As *probe targets*, in general every potentially expressed gene (including RNA genes) could be used. One or more probes can be used for one gene, often to query the 3' end of the gene. Careful selection of the probe sequence is necessary to obtain sensitive and specific measurements: secondary structures and non-unique probes as well as probes that would allow cross-hybridization are to be avoided. Furthermore, the DNA melting temperature of all probes must be similar. On Affymetrix arrays, also mismatch probes exist, that differ in their central position from the so-called perfect match probe by using the Watson-Crick complement of the central base of the perfect match probe. This setup is used for estimating the amount of unspecific hybridization. For so-called *tiling arrays*, fragments of genomic DNA, chosen by a regular pattern are used instead (see figure 2.4).

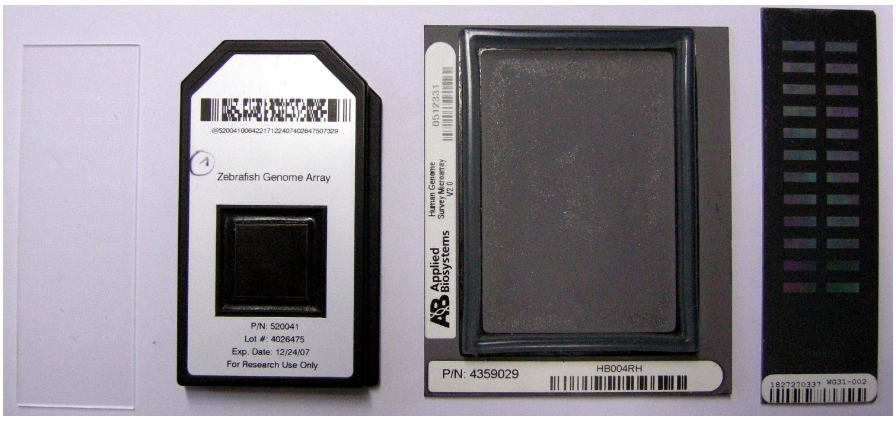


Figure 2.5.: Microarrays of different platforms. From left to right: custom made cDNA microarray printed on a glass slide; Affymetrix GeneChip for *Danio rerio*; Applied Biosystems whole genome microarray for *Homo sapiens*; Illumina BeadArray for *Homo sapiens.*

Using known genes is most useful for gene expression analysis. Tiling arrays, while not optimized for this purpose allow to detect new transcripts and can be used for DNA analysis studies (see below).

2.2.3. Development and Standards

The microarray technology has been established during the 1980s and 1990s, starting with rather small designs querying a few dozen of genes in parallel. With great efforts in miniaturization and genome projects being completed at an increasing rate, whole genome arrays became feasible towards the end of the 1990s. Today, numerous whole genome arrays for human and many other organisms are available from a variety of suppliers (Affymetrix, Agilent, Applied Biosystems, Illumina, etc, see figure 2.5 for examples).

Considerable theoretical work on the normalization and data analysis (see 2.5 for a short summary) and key studies performed by Eisen *et al.* [52], Spellman *et al.* [169] and Golub *et al.* [66], established microarrays as a significant research tool. Standardization and reproducibility became increasingly important, which lead to the establishment of the MIAME (Minimum Information About a Microarray Experiment) standard [21], which dictates the minimum amount of information researchers must disclose to make their data reproducible. The establishment of public databases, mainly GEO [50] and Array Express [22] allow researchers to share microarray data structured in this way. Quality Control efforts, like the Microarray Quality Control project (MAQC) [164] and data integration between microarrays and other experimental platforms have lately received much attention.

2.2.4. Experimental Workflow

A typical microarray experiment consists of several steps:

1. *Design of study.* At this point, the microarray platform is chosen. Based on this choice, the experiment design (see [208] for an overview), is devised, taking into account the number and nature of the samples and the number of replicates. For two-channel platforms, the experiment design requires careful consideration: Indirect designs are more extensible and fault-tolerant, but require more arrays. Direct designs allow direct comparisons, and require potentially fewer arrays, but the loss of one array can destroy the entire study.
2. *Preparation of arrays and samples.* This includes the production of the microarrays and the extraction and labelling of the biological samples.
3. *Hybridization* of arrays with samples. In this crucial step, the binding of sample to probe takes place. Afterwards, washing steps are required to remove non-bound sample material.
4. *Scanning and data acquisition.* This step consists of detecting the spots on the array, discriminating them from the slide background (segmentation) and measuring intensity values for each spot. This concludes the laboratory work. Subsequent steps are the normalization and statistical analysis of the resulting data (see section 2.5).

In each step, sources of variation are introduced, which have to be removed during normalization. Therefore, to ensure that the experiment is reproducible, each of these steps needs to be documented according to the MIAME [21] standard.

2.2.5. Microarrays for Genomic Analysis

The hybridization-on-chip principle has been applied to more than gene expression analysis. Genomic analysis, especially for the purpose of sequence and variation determination, as well as the analysis of protein binding and epigenetic studies have been pursued using this technology. *Resequencing microarrays* [69] are a special case of the sequencing by hybridization (SBH) concept [9]. Originally, in SBH, determining a DNA sequence of length n would require 4^n probes. In resequencing, deviations from a known reference sequence are determined. Since the reference sequence is known, a single substitution can be detected using 4 probes, one for each possible base at the interrogated position. Therefore, when limited to substitutions, $4n$ probes are sufficient to determine an individual gene sequence. However, insertions and deletions cannot directly be detected in this way. For this purpose, specific strategies are discussed in [69]. Resequencing microarrays are a useful tool in clinical diagnostics, especially for detecting genetic diseases. Research applications include studying genomic variations of bacteria [2, 42], viruses [176] and other short genomes, e.g. the human mitochondrion [124]. A similar principle is commonly used for the determination of single nucleotide polymorphisms (SNPs). Here, the microarrays are designed to query known variant positions of a genome.

Other genomic studies performed with microarrays aim at detecting protein-DNA interactions. Chemo-immunoprecipitation on microarrays, ChIP-Chip for short, allows to identify protein binding sites in the genome [78]. For this purpose, usually tiling arrays are used. Epigenetic modifications of the DNA can be analyzed in similar protocols.

2.3. RNA-Seq

Recently developed technologies for high throughput sequencing allow to get a quantitative measurement of gene expression. The current technologies allow fast sequencing with deep sequence coverage, which is suitable for both genome sequencing and gene expression analysis. This technology involves the sequencing of libraries of RNA molecules and is therefore called *RNA-Seq* for short. Several different technologies for RNA-Seq exist. While the Sanger sequencing method allows RNA-Seq in principle, the new technoligies made it practical and affordable.

The advantage compared to the classical Sanger sequencing is that a large number of reactions are performed in parallel. This allows to produce millions of relative short reads (30-400 bp depending on the technology [200]). RNA-Seq protocols based on such technologies allow the direct quantification of RNA populations. In contrast to expression and tiling microarrays, RNA-Seq can identify the transcripts on a single base pair resolution. Furthermore, the background noise is considered lower, and the dynamic range is higher for RNA-Seq than for microarray-based technologies [200]. A reference genome is not necessarily required for all applications, for example the unsupervised profiling of transcripts is possible. Novel transcripts and new variants of known transcripts can also be detected.

2.3.1. Next-Generation Sequencing

Several different technologies for high-throughput sequencing are available. The earliest technology, from Roche/454 employs *pyrosequencing* [152] in microtiter vessels to generate reads of length of up to 400bp. Major applications of this technology are genomic sequencing: In a 2005 landmark study, Margulies *et al.* presented the sequencing of the complete genome of the bacterium *Mycoplasma genitalium*, with 238,000 reads of average length of 110 bp in a four hours effort [127].

The Illumina GenomeAnalyzer technology [15] works in a flow cell, on which DNA molecules with specific adapters are immobilized. Then, bridge amplification is used to create identical copies of each immobilized fragment. Resulting clusters of DNA have a high density and are the templates that are subject to sequencing. A *sequence-by-synthesis* approach is used, in which in each step, four different fluorescence-labeled reversible terminators (one for each base) are added to the flow cell. DNA polymerase appends the terminators at the end of the growing sequence, which is the complement of the template. Fluorescence scanning is used to determine the type of the last incorporated base, which is recognized by the specific fluorescence color of the terminator. The terminator is removed by the laser excitation during the fluorescence scanning, allowing to begin the next iteration. In this way, sequence reads with length of up to 100bp can be acquired [82].

Other technologies in this area are the SOLiD technology which is based on litigation-based sequencing and the Helicos technology for single molecule sequencing. Applications of next-generation sequencing include, besides genome sequencing and expression analysis, protocols for the detection of polymorphisms, protein binding sites (Chip-Seq) and epigenetic modifications.

RNA-Seq experiments require the creation of a library of RNA which is extracted from a biological sample. Fractioning of the RNA molecules allow to restrict such libraries to poly-adenylated RNAs [200]. This might require the fragmentation of longer molecules. Then, cDNA is produced from the library, which is used as an input for the supplier specific sequencing protocol. The result is a large number of sequenced reads from the input RNA library, along with quality values.

2.3.2. Normalization and Analysis

Several computational challenges arise during the process of RNA-Seq. This includes base calling, which involves the conversion of intensities into a corresponding sequence of DNA bases. Base calling must adjust to the properties of the underlying technology.

The primary output of RNA-Seq is a large number of short sequenced reads. The aim of the analysis of such data is to calculate an expression measure from it. In general it is assumed that the expression level of a gene is proportional to its number of reads. As the read length is short, many overlapping reads are generated for all expression genes. The first step of analysis involves the *alignment* of all reads to a reference genome. For this purpose, traditional pairwise alignment techniques based on dynamic programming are too slow. Therefore, several tools like Bowtie (based on the Burrows-Wheeler transform), MAQ, RazerS (both using hashing techniques) have been developed to overcome this problem (see [117] for a overview). A challenge is to allow for gapped alignments, which is useful to map reads that span exon/intron boundaries. The identification of chimeric transcripts from distant loci is also of interest.

The alignment gives a raw number of reads for each exon. The values need to be *normalized* to be comparable within and between experiments. A well established method for this purpose is $RPKM$ [136] – Reads per Kilobase per Million Reads. Let C be the reads mapped to an exon, N the total number of reads, and L the length of the exon in base pairs. Then $RPKM = 10^9 \frac{C}{NL}$ [136]. This method normalizes the mapped reads per kilobase of exons (which normalizes the arrival of reads within the sample) by the number of overall reads generated in the sample. 10^9 is a scaling factor which compensates for the order of magnitude of the denominator. Normalization is also necessary to ensure comparability between samples. For this purpose, global scaling methods [151] and model based procedures [128] are used. The effects of on the detection of *differentially expressed* genes of several normalization strategies, including quantile normalization are compared in [24]. Besides general topics in gene expression data analysis as summarized in section 2.5, like identifying differentially expressed genes in comparative studies, the identification of *novel transcripts* is possible.

2.4. Metabolomics

The *metabolome* is defined as the population of low weight molecules found in an organism. These molecules are substrates and products of physiological reactions. Therefore, measuring metabolites is considered, in theory, to be the most direct analysis of processes in an organism. *Metabolomics* is the process of analysis of this populations and complements transcriptomics and proteomics in systems biology research. Like the other "omics", metabolomics aims at acquiring quantitative measurements. In connection with functional genomics, metabolomics analysis can help to identify the function of genes with phenotypes that would be otherwise invisible [131].

The complexity of the metabolome is currently a subject of research. It has been conjectured that 200,000 different metabolites are present in the plant kingdom [67], while the respective number for human is up to 20,000. Currently, the human metabolome database lists 7900 metabolites [207]. This potential gives rise to various applications, including toxicology, profiling of diseases, nutrition, and discovery of drugs and natural products [131]. Clinical applications of metabolomics encompass efforts to identify biomarkers, which can indicate whether or not a certain medical condition is present in a patient.

The analysis of the metabolome employs general technologies from chemical analytics. For metabolomics quantification, experimental setups involving molecular or mass spectrometry are used, commonly preceded by gas or liquid chromatography. For liquid samples, nuclear magnetic resonance (NMR) spectroscopy can be used [67]. Challenges in this area arise from the multitude of metabolites, methods for sample preparation, especially addressing the experimental bias resulting from sample preparation. Computational challenges include the deconvolution of NMR spectra, and the identification of metabolites from mass spectrometry and chromatography data. A frequently used solution for the latter problem is XCMS [167].

Different strategies for metabolomic studies are used: *Target analysis* of metabolites involves the profiling of a limited number of metabolites, which are relevant for the process of interest. It requires the actual identification of the metabolites as well as an absolute quantification [35]. In large scale comparative studies, statistical analysis, clustering and other analysis methods can be applied. Time series of concentrations of identified metabolites can be used to infer biochemical models. In contrast, *metabolite profiling* requires no identification and relies on unsupervised methods like dimension reduction and cluster analysis to characterize and compare samples. Furthermore, metabolomics data can be acquired either in *metabolomic footprinting* studies, in which extracellular molecules are investigated and *metabolomic fingerprinting* which focuses on intracellular molecules. The latter technique suffers however from low reproducibility that arise as a consequence of the sample extraction procedures.

2.5. Gene Expression Data Analysis

The analysis of gene expression using the above tools, especially when they give a quantitative measurement, leads to *raw data* that needs to be processed. In the case of microarray data, this is usually a set of images along with spot intensity values for each image. This data, however, is usually not ready for drawing conclusions from it. Raw microarray data contains effects from several sources of variation, which should be reduced by data processing and normalization in order to obtain high quality data.

The analysis of gene expression data requires several steps, with numerous potential methods available for each step. Unfortunately, there is no "silver bullet", no method that fits for all problems. Instead, depending on the technology and the experiment, methods must be chosen carefully. In this section, common methods are discussed for the analysis of gene expression microarrays, beginning with *normalization*. Furthermore, methods for the *statistical* and *higher level analysis* of microarray data are described. These methods usually require an expression matrix as an input and therefore generalize well to other experimental platforms.

Every technology requires a thorough quality control, that assess the quality of the resulting raw data, based on the specific sources of variation and technological problems. For microrarray data, it is common to remove at some point the spots that failed to meet quality criteria during the scanning step and the normalization steps.

2.5.1. Microarray Data Normalization

This section is focused on expression microarrays. Resequencing microarrays require different treatment, which is discussed indenpendently in section 4.1. Normalization of microarray data is necessary due to the random effects that are imminent to microarray experiments. A large number of variation sources has been identified [47, 208] that affect all steps of a microarray study.

For the normalization of two-channel microarrays, typically three steps are performed, which mostly are also necessary for all other types of microarrays.

1. *Background Correction.* This step is necessary to eliminate the error in the spot intensities that arise from unspecific signals. The background is not uniform over the array, and may have strong spatial effects (see figure 2.6 for an example). An additive model is often used for modelling the background [208]: The measured signal S is given by $S = B + T$, where B denotes the background signal and T is the true signal. B and T cannot be measured directly. Therefore, most background correction methods subtract an estimate B' from S in order to obtain an estimation for the true (foreground) signal: $\hat{S}$ as $\hat{S} = S - B'$. Several methods for globally and locally estimating the background have been introduced. Depending on the background correction method, spots with negative signal may result ($\hat{S} < 0$), which is undesirable and often mitigated by adding an offset or setting $\hat{S}$ to a minimum positive value. For many application, the *normexp* method which estimates the true background value via a convolution model gives satisfactory results in combination with an offset value [148].

2. *Within-Array Normalization.* This step aims at making data comparable within one array. It is most important in the analysis of two-color arrays, where the aim is to remove the bias that results from different properties of the commonly used fluorescent dyes. The dyes *Cy3* (green) and *Cy5* (red) differ in their incorporation properties and their photo bleaching properties. Furthermore, the dye effects can depend (not necessarily in a linear way) on the spot intensities [208]. *Dye swap normalization*, which exploits two hybridizations with inverted dyes, can be used to solve this. More frequently, *intensity dependent dye normalization* (see for example [208, 47, 116]) is applied to this problem. For this purpose, a regression model is fitted on the transformed red and green channel intensities, R_i, G_i. Then, the MA-transformation is given as

   $$M_i = \log_2(R_i) - \log_2(G_i)\, A_i = \frac{1}{2}(\log_2(R_i) + \log_2(G_i)) \tag{2.1}$$

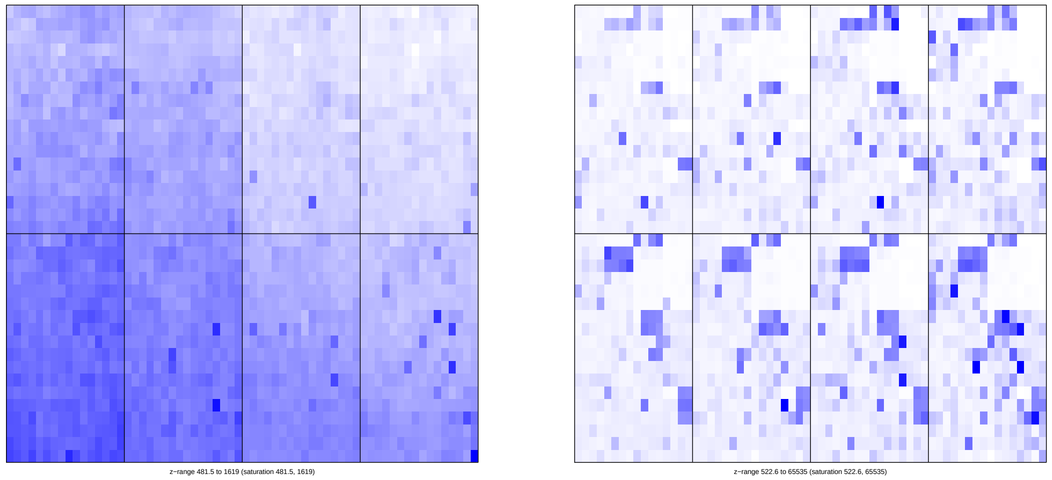
   This transformation gives rise to the MA scatterplot (see figure 2.7) to which a smooth curve f is fitted, using either all, or an invariant subset of the genes [157]. For this, the LO(W)ESS (Locally Weighted Regression) methods (using linear or polynomial regression models, see [47] for an overview) has become the standard procedure, however many other methods have been devised and successfully applied, for example smoothing splines [208]. The corrected values $\hat{M}$ and $\hat{A}$ are calculated as

   $$\hat{M}_i = M_i - f(A_i)$$
   $$\hat{A}_i = A_i$$

   Figure 2.7 shows an example of this procedure. From this point on, the data is represented on a $\log_2$ scale, which is has a variance stabilizing effect and allows an easy interpretation of fold changes [208].

3. *Between-Array Normalization.* Subtle differences between the processing of two microarray slides in the laboratory make between array normalization necessary. A standard method is scaling the data so that the mean or median value of each slide match [208]. Quantile normalization [17], which makes the value distributions of all arrays identical is often used for between-array normalization.

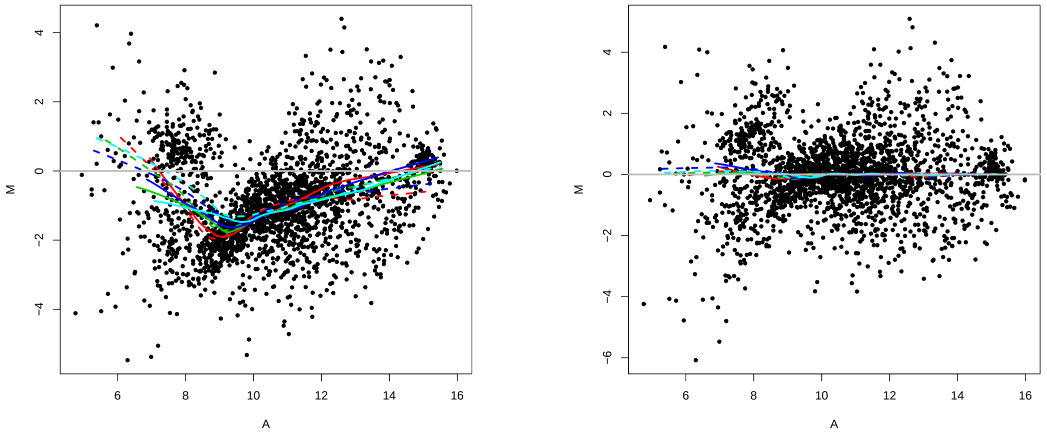
For single channel microarrays, especially Affymetrix GeneChip$^{\text{TM}}$s, slightly different steps are performed, for example in background correction, where several specialized methods exist. Since only a single channel is available, no dye effect normalization is necessary. Intensities of the mismatch probes are used to correct for unspecific hybridization and to estimate the true probe values. Finally, the overall expression of a probe set is estimated from the corrected probe values (using Tukey's Biweight in the MAS5 package). Between-arrays normalization is also performed. Usually, fixed collections of methods for background correction, normalization and probe set expression level estimation like MAS5, dChip and RMA are used. Zhu *et al.* [213] have investigated more than 150 different options for normalization using a large scale spike-in dataset. Their study concluded that a process with the GCRMA (a variant of the RMA method) background



(a) Background intensities

(b) Foreground intensities

Figure 2.6.: Background and foreground intensities of a cDNA microarray. The spatial pattern of the background could lead to problems without adequate background correction. The two images show the intensities using white-blue color gradients, however, (a) and (b) use different scales.



(a) *MA* plot (unnormalized)

(b) *MA* plot (normalized)

Figure 2.7.: Effect of loess normalization: (a) shows the *MA* plot of a cDNA microarray before normalization, along with several curves fitted to the data; (b) shows the same data after normalization.

correction and median polish probe set summarization outperformed all other methods. Variance stabilization [80], without any background correction also performed well. Other platforms, e.g. the Illumina BeadArrays require similar, but adopted strategies for background correction and normalization.

The normalization of microarray data is complex, and a host of different options is available. Successful normalization requires choosing an adequate set of methods. *Quality control* on all levels is necessary to ensure high quality data. This involves identification of poor quality array features or even arrays with an overall poor quality, which are then excluded from further analysis. In most steps, visualization can help the researcher to make decisions. For a summary of visualization tools, see chapter 3.

After performing these steps, the data is available in the form of a background corrected, normalized, $\log_2$ transformed matrix, which is the *primary data* used in further analyses. Usually, for two-channel data it is given in the form of a matrix of fold changes or, from single channel data, as a matrix of normalized probe expression levels. Generally, we refer to this matrix from now on as the *expression matrix.* A row in this matrix represents the expression profile of one gene (sometimes called a probe) in various samples or experiments.

2.5.2. Statistical Analysis

Gene expression studies are often comparative in nature. The aim of many studies is to characterize the differences between two or more groups of samples. These groups may arise from different conditions, cell lines, treatments, diseased or healthy tissues, or any other biological or medical factor that is of interest. Time series data is also of high interest. Here, differences between time points or between before and after perturbations happened are investigated.

For this purpose, one is interested in finding the genes that are *differentially expressed* between conditions. The $\log_2$ *fold change* of a gene g expressed in groups a and b is defined as $fc_g = \overline{g_b} - \overline{g_a}$, where $\overline{g.}$ denotes the average expression value of g in a or b. Due to the variation in the values, it is not sufficient to calculate fc_g and declare g to be differentially expressed when $|fc_g|$ is higher than some cut-off. Instead, the variance of the must be taken into consideration. For this purpose, the whole repertoire of statistical tests can be used. Classical statistical tests like the t-test or the Wilcoxon rank-sum test have been applied as well as microarray specific adoptions, like SAM [193]. Numerous non-parametric methods with different test statistics have been developed, examples are RankProduc [23] and Weighted Average Difference (WAD) [87]. The performance of various methods, also in dependence to normalization methods, have been evaluated in [213] and [87]. For the identification of differences between multiple groups, ANOVA and the non-parametric Kruskal-Wallis Test can be employed.

Statistical significance, which can be found by the above methods, is not a sufficient criterion for biological relevance. Here, the magnitude of the fold change should be taken into account as a filter: genes with $|fc_g| \leq 1$ are usually not considered to be differentially expressed.

The *multiple testing problem*, especially when analyzing large whole-genome arrays, requires a correction of the p-values resulting from the various methods. Single step methods for controlling the family-wise error rate (FWER) like the Bonferroni or Šidak procedures are often considered too conservative. Instead, the Benjamini-Hochberg step-down method for controlling the false discovery rate (FDR) is often used [208]. In addition, the number of tests performed can be reduced by preprocessing steps, like cluster analysis (see below) or removing invariant genes from the analysis.

2.5.3. Cluster Analysis

Cluster analysis is among the most frequently used steps during gene expression analysis. Identifying clusters of co-regulated genes allows to structure the data into smaller units which makes handling and visualization easier and more effective. Furthermore, it is conjectured that there might be biologically relevant relationships between co-expressed genes: co-expression might indicate co-regulation. Similarity of expression profiles, is, in most clustering methods, measured using one of many distance measures. The most important among them are the Euclidean distance, the Pearson correlation distance (which is invariant to scaling) and the Manhattan metric. However, various others distance measures exist [47].

The usage of *hierarchical clustering* methods on microarray data has been popularized by Eisen *et al.* in 1998 [52], who used average linkage clustering on expression profiles from a study on *Saccharomyces cerevisiae*. Following this influential paper, various other linkage methods and methods originating from phylogeny, like Neighbor Joining, have been used. Usually, a heat map with the resulting clustering tree is used for visualization. Clustering the experiments is also common, as it allows to inspect the similarity of experiments and to discover classes.

Partitioning clustering algorithms allow to separate genes into an (often predefined) number of clusters. The k-means clustering method [121] and adaptations that allow the usage of other distance measures allow a relatively fast clustering into k groups. Self-organizing maps (SOM) provide a better and meaningful topology of the cluster centers. Quality-based clustering [75] requires no predefined number of clusters and can reject to cluster genes.

Further topics in clustering microarray data includes using dimensionality reduction methods like Principal Component Analysis (PCA) to induce and visualize clusterings. Quality assessment and comparison of clustering results are as well of interest as biclustering, which allows to simultaneously identify similar subsets of genes and experiments. For an overview, see [32] or [47].

Often, dimensionality reduction methods are used, most commonly principal component analysis (PCA). The aim of PCA is re-expressing the input data in a way that best reflects the variance of the data and reduces redundancy (see [47] for an overview) For this purpose, a basis transformation is performed, the new basis are the eigenvectors of the covariance matrix of the input. Let A be a centered matrix, then for every row (column) x, y, let $\bar{x}$, $\bar{y}$ denote the expected values. Then, the covariance is defined as $\mathrm{Cov}(x, y) = E((x-\bar{x})(y-\bar{y}))$. E denotes the expected value function. In this way, the covariance matrix C is calculated from each row (column) of x. Then $C = VDV^{-1}$, where V is a matrix consisting of the eigenvectors of C, ordered by their eigenvalues; D is a diagonal matrix of the ordered eigenvalues. The sum of the first n ordered eigenvalues is the fraction of retained variance in the first n components. Let W be the first n columns of V. Then $x' = xW$ is the n-dimensional reduction of vector x.

2.5.4. Higher Level Analysis

Based on the results of statistical analysis and cluster analysis, several higher level analysis steps can be taken. These usually include external data and extend the focus beyond the statistical analysis of the data.

Biological questions are seldom answered with a single gene. Often the overall regulation of a whole set of genes is of interest. Common gene sets of interest are for example genes related to a metabolic pathway or genes that have a certain Gene Ontology [6] term. For identifying differential behaviour of gene sets, several methods for *gene set enrichment* have been proposed. Subramanian *et al.* published GSEA – Gene Set Enrichment Analysis [175]. Over-representation analysis and several similar methods based on hypergeometric tests and other parametric and non-parametric test are also used.

For many comparative studies, especially in cancer research, it is interesting to build a model for automatically distinguishing between two or more classes of samples. A landmark study was performed by Golub *et al.* [66] who presented a *classification model* to distinguish between two forms of leukemia. Following studies employed a large variety of classification models on different microarray studies, with varying results. Common procedures are feature selection to identify genes that discriminate between classes and the training and evaluation of classification models. See [209] for an overview of methods. Other medical applications of machine learning are using gene expression profiles to predict the disease outcome or the survival time.

Further topics in higher level analysis of gene expression data encompass the *inference of regulatory networks*, the prediction of operons or the validation of non-coding RNAs. In general, for the generation of biological hypothesis and for the coming to conclusions from gene expression data, external data must be referred to. This encompasses many different sources, from gene annotations to biological networks. A discussion of such meta information and ways for extracting relationships from expression and meta data is provided in chapter 5.

Much of the knowledge of the biomedical community is condensed in a multitude of biological, biochemical, signaling and other *pathways*. Pathways represent series of biochemical reactions and are an important tool for understanding and structuring the overwhelming number of physiological reactions. They constitute a condensed conceptual representation of biological knowledge, which is gathered from many studies. Using pathways, including the topology, for the analysis of gene expression data, can give good answers to many biological questions. The possibility of crosstalk between pathways underlines, that biology is in fact a network-based science [10]. Therefore, gene expression must be seen in the context of networks, tools for which are introduced in chapter 6.

3. Data Visualization

In this chapter, the theoretical foundation of this work is summarized. This chapter starts with a short summary of the topic of data visualization. The basic principles are discussed, as well as important data analysis strategies based on visualization and applications to gene expression data (section 3.1). Furthermore, the most important visualization plots are described (section 3.2). An equally valuable tool for dealing with complex and high dimensional data are graphs. A brief overview of graph theory and visualization is given in section 3.3. The combination of visualization and graphs is relevant for working with biological networks, which are described in section 3.4. A step away from the theoretical framework, Mayday, the technical foundation of this thesis is summarized in section 3.5.

3.1. Data Visualization

Data visualization is the process of turning data into an image that represents the data and allows to answer questions related with the data. This is achieved by choosing a *visual representation* of the data which may amplify several aspects and omit others. In general, data visualization allows important insights into data, even by simple means. A landmark example for the importance of inspecting data before performing statistical analyses is *Ansconbe's quartet* [5]. It contains four datasets of two variables x and y. For each dataset, mean and variance of x and y, respectively, are equal or nearly equal. The correlations between x and y and linear regression line are also practically equal for all four datasets. However, as it can be seen from figure 3.1, the datasets differ greatly in shape, what is difficult to detect by statistical analysis alone. Furthermore, in datasets 3 and 4, outliers have a large influence on the characteristics of the datasets, especially for dataset 4, where the regression line could not be calculated without the outlier.

One of the earliest application of data visualization were *geographical maps* [188]. Maps represent large datasets: area boundaries or roads and topological features are represented as polygons or polylines supported by a large number of vertices. An example that highlights the importance of visualization of geographical data is John Snow's map of deceased patients during a cholera outbreak in London, 1854 [189], which helped identifying a contaminated water pump. No summary of data visualization would be complete without mentioning Joseph Minard's high-dimensional map-based visualizations of Napoleons military campaigns, most importantly of the invasion of Russia (see figure 3.2a). Early applications of visualization of *time series* data arise in astronomy beginning the 10th century C.E. [188], although the first concise uses date back to the late 18th century. The same era gave rise to the first visualized economical statistics by William Playfair (see [188] for a review), who invented *line graphs*, *bar charts* and *pie*

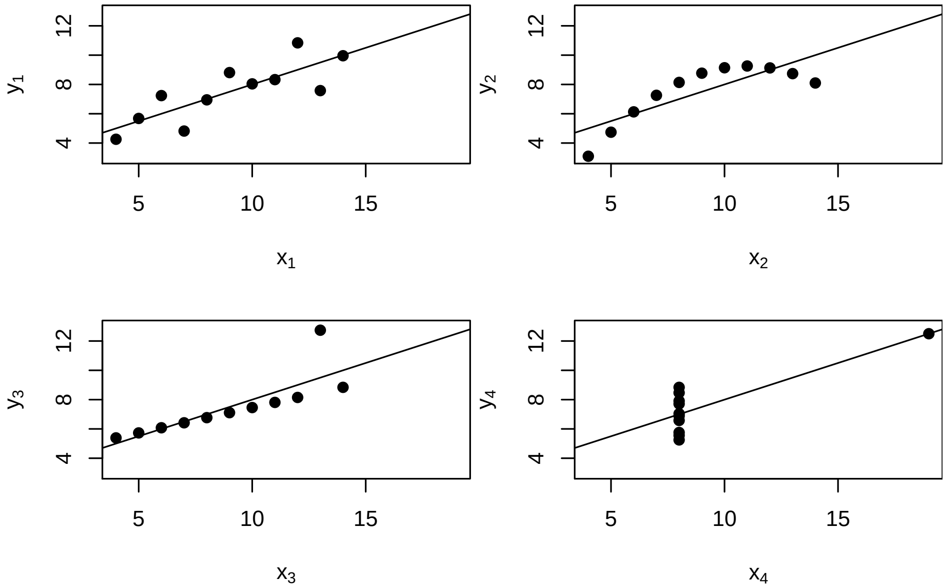
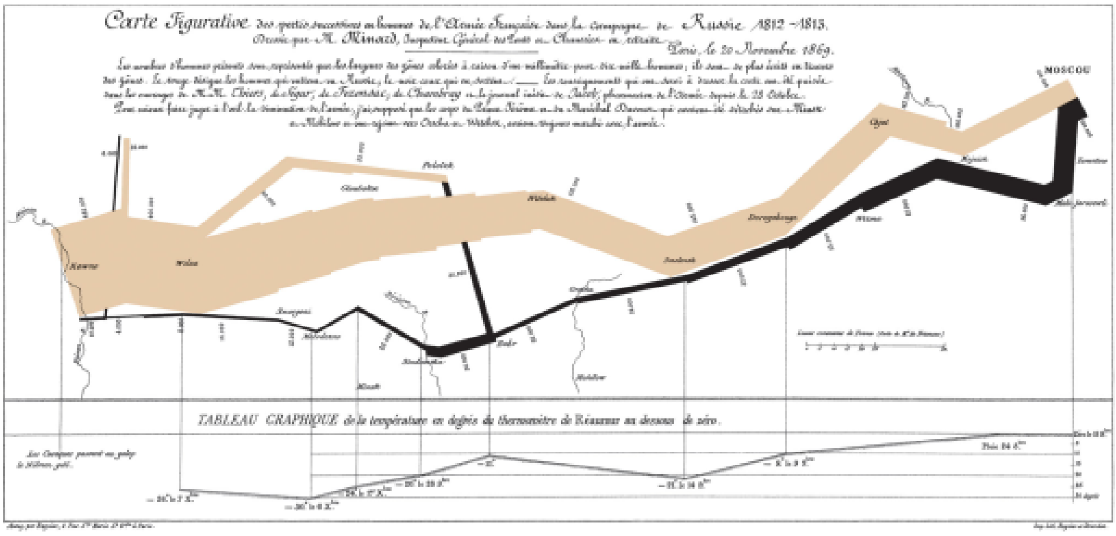


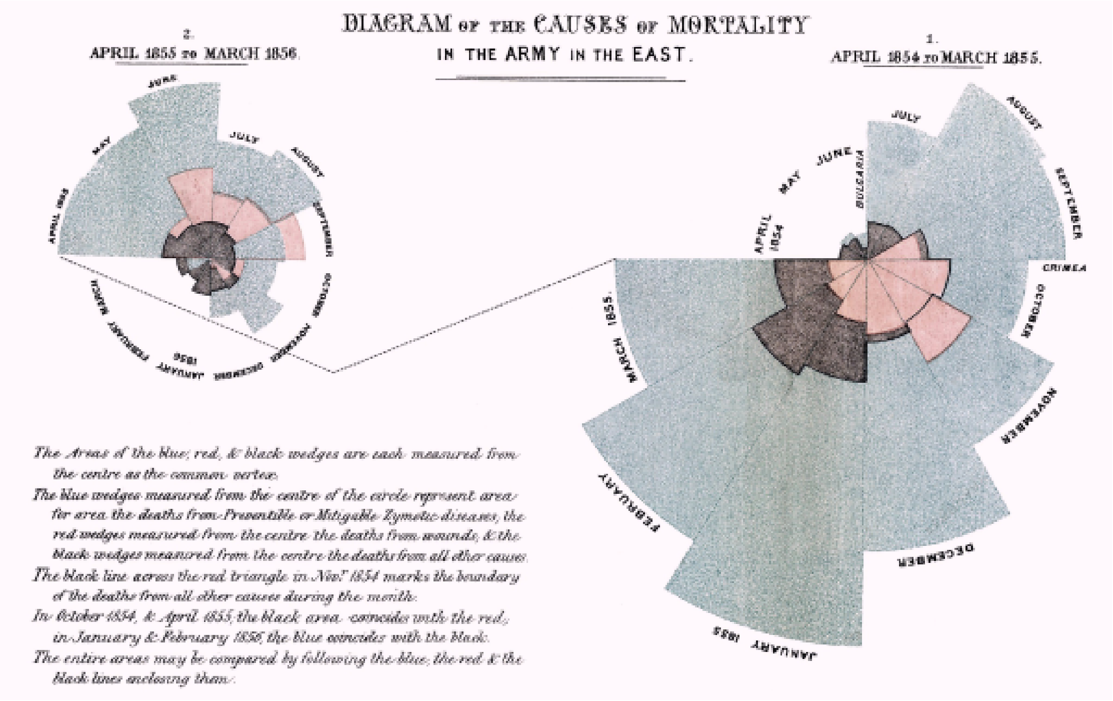
Figure 3.1.: Anscombes Quartet. Several important properties of the datasets are identical, however they differ greatly in shape. Note that the outliers in datasets 3 and 4 dramatically change the properties of the dataset.

charts. This started the era of decision making based on charts. The *radial bar plots* presented by Florence Nightingale on causalities during the Crimean war (1853 – 1856, see figure 3.2b) showed that most soldiers died from preventable causes and triggered decisions to enhance health care and sanitation. Enhancements of data processing (first patents on punched cards by Herman Hollerith in 1889), statistics and printing allowed advances in data basis, interpretation and presentation of visualizations. However, the focus in statistical analysis laid mostly on mathematical methods. This was attacked by works by Tukey [191, 192] and Anscombe [5], which paved the way for *exploratory analysis* of data. The introduction of *parallel coordinates*, *Chernoff faces* and the increased use of *heat maps* allowed to produce *multivariate* visualizations. Conceptual work was first done by Bertin [27], later Wilkinson formulated data visualization as a formal grammar [205]. Quality considerations (e.g optimizing “data ink” [188]) and inclusion of results of cognitive science, psychophysics and other disciplines gave rise to new topics and fields, including visual analytics. Computer graphics and data visualization became a successful combination, greatly increasing creation speed and accuracy of visualization.

Current challenges include the handling of large scale heterogeneous data, arising form biology, astronomy and physics. Interactivity and especially collaborative work are further topics of research. Presentation of visualizations (“story telling”, [161]), info graphics, and data journalism are further aspects of data visualization that currently receive much attention, also from the general public.


Carte Figurative des pertes successives en hommes de l'Armée Française dans la Campagne de Russie 1812-1813.
Dressée par M. Minard, Inspecteur Général des Ponts et Chaussées en retraite.
Paris, le 20 Novembre 1869.
MOSCOU
Kowno
Wilna
Polotzk
Witebsk
Smolensk
Dorogobouge
Chjat
Mojaisk
Wiazma
Orscha
Bobr
Studienska
Minsk
Molodezno
Smorgoni
Mohilow
TABLEAU GRAPHIQUE de la température en degrés du thermomètre de Réaumur au dessous de zéro.

(a) Minard's Map of Napoleon's Invasion of Russia



(b) Florence Nightingale's Radial Bar Plots

Figure 3.2.: Classical examples of data visualization. (a) shows the Minard's map of Napoleon's invasion of Russia, which shows latitude, longitude, number of surviving men, and temperature. (b) Nightingales chart shows the causes of soldiers dying in the Crimean war. (Sources: (a): `en.wikipedia.org/wiki/File:Minard.png` (b): `en.wikipedia.org/wiki/File:Nightingale-mortality.jpg`)

3.1.1. General Concepts

Ben Fry discusses in [58] eight steps that summarize the actions necessary as well as the challenges during the process of data visualization. These steps begin with *data acquisition*, *parsing* into data structures, *filtering* in order to find the data of interest. Then, data is to be *mined* for interesting patterns and a suitable visual *representation* is to be found, and *refined*, i.e. optimized to better convey the particular information. Finally, *interaction* methods are to be added [58].

This is only one possible model for data visualization. Wilkinson describes a "Grammar of Graphics" [205], based on object oriented design. In this grammar, statistical graphics are defined in a number of statements that describe *data*, *variable transformations*, *scale transformations*, the *coordinates system*, the elements displayed, and *guide items*. Following Wilkinson and Fry, several aspects of visualization are discussed here.

Data acquisition depends on the general field of application. Methods for data acquisition of gene expression data are described in the previous chapter, including methods for preprocessing and statistical analysis. Identifying the important parts of the data is done using statistical and clustering analysis. As data structures, (subsets of) high dimensional expression matrices are relevant.

The *representation* step involves choosing the graphical representation of a dataset, often using one or more of the plot types described in section 3.2. Further aspects are transformations of the variables to be visualized. Examples are sorting, aggregating (e.g. for boxplots and histograms), discretization and numerical transformation (often logarithmic, exponential etc.).

Another common preprocessing step for data visualization is *dimensionality reduction*. The most common method for this is principal component analysis (PCA); similar methods are singular value decomposition and multidimensional scaling. A related non-linear alternative is kernel PCA.

The *scale of measurements* also influences the data representation. Scale theory was introduced by Stevens in his influential 1946 paper [171]. Stevens identified four scales of data which are characterized by possible mathematical operations and statistical operations

Nominal data is defined as data for which no order can be defined; for examples for person names. The mode and the equality of two measurements can be determined. *Ordinal* data can be ordered, i.e. it is possible to determine whether an object is greater or less than another one. The median and other percentiles can be calculated. School grades are an example. *Interval* data are quantitative measurements made on some interval with arbitrary origin. Additions can be performed on this data, and it allows to calculate mean, variance and correlation coefficients. A common example is temperature measured on the Celsius scale. Calculations requiring divisions are not possible in interval data. *Ratio* data allows divisions of measurements, and geometric and harmonic means can be calculated. Examples are all measurements made in SI units.

The scales of measurements dictate several aspects in a visualization, e.g. the encoding of data, data transformations (of which most are only permissible for ratio data), axis labellings and, potentially, the kind of plot used: for plotting a function graph for nominal data is not possible; instead, a bar chart can be used.

Different *coordinate systems* are used for data representation, most commonly *Cartesian coordinates* in two or three dimensions. Alternatively, *polar coordinates* are applied e.g. to pie charts and radial bar plots. Fisheye transformations direct focus to central elements of a visualization. *Barycentric coordinates* are based on a two dimensional projection (resulting in a triangle) of the three-dimensional space. Also four dimensions can be displayed using tetrahedrons. *Parallel coordinates* (see below) are another useful way of transforming high-dimensional objects into the plane.

The *refinement* step in Fry's process of visualization involves enhancing the clarity and directing attention to selected parts of the data [58]. Possible methods are manual selection of data points to be distinctly represented. Scaling or coloring objects based on relevance measures is also helpful, see [63] for examples. Refinements may also include adding *guides*, i.e. axis descriptions, labels or legends the clarify the meaning of graphical elements in a visualization [205].

The usage of *color* requires some considerations. While color is an indispensable tool for many visualizations, for example heat maps, it has been found to be a poor way of conveying information [36]. However, the color scheme used has a great influence on the legibility of graphics. The often-used *rainbow color map*, which is based on the wavelength of lights of several colors, is not optimal in several aspects. In general, a *perceptual ordering* is required for color maps that should support comparing relative values. The order of values represented by such colors is intuitively clear. Varying *luminance* allows to identify small details. Color maps may not cause artifacts in visualizations. The grayscale color map, black body radiation and green-red color maps are recommended [18]. Tools suggesting color gradients for various purposes, including addressing color vision impaired audiences, exist, e.g. ColorBrewer [71]. Useful selections depend on the type of data, for ordinal data, the gray scale map, and the black body radiation map is recommended [18]. The color gradient classically used in heat maps has not yet received much attention by the perception community.

Interactive visualizations allow the user to explore the data beyond the scope of the initial view [58]. This is especially useful for high dimensional data when the point of view is selectable, i.e. in 3D graphics. In large and detailed visualizations, zooming allows to inspect the data in greater detail, or get a better overview. Gradually moving over the visualizations (panning) allows serial inspection of details. Selecting (picking) data points, is useful for highlighting data points, and possible preparing manually chosen data points for further use, i.e. in a refined visualization or in a confirmatory analysis step.

The concept of *coordinated multiple views* means to have several views of the same data, which are connected and exchange operations performed on the data and its representation ("brushing and linking"). This allows to view several levels of the data, e.g. an overview and details, or objects in the focus and their context, as well as views that highlight differences [150]. Interactive search operations, allow to focus and refine the visualization on relevant data items. Other operations, like on-line data transformations, filtering, changes of refinements or even addition of new data allow users to redefine the visualization and interactively explore a dataset.

There are many applications of data visualizations in gene expression data analysis. Visualization is used throughout the process of measuring and interpreting gene expression. Beginning with data normalization, visualization is a standard tool. For quality control, image plots of microarrays are used (see figure 2.6). *MA*-scatterplots are important tools for assessing the normalization of two-channel microarrays (figure 2.7). For general assessment of normalization, box plots comparing multiple arrays are used. In statistical analysis, volcano scatterplots are used to view the results. Up to this point, visualizations are mostly used in a static way, however often with optimal data preparation and representation. Clusterings are visualized using profile plots and heat maps. Higher level analyses often requires networks to be visualized. For all these topics, interactivity is important.

3.1.2. Exploratory Data Analysis

Exploratory data analysis is a term coined by John Tukey in his groundbreaking 1977 book with the same title. Exploratory data analysis (EDA) describes a concept for investigating data. Aims of this approach are commonly gaining insight into a dataset and its structure. This involves finding outliers, important relationships and formulating hypothesis. Furthermore it can be aimed at quality control and choosing confirmatory tools. A main idea is not to engage into statistical testing before an overview of the structure of the dataset is achieved.

For exploratory data analysis, many visualization tools are used, including the stem-and-leaf plot, histograms, scatterplots, box plots and numerous variants of them. Combinations of visual tools with basic statistics, i.e. mean, standard deviation, median and other measurements enrich the plots and allow further insight. Regression is also commonly used. Numerical summaries like the five number summary (see section 3.2.4) were also introduced in the book [192]. Further relevant tools are data transformations (log, square root) and data smoothing using nonlinear smoothing.

The impact of EDA to data analysis is huge and the methods and concepts suggested by and following Tukey are today common tools for data exploration. While Tukey concentrated on numerical (mostly ratio and interval) data, the focus of EDA is today much broader. Applications of EDA are also relevant to the exploration of graphs and network, like in biological pathways [97]. EDA can be seen in contrast to confirmatory data analysis. The aim here is confirming the relationships found during exploration. Methods relevant for this are statistical hypothesis tests or bayesian inference; visualization is also possible for confirmatory analysis [96].

3.1.3. Visual Analytics

While the roots of exploratory data analysis were based on manual calculations and hand drawn graphics, possibly enhanced by desk calculators, modern methods greatly increased the speed and handling of visualization and exploratory statistics. Interactivity is a further important aspect made feasible by computer-based visualization. In the same time, however, the size and complexity of datasets increased even faster then the analysis tools were improved [95]. New concepts for making sense of large, noisy and heterogenous datasets are required. *Visual analytics* makes use of interactive visualization to support human cognition to analyze and interpret data. The focus lies on the optimal support of *human cognition*, which is considered a powerful tool. For this purpose, methods and results from various scientific disciplines are integrated, including computer graphics, psychology, cognitive sciences and design.

The overall process of visual analytics can be summarized by the *visual analytics mantra*: "Analyse First - Show the Important - Zoom, Filter and Analyse Further - Details on Demand" [95]. It names some of the tools and strategies used in visual analytics. Analysis methods are used to prepare and filter the data to first visualize concentrating on important features [95]. Visualizations are optimally maximizing data density and should allow to easily identify patterns and relationships. Commonly tools used for EDA are applied [27]. Based on an existing visualization, refinements are interactively made: zooming to get a view that is more coarse or fine, filtering to remove irrelevant items and further analyses. Details on objects are shown interactively on demand. This process is iteratively repeated. Each iteration is aimed at providing a useful visual representation that allows the viewer to make more sense of the data.

As visual analytics is concerned with extremely large datasets, several challenges exist. The limited space on visual media, especially screens calls for scalable visualizations (and larger screens) [96]. Analyzing high-throughput stream data can address data storage problems. Another challenge is the analysis of heterogenous datasets, which arise in many fields, including systems biology. Automatization of processes, decision support and evaluation of existing processes are also fields of research in visual analytics.

3.2. Visualization Plots

In this section, a brief overview of several types of common visualization plots is given. While a large number of plots exist, they can be summarized by a few basic types, of which the most important ones for this thesis are discussed here. Some classical approaches, like Chernoff faces, are left out here, as they are rarely used, especially in gene expression analysis. Also not listed here are tree views or tree maps, which do not directly display expression data but can be used to visualize meta information. Genome browsers employ a subset of the tools described below. They are, while capable of visualizing gene expression data, not in the focus of this thesis.

A classification of the different approaches discussed here can take into account the *graphic elements* used, as well as the number of high dimensional objects and the number of variables that can be plotted simultaneously (see table 3.1). The theory of graphic elements has been introduced by Cleveland *et al.* [36]. A graphic element is the way a plot conveys its information. A ranking of the most effective elements places the position on an aligned scale at the top, and color at the bottom of the ranking. This ranking, has however been challenged, as different tasks give rise to different rankings, and often many

Plot	Strategy	Objects	Variables
Scatterplot	aligned scale	< 10000	2
Parallel coordinates	aligned scale, angle	< 10000	< 100
Box plot	aligned scale	unlimited	1
Bar plot	aligned scale, length, area	< 100	1
Histogram	aligned scale, length, area	unlimited	1
Heat map	color	< 10000	< 100
Pie chart	angle, area, length	< 10	1
Radial bar plot	area, length	< 20	1
Star plot	length, non-aligned scale, angle	< 20	< 10000

Table 3.1.: Properties of visualization plots. The table shows the name of the plot, the strategy for conveying information it uses (partly based on [36]) and typical maximum numbers of objects that can be visualized and the number of variables that can be shown for these objects.

elements perform equally well [172]. Most plots use make use of two or more elements, and most of them usually convey information effectively. Heat maps, while very dense, use color, which might be an issue, but it has been found that for interpretation tasks, color shade and saturation can be interpreted with high accuracy [172].

In general, plots capable of displaying high dimensional data are used for many purposes. However, it is often useful to provide multiple plots with a reduced number of elements, especially when a clustering or classification of objects is available. This concept is sometimes called small multiples [188]. Usually, multiple plots of the same type, with the same scales are presented. The small multiple concept is well established and their first use can be tracked back to the late 18th century [188].

3.2.1. Scatterplots

Scatterplots are commonly used to visualize two variables x and y, for n objects, i.e. $x_1, \dots, x_n$ and $y_1, \dots, y_n$. Usually, continuous variables are used, however general ordinal variables are also possible. Data from all scales has been visualized using scatterplots. Scatterplots work by visualizing points (x_i, y_i) for all i in Cartesian coordinates (an example can be found in figure 3.1). Applications of scatterplots are the visual comparison of two variables. With scatterplots, the correlation of x and y can be detected. Therefore, scatterplots are often extended by plots of regression functions. Additional data dimensions can be encoded by color, point size or using different symbols. A problem often observed in scatterplots is overplotting. If a large number of points is to be drawn on a small area, they overlap each other. In this case, an estimate of the density of points is useful, for example using continuous scatterplots [7].

Many special cases of scatterplots exist. They arise from applying transformation to the input variables. In microarray data analysis, *MA*-plots (see eq. 2.1, figure 2.7) are used which present a rotation of the data by 45° and are in principle not limited to microarray data. QQ-plots are used to compare the quantiles of samples, in order to infer whether they are drawn from the same distribution. Dimensionality reduction methods, especially PCA use scatterplots to show the data reduced to the first principal components.

Three-dimensional scatterplots are also routinely used for visualization of three variables. Here, interactivity is important, to inspect the data from different view points. For high-dimensional data, a scatterplot cannot be extended to more dimensions. Instead, *scatterplot matrices* are used. They show a scatterplot for each pair of variables. Large scatterplot matrices can be summarized to correlation matrices using ellipses to denote the correlation. A scatterplot itself can be subject to statistical analysis. The technique of *scagnostics* aims at extracting descriptors from scatterplots that describe their shapes. This is usually applied to and displayed by scatterplot matrices.

3.2.2. Parallel Coordinates

Parallel coordinate plots, in the context of gene expression data usually called *profile plots*, are a method for visualizing high dimensional data. A point $p \in \mathbb{R}^n$ is drawn on n parallel axes by placing i-th vertex of a polyline on the position of the i-th axis that represents the value of p_i. A large number of points can be jointly visualized in parallel coordinates. For interpreting the parallel coordinate plot, the order of the of the parallel axes must be known. Parallel coordinate are used for a discrete number of dimensions, like in discrete time series data. For this data, parallel coordinates are especially useful, as the slope of the polylines is proportional to the difference between two adjacent time points. The coordinates can also be spaced proportional to the distance between two time points. Figure 3.3 shows an example of a parallel coordinate plot for time series data.

Parallel coordinates were introduced in 1959 by Alfred Inselberg (a previous description of this concept was published by d'Ocagne in 1885) [83]. Since then, parallel coordinates have been used in a multitude of applications. An influential paper of Wegmann [204] demonstrated several use cases and interpretations. it included high-dimensional geometric objects and cluster visualization, which is one of the most common applications of parallel coordinates. For this purpose, color is used to indicate cluster membership. Parallel coordinate plots can be extended by adding additional dimensions, for example showing statistical properties of the points.

Plotting many points in parallel coordinates can lead to overplotting. To address this problem, a number of dimension reduction methods have been suggested, e.g. [89]. Alternatively, clusters can be represented by centroids, leaving out all other points. Using semitransparent lines gives a better overview of the density of lines in a plot. The number of dimensions in a parallel coordinate plot is not generally limited. However, a large number of dimensions might lead to tightly spaced coordinates, which can be hard to read. For time series, an aspect ratio that causes the average slope of a line segment to be 45° is considered optimal [74]. This might lead to a trade-off between size and readability.

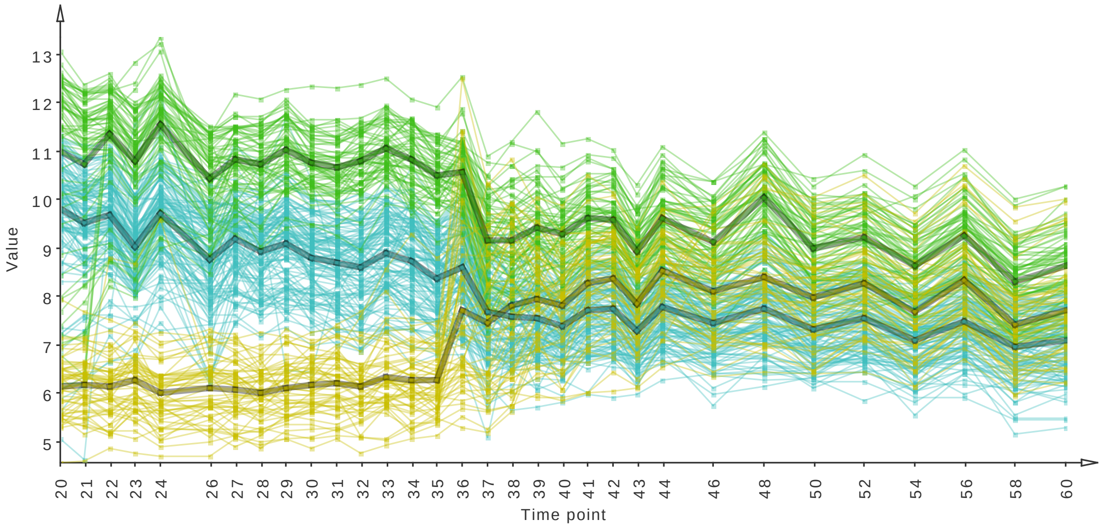


Figure 3.3.: Parallel coordinate plot (profile plot) of three clusters of gene expression profiles in a time series. The coordinates are spaced proportionally to the time points. The centroid profiles of the clusters are shown using thick lines.

3.2.3. Heat Maps

Heat maps are a tool for visualizing a large number of high dimensional objects. Let $M_{n \times p}$ be a matrix, e.g. an expression matrix, with n objects described by p variables in arbitrary but fixed order. Let c be a function that maps values of M to a color, proportional to the value. Then a heat map displays M as an $n \times p$ array of colors, given as $c(M_{i,j})$ for all elements of M. In general, for the entire heat map, the same color gradient is used and all cells are drawn as rectangular boxes. As heat maps rely on color for data representation, a color palette must be defined that maps the values of M to a corresponding color in linear or sigmoid way; common color gradients are green-black-red or blue-white-red.

Enhancements to heat maps are made by hierarchically clustering the objects or variables to induce a reordering of rows or columns. This allows to group together similar objects and variables. The trees resulting from hierarchical clustering are plotted above and left to the heat map. Further enhancements can be made by adding additional columns, e.g. containing statistical measurements or annotations to the objects [63]. Scaling the height of rows is also possible to indicate their relevance.

Any discrete or continuous matrix can be visualized using heat maps. Social sciences, usability studies, geographical information systems, and finance use heat maps. In gene expression data analysis, heat maps are a standard tool to view (subsets of) expression matrices. The usage of clustered heat maps for this purpose has been popularized by Eisen *et al.* [52]. While Eisen's paper was influential, the fist use of heat maps date

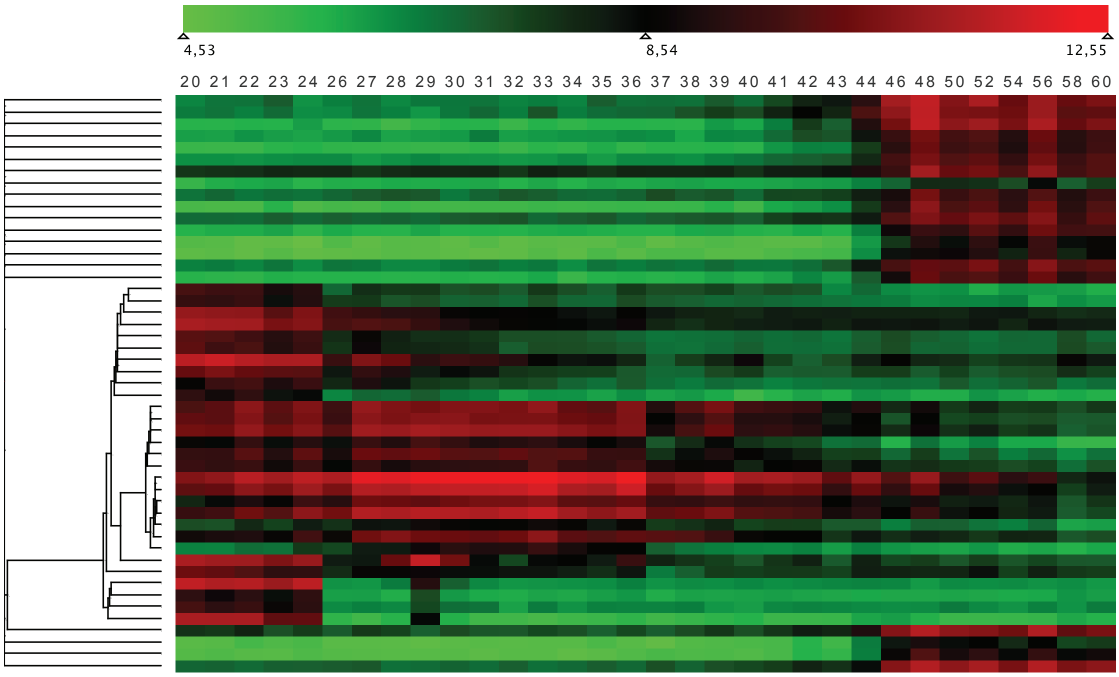


Figure 3.4.: Clustered heatmap showing 49 expression profiles in *S. coelicolor*. The clustering was performed using neighbor joining (using Pearson correlation distance). The color gradient used is depicted at the top of the figure.

back to the late 19th century, and clusterings in heat maps were introduced in the 1970s (see [206] for a historical review).

Similar to heat maps are plots, which visualize a single variable for spatially distributed objects. Examples are thematic maps, potentially with scaling the areas (cartograms), which are used to present statistical values for areas. Image plots are used in microarray quality control to visualize values for individual features (see figure 2.6).

3.2.4. Box Plots

Box plots (also called *box-and-whisker plots*) have been introduced by Tukey as at tool for exploratory data analysis [192]. Box plots are a simple, but effective mean of visually describing key characteristics of a data sample. Let $\mathbf{x}$ be a vector of n numbers, then, a box plot visualizes a *five-number summary*, consisting of $\min(\mathbf{x})$, the first quartile Q_1, the median, the third quartile Q_3, and $\max(\mathbf{x})$. The first and third quartile are represented by the box (see figure 3.5). Values larger than $Q_3 + 1.5(Q_3 - Q_1)$ and smaller than $Q_1 - 1.5(Q_3 - Q_1)$ are considered outliers (sometimes called $1.5 \cdot IQR$ rule) and are indicated as individual points. The whiskers extend from the box to the smallest and largest values

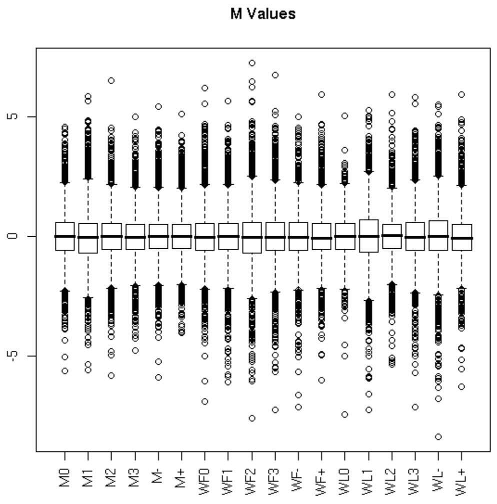


Figure 3.5.: Box plot of an expression matrix, containing 18 normalized samples of 2304 genes.

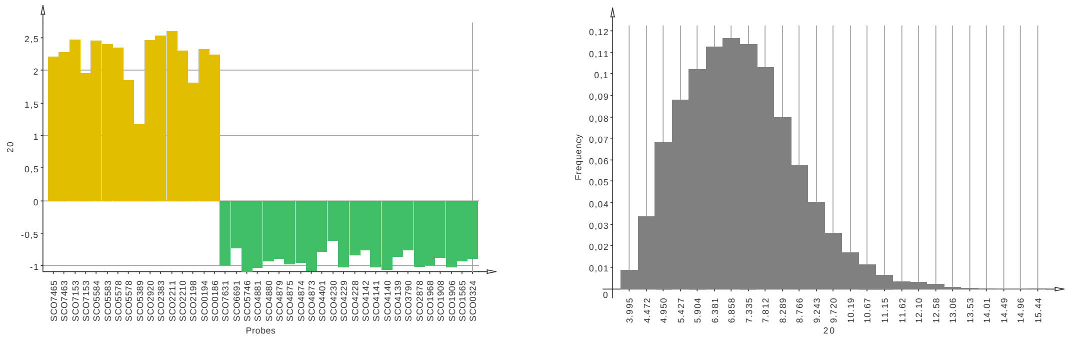
not being outliers by this rule. This allows to inspect the main characteristics of a data sample, as it gives a an overview of the location, deviation, skewness, and the number of outliers. For a $N(0,1)$- distributed values, based on the $1.5 \cdot IQR$ rule, the box contains 50% of the values, the upper and lower whisker in total represent 49.3% of the values, and 0.7 of the values are outliers.

Applications of box plots arise in all fields whenever single or multiple samples are to be inspected. In gene expression data normalization, it is often useful to compare the box plots of normalized and unnormalized values, or to inspect whether a number of normalized arrays have a similar distribution (see figure 3.5). In cluster analysis, it can be used to characterize clusters.

Variations of box plots are common: often, whiskers and outliers are omitted, in addition to the median the mean is indicated. Other outlier detection rules or summaries are also possible. *Violin plots* [76] extend the concept of box plots. A rotated and scaled kernel density plot is drawn in addition to indicators for the median and the whisker lines.

3.2.5. Bar Plots and Histograms

*Bar plot*s (see figure 3.6a) are a common chart type usually produced in order to visually compare a vector of numbers, presented as rectangles of heights (or widths) proportional to their value. In biological and other applications, bar plots showing means commonly have *error bars*, that show the standard deviation. A common variation are stacked bar plots, that allow to present fractions of a total within each bar.



(a) Bar plot

(b) Histogram

Figure 3.6.: Bar plot and histogram of gene expression data. (a) shows z-scored expression values at single time point of 40 genes. (b) shows the expression value distribution of a whole genome microarray in one sample.

Histograms are used to display frequency distributions. The input sample $\mathbf{x}$ is split into a number of (commonly, but not necessarily) equidistant bins. The sizes of the bins are then displayed as rectangles with their heights scaled to the bin size. The histogram can therefore be seen as a special case of the bar plot. The number and width of the bins are free parameters. Histograms show the full distribution instead of summaries, like the box plot and are therefore useful to identify multi modal distributions, or estimate shape, location, deviation and other properties of a distribution. *Stacked histograms* allow to compare multiple distributions. In stacked histograms, the rectangles representing the bins are stacked and scaled to reach a uniform height. *Kernel density plots* are used for the same purpose, but they give a continuous estimation of the distribution. They also allow comparisons of distributions, as several so-called density plots can be drawn in the same visualizations. Investigating histograms and density plots is often done in gene expression data analysis in order to check if distributions of values match expectations or requirements, and to detect unexpected peaks that might indicate quality issues.

3.2.6. Sequence Logos

Sequence logos were introduced in 1990 by Schneider *et al.* [159] as a tool to visualize consensus sequences. They are mainly used for visualization of motifs in nucleic acid or protein sequences. Sequence logos are created from multiple sequence alignments. Let A be an alphabet of symbols, for example nucleotide bases or amino acids. Let $f_{b,i}$ be the frequency of symbol $b \in A$ at position i of the alignment, then the *entropy* H_i is given as

$$H_i = -\sum_{b \in A} f_{b,i} \log_2(f_{b,i}) \tag{3.1}$$

The *information content* R_i of position i of the alignment is given as

$$R_i = \log_2(|A|) - (H_i + e_n) \tag{3.2}$$

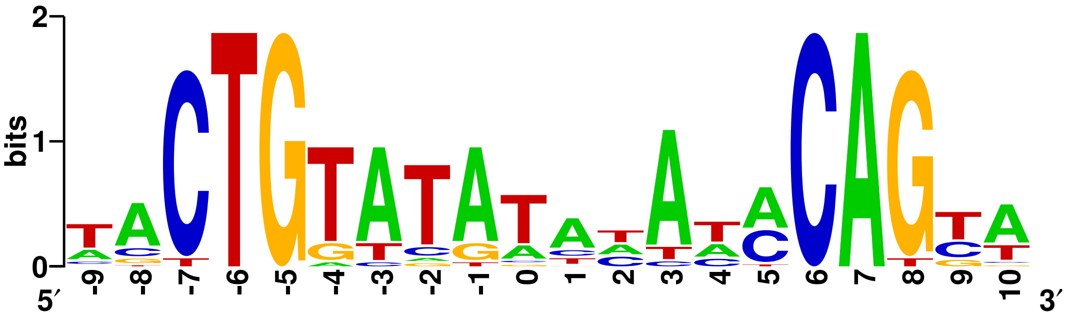


Figure 3.7.: Example of a sequence logo for a DNA sequence, created by WebLogo [38]. It shows an alignment of 19 DNA binding sites of the LexA repressor in *E. coli*.

R_i is measured in bits; e_n is a correction factor for small alignments of n sequences [160]; given as

$$e_n = \frac{|A| - 1}{2 \cdot \ln(2) \cdot n} \tag{3.3}$$

R_i is maximal ($R_i = \log_2(|A|)$) for a uniform position, which has zero entropy. The minimal value is reached when each symbol b has a frequency $f(b,i) = \frac{1}{|A|}$. Then, $R_i = 0$. The height $h_{b,i}$ of a symbol b at position i is given as

$$h_{b,i} = f_{b,i} \cdot R_i \tag{3.4}$$

A column of a sequence logo is then plotted by drawing each symbol b with height $h_{b,i}$ and unit width in a stacked way, in increasing order of height. Symbols are commonly drawn as the corresponding letter scaled in height. Each column in a sequence logo represents a position i in the multiple alignment (see figure 3.7 for an example). A sequence logo can be seen as a stacked histogram scaled by the information content of the underlying distribution.

Sequence logos are rich in information [159]. They show a summary of the entire alignment, with the consensus sequence highlighted. Each column shows the information content of the corresponding alignment position, and the ordered frequencies of the symbols.

3.2.7. Radial Plots

Several radial plots are used to visualize data. *Pie charts* are used to visualize a set of fractions. Each fraction is represented by circular sectors, with an arc length proportional to the fraction. Pie charts have, despite their ubiquity, been criticized. User studies found that it is complicated to compare quantities in pie charts [205]. As an alternative, bar charts can be used, which convey quantities by height, which is considered more efficient. Another alternative are *radial bar plots.* Radial bar plot (also called *polar area diagrams*) display values by circular sectors of equal angles. The sectors have a radius proportional to the values they represent. Radial bar plots can be seen as a bar plot transformed to polar coordinates.

Star plots, or *radar charts* are a variant of parallel coordinate plots transformed to polar coordinates. For a point $p \in \mathbb{R}^n$, the i-th axis is drawn at an angle of $\frac{1}{n}2\pi i$. Each p_i is then represented by a vertex of a closed, irregular polygon, placed on a position on the axis proportional to its value. The applicability of star plots depends on the data to be presented. Time series data or other data with defined order and meaningful differences between dimensions should be used.

3.3. Graphs and Networks

A powerful formalism for representing connected objects are *graphs*. The objects are called *nodes* or *vertices*, and the connections are called *edges*. In general, a graph is denoted as a pair of sets, i.e. the set of nodes, usually named V, and the set of edges, E, thus $G = (V, E)$. Edges connect two nodes, and are often denoted as an unordered set $e = \{v_1, v_2\}$. Two nodes u and v are called *adjacent* if $\{u, v\} \in E$. A graph with unordered edges is called *undirected*. A *directed* graph (*digraph*) has a set of directed edges, denoted by ordered pairs (u, v). Here, adjacency is an asymmetric relationship. For every node, the number of edges adjacent to this node is called the *degree* of this node, $deg(v)$, for digraphs, the *in-degree* $deg_i(v)$ and *out-degree* $deg_o(v)$ represent the number of incoming and outgoing edges. Edges can be weighted, let $w(e), e \in E; w(e) \in \mathbb{R}$ denote the weight.

Graph theory is an important discipline of mathematics and computer science, and has seen many theoretical advances since Euler's work on whether or not there is a path using all the seven bridges of Königsberg exactly once (which was not) [143]. Graph theory has also many theoretical and practical applications. Shortest path problems are applied in many fields, including navigation and optimization problems. In bioinformatics applications, graph representations are used for many problems, including genome assembly (De Bruijn graphs, spanning trees), phylogeny (inferring trees or networks), and representing molecular structures. In the analysis of biological networks, clique problems, modularity and centrality problems are investigated, with the purpose of identifying key nodes and subgraphs. In the analysis of biochemical pathways and reactions, maximum flow problems are of interest.

3.3.1. Selected Problems and Algoritms

In this section, some central aspects of working with graphs are summarized, however only topics relevant for this thesis are mentioned.

Common data structures for representing graphs are usually *adjacency matrices* and *adjacency lists*. An adjacency matrix is a square matrix denoting at position i, j the existence or weight of an edge between v_i, v_j. An adjacency list stores for each node v the nodes adjacent to v. The choice of data structures can influence the complexity of basic operations [143]. Space efficiency of the data structures depends on the number of edges in a graph. For sparse graphs, adjacency lists are more effective, for dense graph adjacency matrices are optimal, the exact turnover points depend on the complexity of the edges.

The foundation of many algorithms and applications in graph are methods for traversing and searching in graphs. Two basic principles exist: *depth-first search* (DFS) and *breadth-first search* (BFS, see [143, 68] for an overview). Input for both strategies is an input graph $G = (V, E)$ and a target node v_t. Both method traverse the graph in a tree-like manner, however, the strategy of building this tree differ, which is also reflected by the choice of supporting data structures.

Depth first search begins at some root node r. For each node v encountered, an expand step is performed, that pushes all previously unvisited neighbors of v on a stack. The topmost node on the stack is then investigated using DFS, unless v_t is found or the stack is empty. In summary, DFS works by going as deep as possible from a single node, before exploring the other neighbors of this node.

Breadth first search works in a similar way. However the entire neighborhood of a node is explored before going deeper into the search tree. Nodes are expanded by placing unvisited neighbors in a queue; the nodes are then inspected and expanded in the order of the queue, unless v_t is found or the queue is empty.

The worst case time complexity of both methods is $O(|V| + |E|)$, in the case of a single, unrepeated traversal of the graph, as every node and edge needs to be inspected. The space complexity is in linear in the number of nodes inspected, as they need to be placed in the stack or queue [143].

A *spanning tree* of an undirected graph $G = (V, E)$ is a tree, that contains all nodes and a subset of the edges of G, i.e. $T = (V, E' \subseteq E)$. If G is unconnected or directed, a *spanning forest* may be calculated. DFS and BFS can be used to find spanning trees in a graph, by adding every edge to E' that is connected to a previously unvisited node during the expansion step in the search. These spanning trees are not subject to any optimization. For weighted graphs, minimal spanning trees are often required, for which the total weight of the edges in E' is minimal. Algorithms for calculating *minimal spanning trees* are Kruskal's algorithm and Prim's algorithm [143].

For a digraph $G = (V, E)$, which contains no cycles, an ordering of the nodes $v_1 \prec \ldots \prec v_n$ can be given, that for every edge (u, v) $u \prec v$ holds. This is called a *topological ordering* [37]. Calculating a topological ordering (using Kahn's algorithm) involves first identifying source nodes, i.e. nodes v with $deg_i(v) = 0$. These nodes are then removed from the graph, and inserted into a sorted list T. The adjacent edges are then removed from G. Nodes which have no more incoming edges are added to T and, the process is iteratively on the resulting graph. If no more edges can be removed, and the $E = \emptyset$, T contains a topological ordering of V. Otherwise, the graph contains a cycle and no topological ordering can be established. The runtime of this algorithm is $(O(|V| + |E|)$, as every node and edge needs to be visited [37].

A problem often occurring is the question of whether or not a graph consists of a single component, or whether there are several unconnected parts of a graph. Let $G = (V, E)$ be an undirected graph, and $u, v \in V$. Then u and v are called *connected*, if there is path $u, \ldots, v$ in G. A *path* is an ordered sequence of nodes, of which successive nodes must be adjacent in G. If a path starts and begins at the same node, i.e. $u, \ldots, u$, it is called a *cycle*. The same definitions apply for directed graphs, based on the directed formulation of adjacency [68].

A graph G is called *connected*, if a path exists for every pair of nodes u, v. If a graph is not connected, two or more *connected components*, $C_i \subset V$ exist, which are connected or consist of single nodes.

The definition is extended to digraphs in two ways: a digraph is *weakly connected*, if an undirected path (i.e. with all edges considered to be undirected) exists for each pair of nodes. A graph is called *strongly connected*, if a directed path exists between each pair of nodes. The definition of weakly and strongly connected components is analogous to connected components in undirected graphs, however, a singe node v must be adjacent to an edge (v, v) to form a strongly connected component. Note that each strongly connected component is a cycle.

Testing connectivity can be performed using a depth-first traversal of the graph, in which every node is counted on the first encounter. If the final number of counted nodes is equal to the total number of nodes, then the graph is (weakly) connected. Finding connected components is done based on the same principle, restarting the traversal at an unvisited node for each new component. Both algorithms run in linear time, i.e. $O(|V| + |E|)$[143]. In order to find strongly connected components, Tarjan's algorithm [182] which is also based on depth-first search can be used. The runtime complexity is also linear in the number of nodes and edges, $O(|V| + |E|)$.

Extending from connectivity, identifying clusters in graphs is an useful for investigating many kinds of networks in social, biological and computer networks research. As a target function for optimizing such clusterings, *modularity* has recently been introduced [139]. The modularity of a clustering represents the fraction of edges within groups, normalized by the expected fraction of such edges in a random graph. *Modularity clustering* was already successfully applied to bioinformatics studies [173]. Furthermore, it has been shown that modularity clustering and force-based graph layout share many properties [142].

3.3.2. Graph Drawing

Graph drawing is the process of visual representation of a graph. The most common scenario is an embedding of nodes and edges on the euclidean plane. Commonly, nodes are represented by geometrical shapes (spheres, boxes) and edges are drawn as lines, polylines or curves. No ultimate solution for the problem of graph drawing exists, instead, the adequate method depends on the type and structure of the graph and the application. There are several general aesthetic requirements. Most importantly, nodes should not overlap each other and ideally, edges should neither overlap nodes or other edges. The latter problem can only be solved optimally for planar graphs. Other requirements depend on the application, and may address distribution or size of nodes and length and shape of edges (see for example [146]).

Different methods are applied to this problem. *Force-based* graph layouts model nodes as charged particles and edges as springs connecting the particles (executing an attracting force). They can be applied to a wide range of graphs and often lead to good results.

Classical examples are the algorithms by Fruchterman and Reingold [57] and Kamada and Kawai [88]. In general, force-based algorithms use an energy term, which is to be minimized during layout. *Attractive forces* (springs) can be modeled using Hooke's law, the *repulsive forces* are modeled using Coulomb's law, or modifications of both methods. This is used to iteratively calculate the forces acting on nodes and moving the node according to it, in order to minimize the total energy of the graph. To avoid early convergence to suboptimal local minima, optimization strategies are used, like simulated annealing in the Fruchterman-Reingold algorithm.

Directed acyclic graphs can be laid out using a *layered method*, in which nodes are distributed into layers, the nodes in which are then iteratively reordered in order to minimize the number of edge crossings. The Sugiyama framework [48] encompasses the following steps: (1) *cycle removal* (2) *layer assignment*, (3) *vertex ordering* and (4) *positioning of nodes*. For the first step, using an algorithm that addresses the *feedback arc set* problem, edges inducing cycles found and then be inverted or removed, allowing to use hierarchical layout for graphs with cycles. This is required for (2), as the layering of a graph is only defined if this graph is acyclic. A layering of a graph is established by placing a node in a layer, so that all previous nodes are in a higher layer, and all following nodes are in a lower layer. Layer assignment algorithms are the longest path layout and the Coffman-Graham algorithm. The *vertex ordering* step arranges the nodes in each layer (3) so that the overall edge crossings are minimal. This problem is NP-complete [48]. Several heuristics are applied here. In the final step, the ordering of nodes in the layers is transformed to actual coordinates, adhering some constraints.

Based on a spanning tree of the graph, a tree layout method, e.g. the Reingold-Tilford algorithm can be used to embed a general graph. An example for a *radial layout* based on a spanning tree can be found in [211]. Grid-based methods strive to place nodes on positions in a grid, which allows to take the node size into account. Examples exist for the layout of biological networks [118, 106]. *Spectral graph drawing* uses matrix representations of graphs of which eigenvectors are employed. Large graphs can be swiftly laid out using this method, possible with respect to additional criteria. An adjacency matrix of the graph or transformations of it are used [108].

3.4. Visualization of Biological Pathways

Biological pathways arise in many fields of biology and are the main building blocks of our understanding of physiological processes. Textbook examples of *metabolic pathways* come from the carbohydrate metabolism: glycolysis (via the EMP pathway) and the TCA cycle are central parts of the metabolism in a wide range of organisms. Pathways can be seen as a set of reactions, which form a network, that might be linear, branched, circular or otherwise complex. *Signaling pathways* describe how cells react to internal or external signals. Usually, this encompasses a receptor molecule located on the cell membrane and a cascade of second messengers that induces cellular actions, for example by altering gene expression or inducing cellular or metabolic processes. Internal signals can also trigger many signaling pathways. Important examples are encompass major cellular events like apoptosis and cell differentiation.

The focus of research lies in identifying new pathways and finding extensions, shortcuts and new effects of pathways. Optimization of especially metabolic pathways are a tool of biotechnology for efficient production of compounds, e.g. antibiotics. Visualization of pathways is a useful tool for such activities.

An early application for a comprehensive visualization of biochemical pathways is the Roche pathway map (based on [132]). It provides a detailed view of common biochemical pathways, including chemical structures for most participants. To enhance search, it has a grid and an index book in which for every compound, the grid position is listed.

In computer generated visualizations of biological pathways, the natural representation of pathways is the graph, and many applications use this concept (see 6.2 for an overview of applications in this field). Requirements for visualization of pathways have been analyzed in user studies in [156]. These encompass easy *integration* of pathway data, *information overlay*, which means adding biological relevant information to the pathway graph and *enriching pathway graphs* with high-throughput data. Furthermore, *overviews* and *browsing* the pathway landscape have been requested. *Semantical scaling* and *aggregation* of pathways is required for maintaining overview in large pathway systems.

Due to specific aesthetic requirements of drawing biological pathways, common graph drawing algorithms can give suboptimal results. The existence of cellular compartments, requirements on proximity of certain nodes and conventions make optimal layout a complicated task. Specific graph layout methods have been developed for the presentation of biological pathways. An early approach was taken by Karp *et al.* [93], which used decomposition of a graph into easily drawable components. Several similar approaches and enhancements to this approach were suggested [14]. Alternatively, grid based layout algorithms have been employed, some of which allow to use biological properties, e.g. [106]. Many applications however rely on predefined pathway diagrams, and manual layout.

3.4.1. Data Formats

Biological pathways are available in a number of formats, which are briefly described here. All major data formats for biological pathways are based on XML, however they differ greatly in way they model the data. The major sources of biochemical pathways are *KEGG* [91] and a number of databases that provide *BioPax* [43] files, including MetaCyc [29] and others [130, 30, 144].

KGML

The KEGG Markup Language (KGML, `www.genome.jp/kegg/xml`) is a format for the definition of KEGG pathways. A KGML file provides all entries, i.e. chemical compounds, enzymes or map links for a single pathway. The entries are defined by links to the KEGG database. The graphical representation of the entry is defined in a graphics element, that also denotes the position and size of the object. *Reactions* describe a chemical reaction between substrate and product entries. Most objects have a name property, annotations are mostly stored as references to the KEGG database. The “type” attribute of a reaction denotes the reversibility of the reaction. Furthermore, *relations* define relationships between other entries. Relations can describe successive reaction steps, protein interactions, coexpression, and relationships to other pathways.

BioPax

The *BioPax* format is based on a systematic modeling of biological processes [43]. Several versions (“levels”) of BioPax exist. The current level 3 can describe metabolic and signaling pathways, regulatory networks, and molecular interactions. Level 2, which is still common in databases and applications, lack support of regulatory networks and is less descriptive for signaling pathways. BioPax is divided in an *ontology*, which is the abstract representation, and the BioPax file format. The file format is based on OWL (Web Ontology Language), an XML dialect which allows the description of ontologies and of documents based on the ontology (see BioPax level 3 user guide, `biopax.org`). The base class of the BioPax ontology is the *Entity*, It has several subclasses, including *Physical Entity*, which has in turn several subclasses that describe different types of chemicals. *Interaction* is the base class for several types of direct and indirect reactions, including protein interactions and chemical reactions and controlling processes (catalysis, inhibition). Pathways are described by listing the reactions in the pathway. The order of these reactions is indirectly defined by a network of *Pathway Step* entities.

Each BioPax entry is described with a terms taken from controlled vocabularies and enriched with a number of cross-references to databases. Evidence of the described biological fact and availability (legal or otherwise) must be stated. For interactions, the chemical or otherwise properties can be added. This makes BioPax files very rich in information.

Other Formats

The most common format for denoting biochemical reactions is *SBML* [81]. It integrates definitions of reactions, *species* (entities taking part in a reaction), *compartments*, and a mathematical description of the *reaction kinetics*, and the *initial conditions*. Commonly, objects in SBML are described using terms of the *Systems Biology Ontology* (SBO) [113]. The BioModels database [113] provides a curated source of SBML models. These models mostly describe reactions in great detail, and sometimes cover entire pathways, however, SBML is predominantly used to model smaller processes in higher detail.

CellML [39] has a purpose similar to SBML. It is designed as a language that describe models using differential equations. Biological objects are described using “components”, which can contain variables and mathematical definitions. Connections can be defined between components, with respect to variables. A repository of CellML models is available at `cellml.org`.

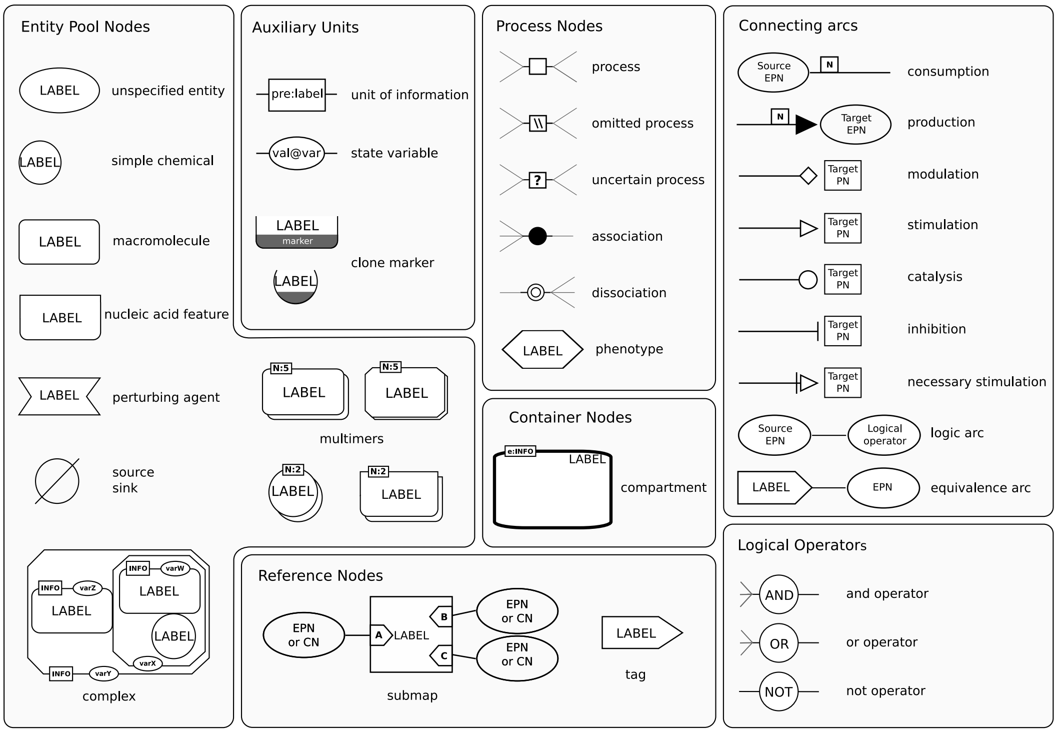


Figure 3.8.: Overview of the SBGN process description language (taken from `sbgn.org`, see also [135]).

The *PSI-MI* format is used to denote protein interactions [100]. PSI-MI uses lists of interactors, described using controlled vocabularies, and interactions, for which confidence values and experimental evidence can be stated. Both an XML and a tab-separated text file variant of this format are available. PSI-MI is used among others by the IntAct database (`www.ebi.ac.uk/intact`).

3.4.2. SBGN

While a vast number of biological pathways, models, and interaction networks have been published, no unique standard for depicting physical and conceptual components has been established for a long time. It has been argued that biology lacks a standard graphical notations with defined meaning of symbols, like for example in electronics symbols. This has lead to a situation, in which the same symbol can denote many different concepts, depending on the publication, or vice versa, a single concept is denoted by various symbols.

To overcome this problem, *SBGN* – the Systems Biology Graphical Notation – was introduced in 2009 by Le Novère *et al.* [115]. SBGN was the result of an international community effort of systems biology researchers. Some of the contributors to SBGN work

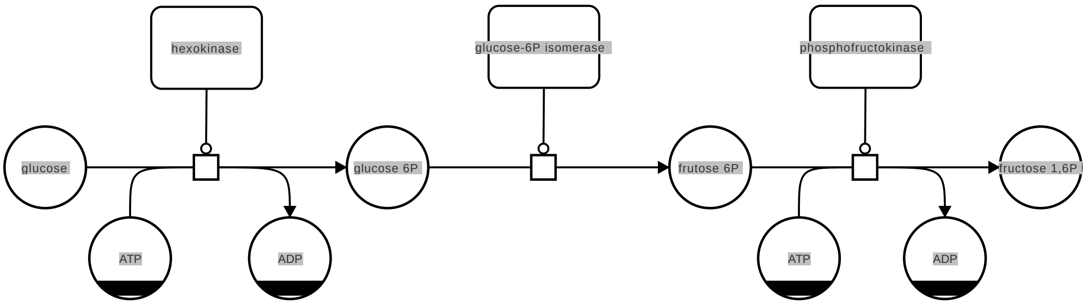


Figure 3.9.: First steps of the glycolysis in the SBGN Process description language. Adapted from [135].

on SBML and other file formats, which ensured compatibility of the underlying concepts. The aim of SBGN was to provide a free, consistent notation, for describing biological objects and interactions. The set of symbols used was to be minimal and visually discriminable. For this purpose. in SBGN, color and size of objects carry no information, which allows allow photocopying and easier drawing by hand. Furthermore, modularity and a mathematical model were required [115]. Previous work on the unification of biological diagrams was done by Kohn *et al.* [105], who introduced *Molecular Interaction Maps*, which already defined a set of symbols and possible connections of these symbols. This was taken up by other researchers, most of whom contributed to SBGN. SBGN addresses some of the information overlay requirements noted above.

Biological processes are diverse and can be investigated on multiple levels. Therefore, SBGN provides three different languages, which focus on different aspects of biological processes. Each language defines a set of *glyphs*, which unambiguously describe an object, and arcs which connect nodes that are represented by glyphs. All languages define rules that direct which nodes are allows to be connected by which arcs. Nodes and edges form a graph, the layout of which is not defined by SBGN. However, specifications exist on node and arc placement to ensure readability.

The SBGN *process description language* (see [135] for the complete specifications) is designed for sequential descriptions of biochemical processes. A process is represented by a process node (see figure 3.8), the participating biological entities are described as entity pool nodes (see figure 3.9 for an example). *Entity pool nodes* describe indistinguishable biological entities, like macromolecules (proteins, etc), simple chemicals, nucleic acid features (DNA or RNA molecules), complexes or multimers of these types. Perturbing agents can also be defined. All elements are defined based on SBO terms. Annotations on objects can be added using *units of information* that can be used to describe certain aspects of an entity pool node, based on SBO terms. The state of an entity, e.g. modifications of proteins is displayed using *state variables. Clone markers* indicate the an entity pool node occurs several times in a diagram, which is not allowed otherwise. Further elements are cellular compartments, and references nodes, i.e. labels and submaps that allow the compressed representation of nested processes. Logical operators allow to model dependencies and alternatives of entities in a process. SBGN process descriptions correspond to textbook pathway diagrams.

Interactions between entities can be displayed in the SBGN *entity relationship language* [115]. This language uses unique entities, which are connected by arcs representing influences. Influence arcs can be connected to other influence arcs, thus create possibly complex networks of relationships. State variables are used to describe internal changes, like allosteric modifications. The main purpose of entity relationship diagrams is describing signaling and other processed that involve many state changes of entities.

SBGN *activity flow* diagrams are used to describe the sequential influences between entities. These diagrams display the influence, but omit the biochemical details of the processes. To this purpose, entities, perturbations and phenotypes are displayed, arcs representing inhibitory, activating or other influences connect these entities. Entities can be annotated with units of information displayed as entity pool nodes to indicate the type of the entity. The style of activity flow diagrams is similar to many informal drawings of signaling pathways.

The three languages of SBGN are an important step towards the formalization of biology. Especially the process description language is widely adopted and is supported already by several tools and databases. In this work, the SBGN process description language is used for formal representation of biological processes.

3.5. Mayday

Mayday [12] is a tool for the integrative analysis of gene expression data. It is designed to analyze and visualize microarray and similar systems biology data. Mayday is platform independent and implemented in Java. The development of Mayday at the Integrative Transcriptomics group at the University of Tübingen began in 2003 as a tool for microarray data visualization and clustering. Many extensions have been added since then, with major code revisions performed in 2005 and 2008. A group of five core developers and 18 contributors collaborated on Mayday over the years. In a recent review of numerous microarray data analysis tools, Mayday was found to be among the most complete and usable of 78 tools investigated [109].

3.5.1. Data Model

The *data model* of Mayday consists of several classes. All information concerning an expression study are contained in a `DataSet` object. The main component is an expression matrix, called `MasterTable` in Mayday. A single measured object, e.g. gene or protein, is modeled by the class `Probe`. The probe stores the expression values and therefore represents a row of an expression matrix. A sample is represented by an `Experiment`. In addition to the primary expression data, *meta information* about dataset, experiments and probes ca be stored. These objects are called *MIOs* – Meta Information Objects – and are organized hierarchically in `MIGroup` objects. MIGroups keep a map of annotated objects to MIOs, of which a number of different types exist. A major concept of data organization in Mayday are *probe lists*. The class `ProbeList` keeps an unordered set

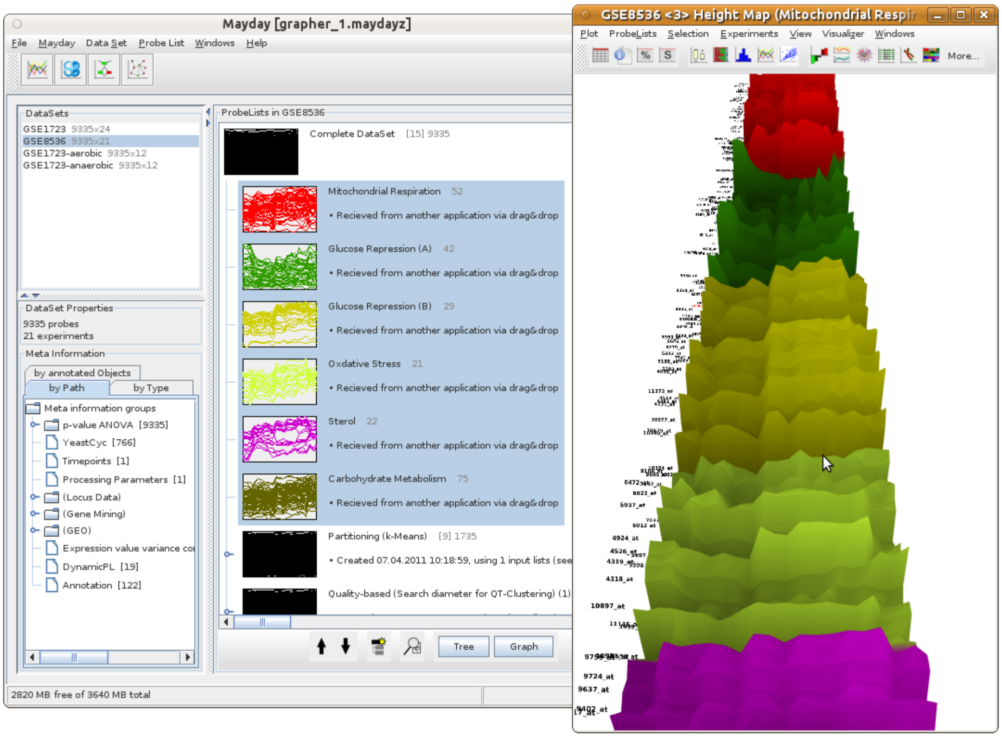

Mayday [grapher_1.maydayz]
File
Mayday
Data Set
Probe List
Windows
Help
DataSets
GSE1723 9335×24
GSE8536 9335×21
GSE1723-aerobic 9335×12
GSE1723-anaerobic 9335×12
DataSet Properties
9335 probes
21 experiments
Meta Information
by annotated Objects
by Path
by Type
Meta information groups
p-value ANOVA [9335]
YeastCyc [766]
Timepoints [1]
Processing Parameters [1]
(Locus Data)
(Gene Mining)
(GEO)
Expression value variance co
DynamicPL [19]
Annotation [122]
2820 MB free of 3640 MB total
ProbeLists in GSE8536
Complete DataSet [15] 9335
Mitochondrial Respiration 52
• Recieved from another application via drag&drop
Glucose Repression (A) 42
• Recieved from another application via drag&drop
Glucose Repression (B) 29
• Recieved from another application via drag&drop
Oxdative Stress 21
• Recieved from another application via drag&drop
Sterol 22
• Recieved from another application via drag&drop
Carbohydrate Metabolism 75
• Recieved from another application via drag&drop
Partitioning (k-Means) [9] 1735
• Created 07.04.2011 10:18:59, using 1 input lists (see
Quality-based (Search diameter for QT-Clustering) (1)
Tree
Graph
GSE8536 <3> Height Map (Mitochondrial Respir
Plot
ProbeLists
Selection
Experiments
View
Visualizer
Windows
More...

Figure 3.10.: Main window of Mayday, showing different datasets studying growth of *Saccharomyces cerevisiae* in different conditions. The selected probe lists are displayed in the height map visualization on the right.

of probes. Probe lists, which can also form hierarchies are the main objects for user interaction in Mayday. The main view of Mayday shows the probe lists defined in a dataset (see figure 3.10). Probe lists are input and result of most data analysis features and visualization tools in Mayday.

The *Mayday core* provides the framework for statistics and graph handling, as well as genomic scale data structures. For extending Mayday, a generic *plugin interface* exists. Many functions, on all levels are implemented as plugins. The plugin interface uses the capability of the Java Virtual Machine to load new classes at runtime. Predefined *extension points* exist for plugins that work on probe lists, datasets and meta information, for data visualization, for distance measures and data import. Plugin packages can define their own extension points if necessary.

3.5.2. Extensions to Mayday

Mayday features packages for many analysis and visualization purposes. Partitioning clustering is implemented in various different algorithms, including k-Means [121], quality threshold clustering [75], self organizing maps and others. Hierarchical clustering is possible using UPGMA and neighbor joining. Several tools for assessing clustering

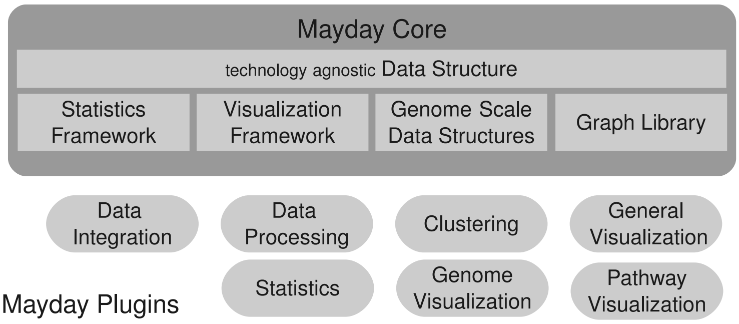


Figure 3.11.: Overview of the Architecture of Mayday. The Mayday core provides data structures and basic frameworks. Independent plugins provide the visualization and analytical functionality for the user.

quality are available. The training and testing of classification models and the inference of association rules is possible via a series of extensions based on the WEKA machine learning library [209]. Several standard and non-standard statistical tests and feature selection methods are implemented, including the t-test, SAM, WAD and Rank Product. Gene set enrichment analysis tools, including ORA (over-representation analysis) and methods for analyzing enrichment of overlapping gene sets are implemented. For data processing and organization, a modular pipeline system is available. Intelligent gene selections called *dynamic Probe Lists* help to filter data dynamically. Using an interface based on the Derby database management system (`db.apache.org/derby`), SQL queries can be performed directly on Mayday's data structures. Access to these data structures is available to programmers via interactive environments for execution of R and JavaScript code in Mayday. The alignment of time series allows to identify time shifts between different time series. For the pairwise and multiple alignment of time series, Mayday *Tiala* - "Timeseries Alginment Analysis" provides visual and analytical tools. A recent major extension is Mayday *SeaSight* [11]. SeaSight allows to import, structure and normalize RNA-Seq data and microarray data from multiple sources, and provides methods for combining data from both technologies.

3.5.3. Visualization Framework

Visualization is a main focus of Mayday. Basic visualization of gene expression data is performed using all the tools described in section 3.2, especially profile plots, heat maps, scatterplots (in several variants), and box plots. These plots feature extensions for basic visualization of meta information, including an enhanced heat map [63]. Three dimensional visualization plots have been recently added, including 3D scatterplots and PCA plots [86]. Visualization of high dimensional data is possible using height maps (see figure 3.10) and radial profile plots, a 3D-extension of star plots. *ChromeTracks* is an extensible, track based genome browser, that allows to visualize expression data in its genomic context [214]. Tracks and other components can be added via plugins. The plots are supported by table views of expression and meta data.

The *visualization framework* in Mayday provides several key feature for all plots. This is based on a uniform and user friendly interface, which allows adding and removing probe lists from the plot, and to create new visualization of the current data. A window list makes handling of multiple views easier. Selections and data transformations are communicated between the currently open plots. Selections can be used to create new probe lists. Plots can be exported in a number of bitmap formats, as well as SVG and PDF for publication quality figures.

The Mayday visualization framework is organized in a *model/view pattern*. The model component is the `ViewModel`, which keeps the probe lists to be visualized, manages the probe selections and notifies plots of changes to these objects. The view model allows the unified access to the data for all plots and keeps a reference to a `ProbeDataManipulator` that performs *online data transformations*. All visualization tools are implemented as Java Swing components, and implement the interface `PlotComponent`. They are child components of a `PlotContainer`, which provides the overall user interface, including the menu, over which the most functionality described above can be reached. The conceptual structure of *multiple connected plots* is the `Visualizer`, which keeps track of all plots for a single view model. Plots are generated via specialized plugins that return a `PlotComponent` instance. For easier access to probe data, the `ValueProvider` class, which also provides a user interface shown in the `PlotContainer`, can be used, the `ColorProver` integrates a configurable `ColorGradient` and transforms probe values into corresponding colors. The actual implementation of a plot can use these components, and can otherwise use any Java component necessary, e.g from an internal chart library, Java Swing and JOGL, an OpenGL framework for Java.

While Mayday provides many useful visualization and analysis tools, some features necessary for working with modern high throughput gene expression data still need to be provided. Based on the framework Mayday provides, many of the necessary tools can be integrated. Visualization specialized for meta information biological models are introduced in the course of this thesis.

4. Analysis and Visualization of Resequencing Microarrays

The analysis of gene expression on transcriptome and proteome level gives important insights into the expression strength of a protein. However, it is not guaranteed that a gene product has the expected function. Single nucleotide polymorphisms (SNPs) can occur in genes, which do not necessarily effect the expression level. SNPs however may introduce changes in the protein structure, which could lead to loss or decrease of function. To identify such effects, it is necessary analyze at the genomic sequence.

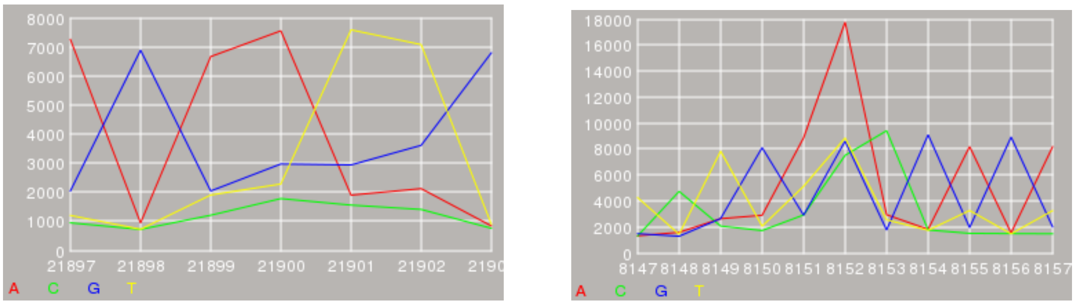
Resequencing microarrays are a useful tool for small-scale sequencing. The main application is the determination of individual sequence deviations from a reference sequence. The major field of applications is in medical diagnostics and research. For these purposes, based on the diploma thesis of Kirstin Weber [203], the application ResqMi was developed. ResqMi addresses common issues related with the most prominent technology of resequencing microarrays, the Affymetrix GeneChip® CustomSeq® microarrays. ResqMi features an efficient base calling algorithm that uses a model-based approach using intensity comparisons and region-wise conformance assessment, as well as an algorithm to revise uncalled positions.

This chapter summarizes my work on resequencing microarrays. First, an introduction into the technology of resequencing microarrays is given. Then ResqMi and its features for base calling and visualization are introduced. To demonstrate these features, ResqMi is applied for quality assessment and base calling. This chapter is concluded by a discussion, which also addresses upcoming technologies in this field.

4.1. Analysis of Resequencing Data

Resequencing microarrays are commonly used for the fast and precise analysis of individual genetic variations. An important application is the identification of *genetic diseases* by resequencing the respective genes. This allows a fast and reliable diagnosis and often directly displays the cause of the disease. Applications include monitoring genetic variation of infectious diseases [176], identifying infectious pathogens [119, 125, 199] and antibiotic resistances [42]. Affymetrix's Human Mitochondrial Resequencing Array 2.0 [124] which interrogates the entire 16kb mitochondrial genome on a single array has been used for the detection and diagnosis of various diseases, such as oxidative phosphorylation disorders and [195], and cancer [133].

Resequencing microarrays are used to determine sequences by hybridization. In contrast to classical sequencing by hybridization, resequencing designs use a known reference sequence. This drastically reduces the number of probes required sequencing. For each



(a) Region with high quality data

(b) Mismatch site: the central position in the chart represents a mutated base

Figure 4.1.: Intensity charts of a resequencing microarray (a) shows positions with high quality: a high signal to noise ratio is observed at all positions in the chart. (b) shows a mismatch site: at the central position, a highly intense signal is observed, corresponding to a base not matching the reference, the neighboring sites have comparably low intensities.

position in the reference sequence, eight probes, typically of length 25 are used. Four probes interrogate the sense strand, the other four are complementary and query the antisense strand. Affymetrix produces arrays that contain for each perfect match probe the three respective mismatch probes. In total, for determining a sequence of length n, $8n$ probes are necessary. As target material, amplified genomic DNA of the genomic region of interest is used.

The assumption used in this array design is that the single central mismatch position is enough to assure that no excessive unspecific hybridization occurs for the probe. In general, every sequence position is assumed to have probes with a high signal to noise ratio. However, non-central mismatches also reduce the signal at a probe, e.g. near a base that differs from the reference. Figure 4.1 show examples of a site with high quality signals, and a site with a mismatch, with respect to the reference. For positions with low intensities and low signal to noise ratio, no reliable signal for calculating the sequence is available, and results into uncalled positions. Instead of `A`, `C`, `G`, or `T`, the sequence contains an `N` at the relevant position.

This design of resequencing arrays allows to identify substitutions, but other mutations like deletions and insertions cannot be detected. While it is possible to add probes for a limited number of events, it is infeasible to add probes that cover all possible events (see [69] for details). Instead, the Affymetrix arrays often query commonly known mutations using a set of probes based on a reference sequence that reflects the mutation.

4.1.1. Base Calling

The analysis of genetic variations using resequencing microarrays involves laboratory protocols similar to those used for expression microarrays (see section 2.2.4). After scanning, however, the analysis of resequencing microarrays is different. Normalization steps,

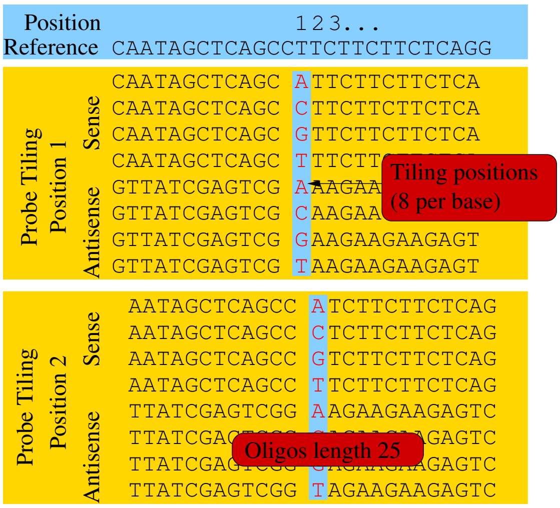

Position
123...
Reference CAATAGCTCAGCCTTCTTCTTCTCAGG
Probe Tiling Position 1
Sense
Antisense
CAATAGCTCAGC ATTCTTCTTCTCA
CAATAGCTCAGC CTTCTTCTTCTCA
CAATAGCTCAGC GTTCTTCTTCTCA
CAATAGCTCAGC TTTCTT
GTTATCGAGTCG AAAGAA
GTTATCGAGTCG CAAGAA
GTTATCGAGTCG GAAGAAGAAGAGT
GTTATCGAGTCG TAAGAAGAAGAGT
Tiling positions (8 per base)
Probe Tiling Position 2
Sense
Antisense
AATAGCTCAGCC ATCTTCTTCTCAG
AATAGCTCAGCC CTCTTCTTCTCAG
AATAGCTCAGCC GTCTTCTTCTCAG
AATAGCTCAGCC TTCTTCTTCTCAG
TTATCGAGTCGG AAGAAGAAGAGTC
TTATCGA GAGTC
TTATCGA GAGTC
TTATCGAGTCGG TAGAAGAAGAGTC
Oligos length 25

Figure 4.2.: Probes on a resequencing microarray. The figure shows the probes for two positions on a resequencing microarray.

apart from background correction, are usually not performed. The crucial analysis step is *base calling*.

Base calling is the process of converting the intensities of probes representing the different alternative bases into a DNA sequence. More formally, let $I^s_{x,i}$ denote the intensity of base $x \in \{A, C, G, T\}$ in tiling position i on the sense strand, and let $I^a_{x,i}$ be the corresponding intensity for the antisense strand. Let R_i be the base and R^c_i the complementary base at the ith position of the reference DNA. Then, a base calling algorithm is a function $f : I^{\{s,a\}}_{x,i} \mapsto A, C, G, T$. This function must be applied to every position of the sequence. Often, base calling algorithms use additional positions or quality measurements for determining the sequence. A naïve base calling algorithm would simply call $B^s_i = \arg\max_x \left\{ I^s_{x,i} \right\}$. However this would lead to unreliable calls when the data is of poor quality. Several more sophisticated algorithms for base calling exist. The most important one, ABACUS ("Adaptive Background genotype Calling Scheme"), has been published by Cutler *et al.* in 2001 [41], and was used in the analysis tools provided by Affymetrix. ABACUS performs data quality control in order to identify positions of poor quality. Then, for each possible base, the likelihood of a specific model is calculated. The base with the highest likelihood is then called [41]. An alternative base calling algorithm, Model-P, was suggested by Zhan *et al.*. It employs a physical model of DNA hybridization which is based on the target sequence [212]. A fast and efficient algorithm was published by Clark *et al.*, which is based on a statistical model [34].

Clark's algorithm was used to find SNPs in a whole genome resequencing project in *Arabidopsis thaliana* [34]. The application to single-gene resequencing for base calling is straightforward. The algorithm works in general by using naïve base calls, unless the quality of data representing the position in question is poor (see figure 4.3 for an overview

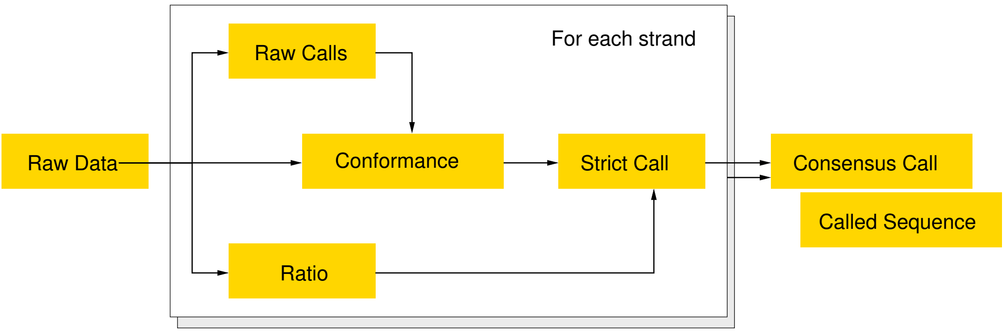


Figure 4.3.: Overview of Clark's base calling algorithm. The figure shows the elements calculated during the algorithm for base calling of a single position. The input (raw data) are the intensities for both strands. See text for details.

of the algorithm). For each position i, calculate separately for both sense and antisense strands:

1. The *raw base call*: $B_i = \arg\max_x \{I_{x,i}\}$
2. The *conformance*, which is a measurement of how much the raw base calls of neighboring positions around i correspond to the reference sequence R. C_i is defined as the fraction of bases for which $B_j = R_j$ is true for all

   $$j \in \begin{cases} [i-10, i+10], & \text{if } B_i = R_i \\ [i-20, i-10] \cup [i+10, i+20], & \text{else} \end{cases}$$

   holds. The ranges on which C_i is calculated are based on the rationale that for a non-reference call the neighboring bases have reduced intensities.
3. The *signal to noise ratio* is calculated from P_i, the highest intensity for base i and Q_i, the second highest intensity, as $\Delta_i = \frac{P_i}{Q_i}$.
4. The *strict call* is calculated using cutoffs μ, ν for C_i and Δ_i:

   $$S_i = \begin{cases} B_i, & \text{if } C_i > \mu \text{ and } \Delta_i > \nu \\ \text{N}, & \text{else} \end{cases}$$

   If S_i is an alternative call, i.e. $S_i \neq R_i$, and there is an alternative call within the range $i-5, \ldots, i+5$, with a higher maximal intensity, the call as position i is considered unreliable, and $S_i = \text{N}$ is called.

Based on the strict base calls for both strands S_i^s and S_i^a, the *consensus call* is calculated. If both calls are complementary, and no other alternative call within the range $i-5, \ldots, i+5$ with a higher maximal intensity exists, the corresponding base is called. If both calls differ, either `N` is called (*strict consensus call*), or the IUPAC nucleotide symbol representing both bases is called (*relaxed consensus call*). Clark *et al.* used the strict consensus calls for finding SNPs by integrating sequences from different strains of

A. thaliana. For base calling purposes alone, both versions of the consensus calls can be used.

This procedure aims at identifying positions with poor quality. These are ambiguous positions which have a low signal to noise ratio Δ, and positions within a region greatly differing from the reference sequence, which show a low conformance C. This is based on the rationale that for an alternative call the intensities at the adjacent positions are reduced, as all the probe targets at these positions have a mismatch compared to their probes. A large part of the intensities for these positions is due to unspecific hybridization [69], which is the reason for rejecting alternative calls near high-intensity alternative calls. The highest intensity alternative is more likely to be actually based on a reliable signal.

All base calling algorithms used for resequencing microarrays have in common that some ambiguous calls remain, so that manual inspection of data is required [195]. No-call ratios of about 5% [1] leave several hundred bases per experiment to inspect by the user in order not to miss an important mutation. Manual inspection and subsequent editing of such large datasets is generally cumbersome and time consuming. Furthermore, GSEQ (GeneChip® Sequence Analysis Software, Affymetrix), the only commercially available software application for Affymetrix resequencing arrays, lacks important visualization features, base editing ability and has restrictive operating system requirements.

Efficient and fast processing of the arrays however is important. In particular, base calling should be fast and reliable. Additionally, user-friendly interaction as well as swift navigation through sequence and intensity data is necessary to find and identify the impact of mutations. Finally, software should provide an overview and position specific visualization of intensity as well as sequence data, which is important for the inspection of critical positions and for manual base calls. All these considerations lead to the development of ResqMi, an analysis software for resequencing microarrays, that is described in the next section.

4.2. ResqMi

ResqMi, short for "*Reseq*uencing using *Mi*croarrays" is a multi-platform, open source software for the analysis of resequencing microarray data. It features visualization of intensity and sequence data, calling algorithms and tools to revise uncalled bases. ResqMi has a user-friendly GUI and can easily be expanded with further functionality via a plugin interface (based on the Qt plugin mechanism).

The development of ResqMi began in 2005, during the diploma thesis of Kirstin Weber [203]. This basic version of ResqMi has been rewritten and extended in this thesis. ResqMi is implemented in C++ using the Qt framework (`qt.nokia.com`) for GUI development. Currently, the project consists of the main application, a shared library providing common components and several small plugin libraries. In total, the current version ResqMi 1.2 encompasses 37,500 lines of code.

ResqMi works on data generated with the Affymetrix's Sequence Analysis platform. It processes raw intensity data (stored in "CEL" files) and processed sequence data (from

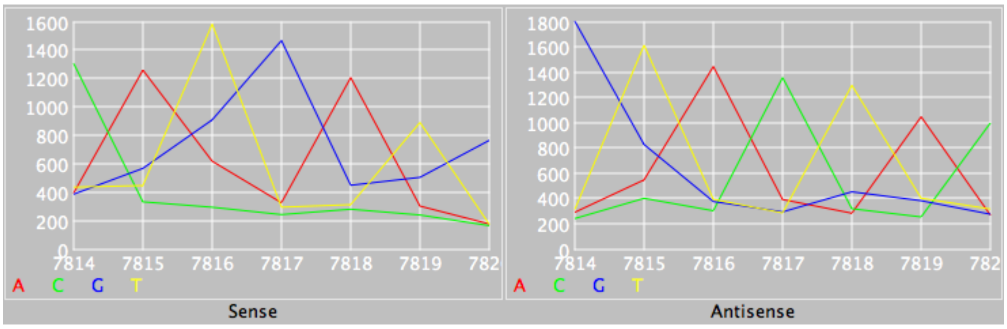
"CHP" files). Called sequences are either provided by external applications or produced by base calling performed within ResqMi. The data is organized in a data tree, which shows CEL and CHP files separately and is the main control panel of ResqMi.

Intensity data can be analyzed using ResqMi. For this purpose, ResqMi features Clark's model based calling algorithm described above. Furthermore, it is possible to inspect the data on several levels. The intensity values can be shown as a list with a connected chart component. The chart component displays either a line chart, spike plots (which resemble chromatograms from Sanger sequencing), sequence logos (see section 3.2.6), or a table view of the selected base and its neighbors. Figure 4.4 shows several examples. The plots can be exported to several image formats. A single intensity window is used for each CEL file; to compare several files reduced views without tables can be used.

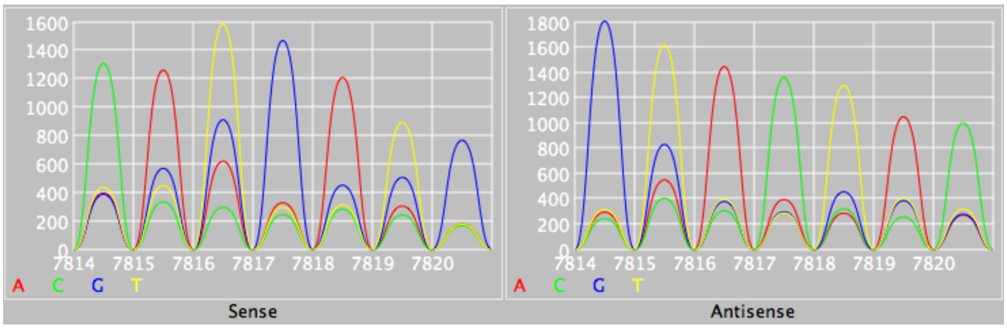
For *quality control* in gene expression microarrays it is common to inspect image plots of foreground and background intensities in order to detect spatial effects. These tools can also be used for resequencing arrays. For this purpose, ResqMi allows to display image maps of probe intensities, standard deviation and coefficient of variation values extracted from CEL files. Furthermore, the base with the highest intensity for each sequence position can be plotted. For comparative purposes, given a reference array R and a target array T, MA-transformed values can be inspected. These values are calculated as $M = \log_2(R) - \log_2(T)$ and $A = \frac{1}{2}(\log_2(R) + \log_2(T))$. Different color gradients can be used, including heat colors, grayscale and green-red-black. Images can be zoomed in, on the highest zoom level, the letter of the base queried by each probe is displayed; export in various formats is also possible.

Sequence data in CHP files can be viewed and edited in the "Resequencing Window", the central window of ResqMi (see figure 4.6). It gives a fragment-wise view of sequence data of several arrays, with the sequences aligned to the reference sequence and highlighting non-reference calls. Editing sequences is done in place by typing or via a context menu. This view also summarizes the base composition of the reference via a colored bar at the top of the window. An overview plot shows the type of base calls present in the alignment: reference calls are gray, missing calls are blue, and homozygous and heterozygous non-reference calls are highlighted in green and orange, respectively. A heterozygous call is present if two variations if a position were found and no single base could be called. To enhance productivity, searching for non-reference sites and bookmarking is implemented. An automatically updated list of all non-reference sites allow to swiftly find positions of interest. The list also allows editing, and it allows to show a detailed view of the position, including a line chart of the intensities and quality values. ResqMi implements the concept of coordinated views at many points. Selections of in the intensity plots are communicated from and to the resequencing window and to the non-reference site list and several other views.

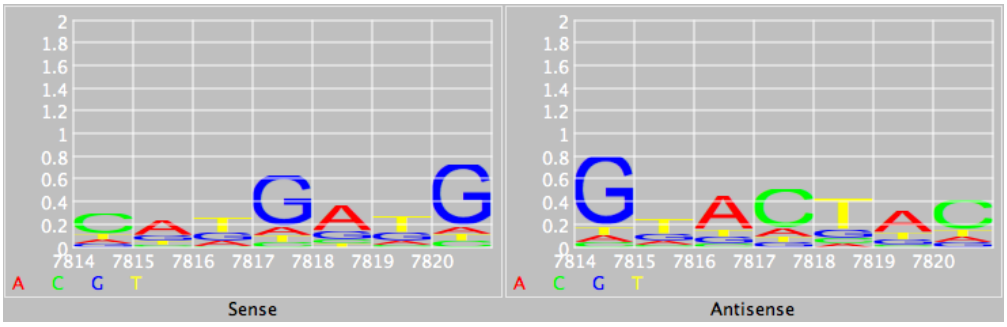
To view and compare the called sequences of a large number of CHP files, a *sequence heat map* can be used (see figure 4.5 for an example). It shows the base composition, with the sequence positions in the rows, the arrays in the columns. The bases are color coded, the encoding can be configured by the user.



(a) Line Chart

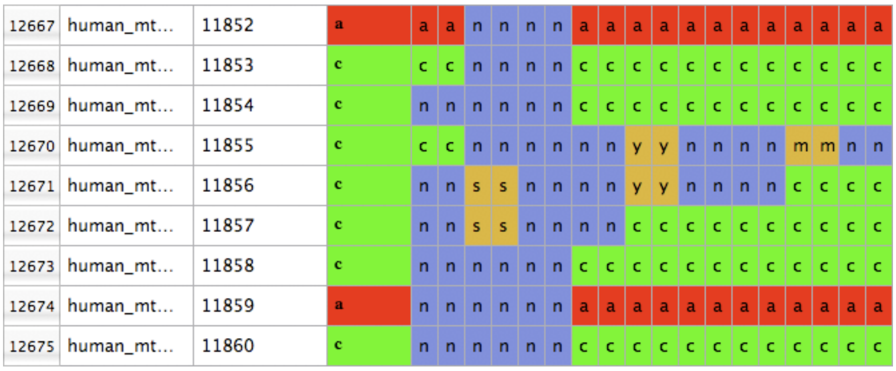


(b) Spike Plot



(c) Sequence Logo

Figure 4.4.: Intensity and sequence visualization in ResqMi. (a) Line chart, (b) Spike Plot (c) Sequence Logo of the same position. (d) Sequence heat map showing the calls at a problematic site for various arrays. Rows represent sequence positions, columns represent different arrays.



12667	human_mt...	11852	a	a	a	n	n	n	n	a	a	a	a	a	a	a	a	a	a	a	a
12668	human_mt...	11853	c	c	c	n	n	n	n	c	c	c	c	c	c	c	c	c	c	c	c
12669	human_mt...	11854	c	n	n	n	n	n	n	c	c	c	c	c	c	c	c	c	c	c	c
12670	human_mt...	11855	c	c	c	n	n	n	n	n	n	y	y	n	n	n	n	m	m	n	n
12671	human_mt...	11856	c	n	n	s	s	n	n	n	n	y	y	n	n	n	n	c	c	c	c
12672	human_mt...	11857	c	n	n	s	s	n	n	n	n	c	c	c	c	c	c	c	c	c	c
12673	human_mt...	11858	c	n	n	n	n	n	n	c	c	c	c	c	c	c	c	c	c	c	c
12674	human_mt...	11859	a	n	n	n	n	n	n	a	a	a	a	a	a	a	a	a	a	a	a
12675	human_mt...	11860	c	n	n	n	n	n	n	c	c	c	c	c	c	c	c	c	c	c	c

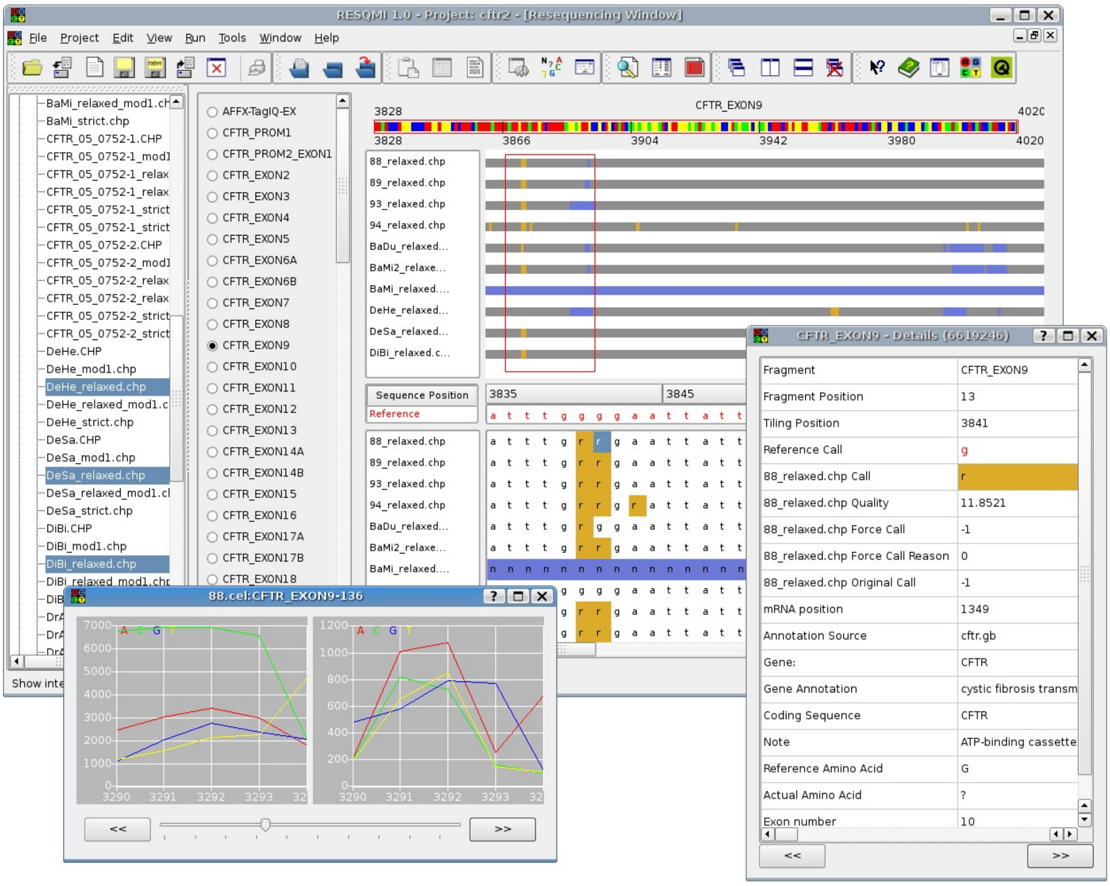
Figure 4.5.: Sequence heat map showing the calls at a problematic site for various arrays. Rows represent sequence positions, columns represent different arrays.

ResqMi features various export functions. It can export intensity into tab-separated files and sequence data into FastA, Phylip and plain text formats. Furthermore, reports can be generated from sequence data, summarizing the resequencing results. The number and positions of uncalled bases and non-reference calls are listed, as well as an overview of the processing of the array is shown.

Identifying the impact of a mutation is an important task in the analysis of resequencing microarrays, especially finding mutations that trigger a change on the protein level. Given a mapping of resequencing fragments to the genomic or mRNA sequence of a gene, ResqMi displays detailed information on the position, whether it is within the coding sequence, intron, exon or if this position has been identified as SNP position. These mappings can be produced using a tool called ResqMap from Genbank annotation files.

Based on the plugin framework for ResqMi, several methods for base calling are implemented in ResqMi. The Affymetrix implementation of the ABACUS algorithm can be called via the “Analyze” plugin, however this is only possible under Windows and requires the Affymetrix software to be installed. ResqMi has a native implementation of Clark’s calling algorithm, which allows to swiftly produce called sequences. The user can select the thresholds μ, ν and the type of call, i.e. which calling strategy (strict consensus, or relaxed consensus) should be used. It is also possible to ignore the sense or antisense strand when calculating calls.

Uncalled positions can be reviewed using *ReAnalyze*. ReAnalyze (see [203] for the original concept) helps to reduce the tedious task of manually reviewing single positions. The concept of ReAnalyze involves producing calls for positions for which a homozygous call can be made. For these position, the call matches the reference and is based on a high signal to noise ratio in both strands. This is done in order to mimic the manual editing of a position. As ReAnalyze does not produce any non-reference call, especially no heterozygous call, it will not introduce false mutations. As an extension to ReAnalyze for increased confidence, a cutoff for the conformance (see above) can also be defined. If a position lacks conformance, it will remain uncalled. Experiments have shown that



Fragment	CFTR_EXON9
Fragment Position	13
Tiling Position	3841
Reference Call	g
88_relaxed.chp Call	r
88_relaxed.chp Quality	11.8521
88_relaxed.chp Force Call	-1
88_relaxed.chp Force Call Reason	0
88_relaxed.chp Original Call	-1
mRNA position	1349
Annotation Source	cftr.gb
Gene:	CFTR
Gene Annotation	cystic fibrosis transm
Coding Sequence	CFTR
Note	ATP-binding cassette
Reference Amino Acid	G
Actual Amino Acid	?
Exon number	10

Figure 4.6.: Resequencing window of ResqMi, including an intensity view (bottom left) and position details, exemplified here for the case of a mutation in the CFTR gene.

Table 4.1.: Key features of the datasets used. Note that each array contains several fragments for sites of known mutations of the main target.

Name	Target	Bases	Fragments	Experiments
CFTR	CFTR	9511	84	17
Mito	Human Mitochondrium	37756	480	14

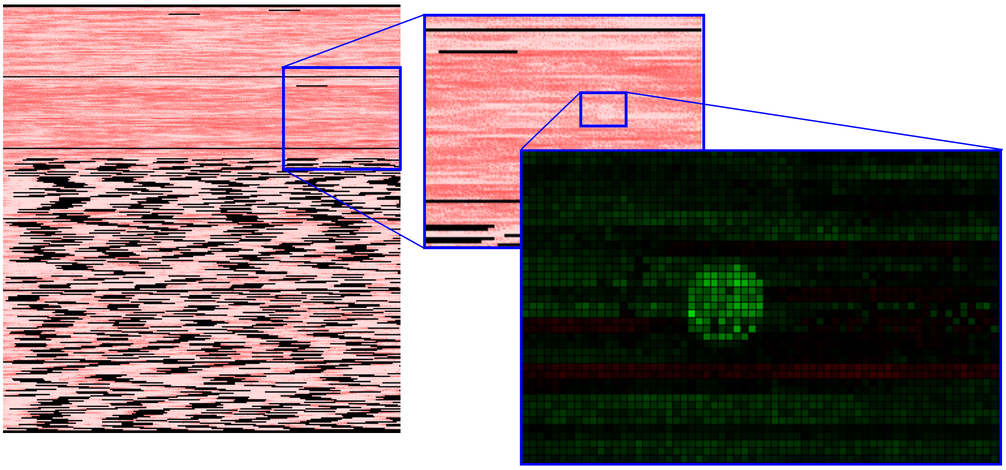
ReAnalyze is capable, with conservative cutoffs, to remove a significant fraction of the uncalled positions [178].

4.2.1. Application: Comparison of Base Calling Algorithms

In this section, ResqMi is applied to typical tasks in the analysis of resequencing microarrays, especially quality control and base calling. Two datasets produced with different resequencing microarrays designs are inspected using ResqMi. The key features can be found in table 4.1. Target sequences are the human cystic fibrosis transmembrane conductance regulator ("CFTR", dataset from [203]) and the human mitochondrion ("Mito"; dataset available from Affymetrix; `www.affymetrix.com`).

ResqMi is first used for quality control of the arrays of the Mito dataset via image plots of intensities and standard deviation. In one of the arrays of the Mito data, we find a circular area of spots with overall low intensities, lower than in all other arrays of the dataset (see figure 4.7). The sequence positions represented by the affected spots can be affected by poor quality calls.

For the comparison of calling algorithms, CHP files generated with the Affymetrix algorithm and Clark's method, using a number of parameters were used. For the latter, every combination of conformance parameter $\mu \in \{0.5, 0.6, 0.7, 0.8, 0.9\}$ and ratio parameter $\nu \in \{1.01, 1.05, 1.1, 1.25, 1.33\}$, both for strict and relaxed consensus calls was tested. For each parameter combination and array we computed the ratio of no-calls and the number of non-reference calls (other than "`N`"). The results are summarized in figure 4.8. The impact of parameters ν and μ depends on the dataset. In general, we observe that large values of the signal to noise ratio ν correlate with higher rates of `N`-calls, especially for the CFTR dataset. The effect of the conformance cutoff μ is higher in the Mito dataset, a very high cutoff, $\mu = 0.9$ leads to high `N`-rates in both datasets. The relaxed consensus method consistently produces lower `N`-rates. Vice versa, the frequency of non-reference calls is higher for the relaxed consensus method than for the strict consensus. The impact of μ, ν on the number of non-reference calls is similar to the effects on the calling rate. For low values of μ and ν, the rate of discrepancies is higher than for more restrictive ones. The ratio cutoff, however, has a far lower impact on the rate as the conformance cutoff. Naturally, the relaxed consensus method leads to more discrepant calls than the strict consensus method, especially at lower values for μ.



left, center: probe intensities: low high right: M-values: low medium high

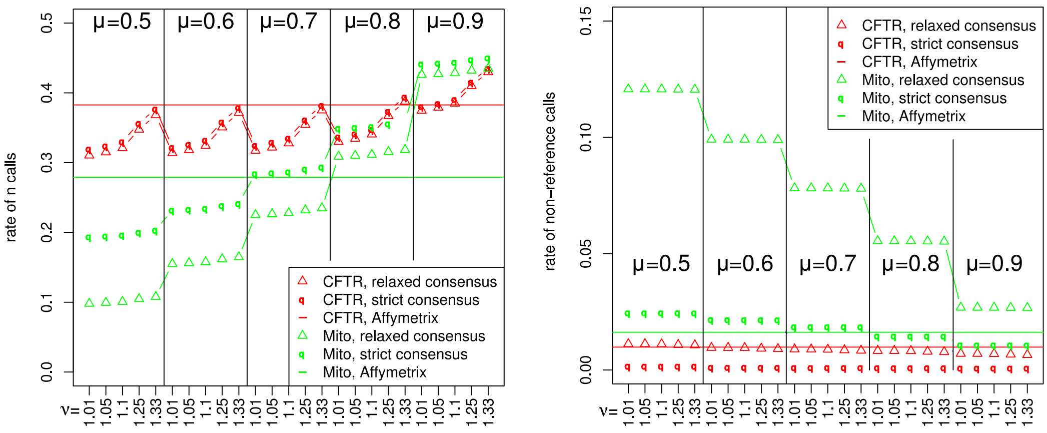
Figure 4.7.: Potential quality issue in the Mito dataset. The left image shows the probe-wise intensities (using a red-white color gradient). A part of the image is enlarged (upper center). The M-value plot shows the comparison with a reference array. The defective spot has considerably lower intensities than on other arrays in the dataset.

Note that generally calls made using low values of μ and ν might be unreliable. Therefore, choosing more restrictive parameters can increase accuracy, albeit at a lower call rate. Especially in the CFTR datasets, some arrays exist for which no base can be reliably called (see table 4.2). These arrays are not usable, even with very relaxed parameters. The no-call rate of the array displayed in figure 4.7 is in fact the highest for the Mito dataset (relaxed consensus: 34.7%, strict consensus: 38.4%).

In general, Clark's calling algorithm performs similarly to the Affymetrix implementation, both in terms of uncalled bases and non-reference bases, depending on dataset and parameters. Furthermore, there is a good concordance between both methods: Kohens K is ≥ 0.54 for both consensus settings and $\mu = 0.8$ and $\nu = 1.25$ (see table 4.2). For a more detailed evaluation of calling methods, see [178].

Table 4.2.: Mean percentage of `N`-calls, percentage of discrepancies and Kappa statistic (K) for all datasets using parameters $\mu = 0.8$, $\nu = 1.25$. Note that K is equal for both consensus methods.

	Strict consensus call			Relaxed consensus call			
Name	%`N`	range	% disc.	%`N`	range	% disc.	K
CFTR	37.3	(18.5...100)	0.1	36.7	(17.9...100)	0.8	0.56
Mito	35.6	(32.3...38.4)	1.4	31.5	(27.2...34.7)	5.5	0.54



(a) Mean no-call ratio

(b) Mean non-reference call ratio

Figure 4.8.: Mean no-call ratio (left) and discrepancies ratio (right) in three datasets using strict consensus calls (triangles) and relaxed consensus calls (q) for different values of μ and ν. For comparison, the two horizontal lines show no-call-ratio of the Affymetrix algorithm on the same data.

4.3. Discussion and Outlook

ResqMi is the first open source and multi-platform software for the analysis of resequencing microarray data. In order to allow a fast and flexible evaluation of intensity and sequence data, ResqMi features a graphical user interface and allows easy data handling. The main strength of ResqMi is the power to visually inspect and analyze many aspects of the data in different ways. The inspection of intensities and comparing them between arrays allows to identify even subtle spatial effects. ResqMi offers several views of the data, from large-scale overviews in image plots, reports and sequence heat maps to spot-oriented views of the intensity data necessary to make most of the data available. This allows the efficient manual revision of base calls. Furthermore, it allows to estimate the actual impact of a mutation to be identified easily. Since all calling algorithms fail to call bases for many sequence positions, efficient manual inspection features like the ones offered by ResqMi are important.

The implemented base calling algorithm is fast and produces results comparable to the Affymetrix method. For data with quality issues, calls can be made with permissive thresholds. However, it must be noticed that the confidence of calls is reduced in these cases. ReAnalyze helps to further increase the calling rate for clearly homozygous loci. While this method is not error-free it complements classical calling algorithms.

A trade-off exists between the calling rate, and the false-positive rate. The approaches relying on conformance and signal-to-noise ratio are prone to errors, as many uncalled positions are due to lack of quality and allow no reliable base call. Evaluations of calling algorithms require a set of golden standard sequences produced with an absolutely reliable method to establish correct estimates for the calling rate. The called sequences determined in the above study are not compared to such a standard and therefore, no false positive rate can be established. However, the approach is useful as it facilitates manual revision of called sequences. Due to the nature of the data and the relevant algorithms, which do not call many base, the presented approach is helpful. The alternative would be ignoring all uncalled positions and potentially miss important mutations or analyzing everything manually. Using ResqMi, a whole analysis of a resequencing dataset of 40 arrays including import, base calling, ReAnalyze and generation of reports takes less than 5 minutes.

4.3.1. Possible New Applications

ResqMi only processes arrays produced with the Affymetrix technology. So far, most resequencing arrays published were produced by Affymetrix, encompassing one catalogue array and many custom arrays. Still, despite their usefulness, especially in clinical application, the use of resequencing microarrays is declining. The recent developments in second generation sequencing have also affected the problem of gene specific resequencing and introduced new methods. Methods for so-called “targeted resequencing” are offered by all suppliers of sequencing technologies. In general, this process involves amplification of the target sequences using PCR. Identifying the target regions is done using marked “bait” RNA molecules [65]. Sequences are then determined using a second-generation sequencing method (see section 2.3). An evaluation of this approach can be found in [70]. The new protocols allow to determine far longer sequences than array-based sequencing ($\geq$ 260 kbp vs 30 kbp), which can also encompass several genes. Furthermore, the new protocols allow to find all mutations, including long insertions and deletions. Using such methods, resequencing can be improved compared to microarrays, as the overall quality of sequencing technologies is considered superior to hybridization based methods. Furthermore, the base calling from intensities would be unnecessary, as this problem is addressed by the sequencing technology.

Gene specific resequencing, regardless of the technology has advantages. If in clinical settings a single gene needs to be inspected, as in the example of cystic fibrosis, it is unnecessary to sequence a whole genome. Even with second generation sequencing technology, the process is too expensive and time consuming, due to the computational and manual data processing. Therefore, especially in clinical practice, gene specific resequencing (also using microarrays) stays relevant.

ResqMi is based on data structures modeled around the Affymetrix resequencing platform. Major parts of ResqMi would need to be reimplemented for next-generation resequencing applications. Therefore, future directions would include using the visualizations and user interfaces evaluated in ResqMi in new applications. In that setting, the current version of ResqMi can be seen as a template for new tools for the analysis of second generation resequencing data analysis. Especially, the visualization and comparison of sequences remains an important task which has not changed by the technology, except for the intensity-based features.

Enhancements on the sequence heat map, following the visual analytics paradigm, would be useful: automatic and manual aggregation of positions and inclusion of quality estimations (quality scores, read count, etc). In general, some of the position based features like the line chart (for quality measurements) and the sequence logo (when comparing different samples) remain useful. Most challenges are posed by the increased length of sequences. This may increase the use of search and highlighting of non-reference sites, like in ResqMi's non-reference site table. The integration of gene expression data and external annotations allows a better informed estimation of mutation effects. Furthermore, information about gene models would be useful to estimate mutation effects on a per-transcript level. In general, based on all available information, new applications should make finding interesting sites easier. Any advanced resequencing analysis tool will share properties with modern genome browsers. Relevant features are visualization, annotation and handling of large datasets.

ResqMi currently works on sequences in a per-fragment style. This is imposed by the Affymetrix technology. However, it might also be useful to generally structure resequencing data in this way if manual inspection is necessary. A triage of fragments is possible, e.g. first inspect exons, then introns, finally intergenic regions. Clever handling of multiple genes is also important. Based on this, fragment-wise sequences can be viewed in conceptually the same resequencing window as shown in figure 4.6.

In summary, resequencing has undergone the same explosion of dataset sizes as the analysis of gene expression. This results in high quality datasets, which introduce new challenges. A part of the methods necessary to handle these datasets, especially overview and search functions, are already part of ResqMi. Scaling the current methods and introducing new ones to match the new datasets can make the new resequencing protocols even more successful than array-based resequencing.

5. Context based Visualization of Gene Expression Data

In this chapter, methods for visualizing gene expression data are discussed. Gene expression data consists of far more than the expression matrix and the raw data that was used to produce it. A large number of gene annotations is available that is important to include into analyses and visualization because it represents the condensed knowledge of the biomedical community about genes. Furthermore, numerical values calculated during the data analysis, for example probe expression level summaries or results from feature selection methods, is important information that should be included in visualizations.

This chapter first introduces the most important types of meta information encountered during gene expression analysis and gives a brief overview of existing strategies of processing and visualizing meta information. The following sections present a set of tools for different ways of visualizing meta information. These tools are partly based on common visualization tools as summarized in chapter 3. For each tool, a theoretical motivation is given. Then an implementation for Mayday is described. This implementation is then used to briefly discuss the interpretation and possible applications of the respective tool. A discussion concludes this chapter.

5.1. Meta Information

With exhaustive assays querying gene expression and growing knowledge of biological systems, gene expression data is analysed in the context of other findings. Gene expression data usually consists of large expression matrices (see section 2.5.1) which contain the measured data. In addition, genes are associated with a variety of meta data that in terms of size and complexity can by far extend the DNA sequence of the gene as well as the measured expression values in a single study.

Common *meta information* available for genes include alternative names and database identifiers, as well as functional, and spatial annotations, such as Gene Ontology (GO) [6] terms. Furthermore all kinds of biological findings related to the gene are meta information, for example regulatory factors and interaction partners (for example from the STRING database [181]) of the gene products. Biological models like pathways (for example from KEGG [91], Reactome [130], MetaCyc [29]) or reaction models (from BioModels [114]) a gene or gene product is part of are meta information about that gene, albeit on a higher level. Activity measurements derived in other studies can complete the knowledge about a gene. Higher level features including phenotypes, diseases and nearby genomic features are also of interest. Molecular structures of proteins and metabolites give insight into the chemical basis of the process. This data has a wide range of cardinalities for each gene and is available in many different data types. Table 5.1 gives a summary of the most common types of meta information. While much information is available in the literature, an increasing effort is under way to structure this information and make it readily available for research.

An extension to this general meta information, *experiment-specific* meta data is accumulated during the analysis of a study. This includes statistical values from quality control, statistical tests, clusterings, and other procedures. Furthermore, probe sequences, read sequences and laboratory processing details are of interest. Again, the cardinality and data types of such information vary greatly (see table 5.2 for examples).

Meta Information	Examples	Data Type
Alternative Names	UniProt id, gene name, locus id	String
Function	Gene Ontology, Sequence Ontology	String, Tree
Spatial pattern	Gene Onology	String, Tree
Functional Context	Pathways, Models	String, Network
Gene Loci	Genomic coordinate	Number
Structure	Protein Structure	Complex, Image
Interaction Partners	STRING	List, Network

Table 5.1.: Typical meta information about genes. This table gives some examples about the most common meta information available about genes in an expression study, along with sample sources of such information, their typical data type and the expected number of elements per gene.

Meta Information	Examples	Data Type
Statistical values	p-value from hypothesis tests	Number
Quality control	Spot quality flags, noise intensity, signal to noise ratio	Bit vector, Number
Summaries	Mean, median, standard deviation	Number
Sequences	Probe sequence, read sequence	String
Processing details	Spot coordinate, probe origin	Tuple, String
Analysis results	Hierarchical clustering tree	any

Table 5.2.: Meta information derived during studies. The examples concentrate on transcriptomics studies, but especially statistical values, summaries and analysis results are valid for a wide range of studies.

Handling of meta information is commonly implemented in gene expression analysis software. Most of the above data can be analysed and inspected with R [147] and the BioConductor [64] suite. Most available microarray analysis tools provides means of analysing and visualizing meta information to some extent. Common operations on meta information range from simple filtering, e.g. on quality control values or statistical test results to higher level analyses like analysis of gene set enrichment, which is most useful for the analysis of GO terms and pathways.

Visualization of meta information can be performed using a variety of classical visualization tools, like scatterplots, histograms or bar plots (see section 3.1 for an overview). For structured meta information, especially for those based on formal ontologies, trees or graphs are a natural representation. For hierarchical clusterings, trees are used for visualization. The heat map can provide a visualization that combines clusterings with the primary data. As another tree based approach, Timeline Trees [25] combine hierarchies and measured data. The enhanced heat map combines the expression data, also displayed in a heat map with numerical and categorical meta information [63].

In general, it is often not useful to visualize a whole expression matrix. Most visualization tools, suffer from overplotting, lack of overview and other issues resulting from too much input data. Therefore, either the number of genes must be reduced (according to meta information or using clustering) or intelligent summaries of the genes must be employed. The same is true for meta information. The extent of meta information can be overwhelming and requires careful structuring, both for handling and visualization.

Mayday features a large number of different visualizations and analysis tools for meta information. The meta information concept used by Mayday is capable of working with a variety of derived values and external annotations, for the latter, often requiring no or only minor preprocessing. Basic visualization tools are available in Mayday, for example the enhanced heat map, histograms and scatterplots. These tools are restricted to few types of meta information (mostly numerical) and make poor use of textual meta information. For example, it is hard to get an overview or a comparison of which annotations are associated with different sets of probes. To address this problem, in this chapter, several methods for context-based visualization of gene expression data are investigated.

In the following sections, methods for graphically structuring gene expression data with the focus on their meta information are presented and discussed. Profile logos (section 5.2) employ a weak form of meta information, as they rely on summarized discretizations of the primary data. Tag clouds and chromograms (section 5.3) deal with textual information, as do term pyramids (section 5.4). Numerical meta information, albeit the easiest to display, can be structured and compared with probe rank plots (section 5.5).

5.2. Profile Logo

Sequence logos [159] (see section 3.2.6 for details) are a common tool to visually summarize multiple sequence alignments. The main application is visualization of motifs in nucleotide and protein sequences. However, the concept of sequence logos can easily be adapted to the visualization of any data matrix. Let A be an alphabet of discrete symbols $b \in A$. From a column of a multiple alignment of sequences of symbols from A, the frequency of symbols $f(b)$ is calculated for all $b \in A$. $f(b)$ is then scaled according to equations 3.1 to 3.4). A sequence logo is a bar chart visualization of these scaled frequencies. To display a numerical data matrix, e.g. an expression matrix, as a sequence logo, a discretization function is necessary. The discretized data can then be displayed in the same way as a multiple sequence alignment.

The application of the sequence logo concept to discretized data matrices is especially useful in the context of gene expression data. It could be employed to characterize clusters of expression profiles and help to identify interesting (i.e. variant or uniform) columns in expression matrices. The name sequence logo is not fitting for this purpose, as several of the characteristics of biological sequences, most importantly the fixed order of the residuals are not necessarily observed in expression profiles. Consequently, a graphical representation of the frequencies of symbols in a discretized matrix, scaled by their information content is here called a *profile logo*.

Next, several discretization strategies and their applicability to profile logos are discussed. Then, an implementation of the profile logo concept along with several examples is presented. The interpretation of profile logos is discussed at the end of the section.

5.2.1. Discretization of numerical data

Discretization (in this context also often called *binning*) is the process of transforming a vector of continuous numerical values into a vector of corresponding discrete (nominal or ordinal) values. These values are from a set of predefined levels and represent non-overlapping subsets of the original data, often called bins. More formally, it can be defined as a function

$$f(x) : \mathbb{R} \mapsto S$$

where S is a set of $s = |S|$ discrete values. In the context of gene expression data analysis, a binary binning with two (e.g. regulated, not regulated) or three symbols (up-regulated, down-regulated, unchanged) is commonly used. Such strategies are often used for supervised and unsupervised machine learning purposes. Several binning strategies have been investigated in this context [209]. Let $x \in \mathbb{R}^n$ be a vector of continuous values, and let $x_{min} = \min(x)$ and $x_{max} = \max(x)$. In *equal-width binning*, thresholds are chosen to form intervals of equal width $w = \frac{x_{max} - x_{min}}{s}$ for an arbitrary $s > 0$. This method, however, has been found liable to outliers [45]. In contrast, *equal-frequency binning* creates bins with identical size. The number of bins can be chosen arbitrarily in both methods. However a single bin, or an extremely large number of bins, so that most of them are empty, are most often not useful.

In general, any set of n thresholds gives rise to a binning with $s = n+1$ bins. Therefore, many strategies exist to choose one or two thresholds based on the overall properties of the data [123]. Some of these strategies are actually special cases of one the above general strategies for discretization. The overall mean or median value of a data sample, possibly combined with its standard deviation can be used to induce a binary or ternary binning. A binary discretization based on the mid-range can be achieved using the *mid-ranged discretization* method, which has been shown to produce biologically relevant results when applied in biclustering [123].

Equal frequency binning results in bins of equal size. Thus, the information content of every column of a resulting profile logo would be zero, making method is unsuitable for the creation of profile logos. Using only the median value as a threshold is a special case of equal frequency binning and is therefore also not usable for profile logos.

For the discretization of time series data, the difference between time points can be used. *Transitional state discrimination* [134] takes as as input a data matrix A, which is z-score normalized, resulting in matrix A'. The discretized form A^d is defined as

$$A^d_{ij} = \begin{cases} 1, & \text{if } A'_{ij} - A'_{ij-1} \geq 0 \\ 0, & \text{else} \end{cases} \tag{5.1}$$

for $j > 1$ or, in order to obtain a ternary binning, given a threshold $t > 0$ for the maximal allowed difference between values to be considered constant:

$$A^d_{ij} = \begin{cases} 1, & \text{if } A'_{ij} - A'_{ij-1} \geq t \\ -1, & \text{if } A'_{ij} - A'_{ij-1} \leq -t \\ 0, & \text{else} \end{cases} \tag{5.2}$$

Note that A^d is defined with one column less than A', so set $A^d_{\bullet,0} = 0$. This method is only applicable to time series data with unique time points or data where the difference between two experiments is meaningful. Several modifications and extensions of this method have been proposed. However, the binary variant (eq. 5.1) of this method also performed well in the biclustering study in [123].

Many supervised discretization methods are also available [209]. While these methods are useful in the context of machine learning, usually no class partitions are available in the context of expression data visualization. Therefore, only unsupervised strategies are considered here for profile logos.

5.2.2. Implementation in Mayday

The Profile logo plugin for Mayday is an implementation of the profile logo concept. It is implemented as a plugin for the Mayday visualization system. The plugin works by calculating a discretization of the input data and transforming it to a profile logo. The profile logo is then displayed on the screen. An extension point exists that allows to add new discretization strategies via the plugin interface of Mayday. All binning strategies must extend the class `AbstractBinningStrategy`. The discretization strategy used for generating the profile logo is configurable. Upon selecting a new strategy or modifying the parameters, the diagram is redrawn. Currently implemented discretization strategies are:

- *Threshold binning*: Arbitrary thresholds can be entered for binning. This allows the users to adopt to special ranges of data (e.g. z-scores or M-values) or specific questions, like identifying strongly regulated genes, e.g. with absolute fold change > 5.

- *Equal width binning* (with 3 bins) is the default setting. The number of bins can be chosen between 2 and 20.

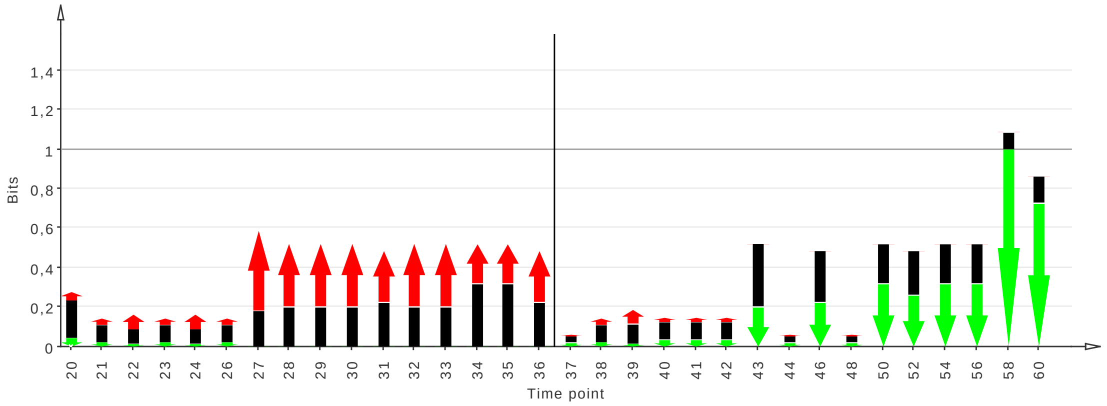


Figure 5.1.: Example of a profile logo, created by the Mayday profile logo plugin. It shows the expression values of a cluster of 13 genes related to energy metabolism in *Streptomyces coelicolor* [140]. Equal-width binning with 3 bins was used. The horizontal bar indicates the depletion of phosphate in the medium.

- *Transitional state discrimination*, as defined in equations 5.1 and 5.2. The threshold t can be chosen by the user and is interpreted on the z-score scale.
- *Average binning* using either the mean or the mid-range as a threshold. In addition, a ternary binning can be achieved by considering all values within one standard deviation from the average to be unchanged.

Similar to sequence logos, in which the symbol frequencies are displayed by a scaled letter, profile logos use arrow symbols to represent bins. The arrows are chosen according to the number of bins. For two bins, the lower bin is represented as a downward pointing arrow (↓), the upper bin is indicated as an upward pointing arrow (↑). For a ternary binning, the middle bin is represented as a box. For more than three bins, boxes are used for all bins. All arrows and boxes are scaled to their respective height using equation 3.4. The coloring scheme used for the bins can be configured by the user. To indicate special positions or regions of interest, vertical lines can be introduced to the profile logo. An example of a plotted profile logo with 3 bins can be found in figure 5.1. It also shows an automatically created legend that indicates the thresholds used for discretization.

In contrast to sequence logos, the largest bin of probes is not necessary on the top of a profile logo column. While the ordering of the one letter codes for DNA or protein residuals is more or less arbitrary, the ordering of the bins is well-defined and is therefore preserved in the profile logos created by this plugin, even if the bottommost bin is the largest by size.

5.2.3. How to Interpret Profile Logos

Schneider *et al.* [159] point out five features of sequence alignments that are displayed in sequence logos. Based on these points, the features of profile logos, depending on different discretization strategies are discussed. As an example, a cluster of 24 antibiotics related genes in *Streptomyces coelicolor* is used. Figure 5.2 shows the expression profiles of the genes as a profile plot and three profile logos created using different discretization methods. The features Schneider *et al.* mention are:

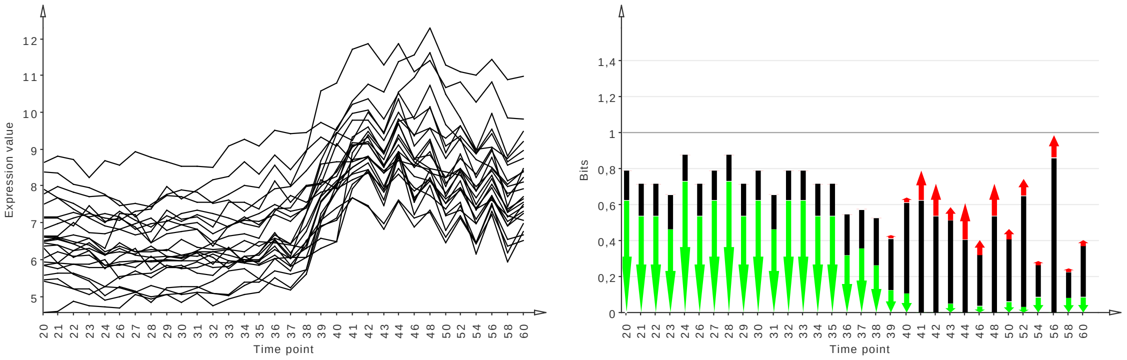
- "The general consensus of the sequences". Profile logos show the consensus sequence of symbols representing discretized values. The most frequent symbol is, however, not placed on the top of each columns, as the order of the symbols is preserved. Still, the consensus sequence can be identified by finding the largest symbol in each column. For example, by visual inspection, users can easily determine that in figure 5.2d, most profiles are above the grand mean after 40h. From figures 5.2c and 5.2b it can be seen that the profiles remain relatively stable on a lower level for the first half of the experiments. In a nutshell, profile logos summarize the overall profile of cluster of profiles, even though the most frequent symbol is not painted atop.

- "The order of predominance of the residues at every position". The order of predominance cannot as easily found in profile logos as in sequence logos, due to the ordering of the symbols by their numerical values. It is possible to obtain this ordering by mentally sorting the symbols by height, but this is not as intuitive as in sequence logos. Still the natural order of bins should be preserved, as the natural ordering of the values helps to map the symbol to the range of values they represent. Furthermore, in most cases, profile logos show two or three symbols, which makes determining the largest symbol very easy, as can be see in the examples shown in figure 5.2.

- "The relative frequencies of every residue at every position". This features is preserved in profile logos. The height of each symbol is proportional to the frequency of each symbol. However, the height of two symbols cannot be directly compared between columns due to the scaling of the overall column height by the information content. A stacked histogram or a bar chart would be more useful for this purpose.

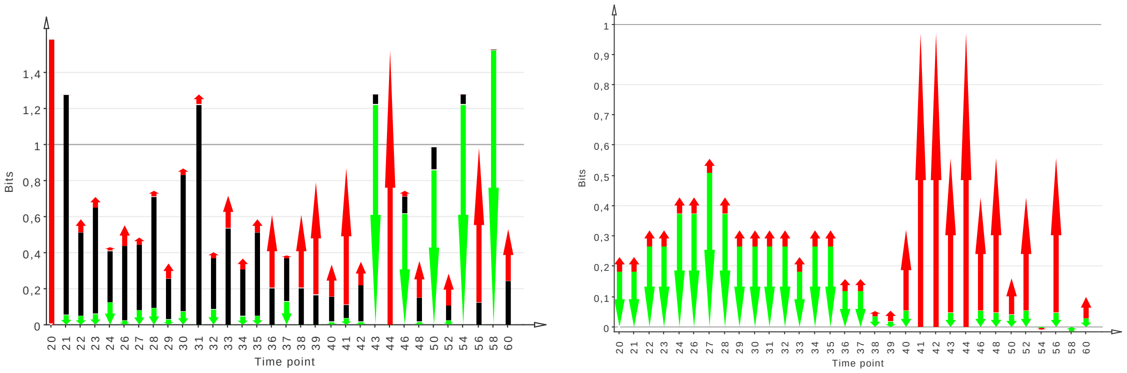
- "The amount of information present at every position in the sequence, measured in bits". This feature is indicated by the total height of a column, which can easily be determined in profile logos. In most cases, for columns with high information content, the values are more uniform, for a low information content, the values are more scattered.

- "An initiation point, cut point, or other significant location (if appropriate)". This feature is useful to indicate key features of the profile in a straightforward way. An example can be found in figure 5.1. There, the vertical bar indicates the time point during the fermentation when the phosphate in the medium supporting the bacteria was depleted. As it can be seen from this figure, this correlates well with a significant (downward) change in the profile of many genes, which can also be seen in figure 5.2d.



(a) Profile Plot

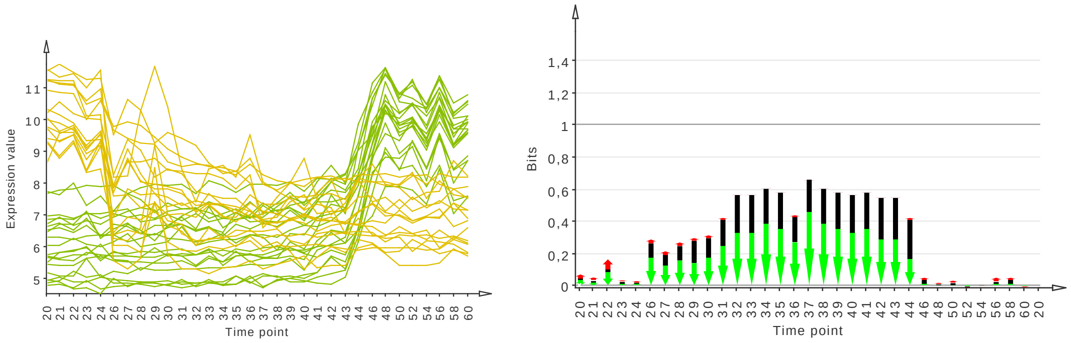
(b) Equal Width Binning



(c) Transitional State Discrimination

(d) Average Binning: Mean

Figure 5.2.: Profile plot and three different profile logos of a cluster of 24 antibiotics related genes in *Streptomyces coelicolor* [140]. (a) Profile plot; (b) equal width-binning with 3 bins; (c) Transitional state discrimination with $t = 0.25$ and (d) average binning with the total mean as cutoff.



(a) Profile Plot of two clusters (yellow, green) (b) Profile logo of the same cluster

Figure 5.3.: Profile logo of a mixture of clusters. The information content in the first and last time points is low, because the clusters have different profiles. The profile logo is a poor representation of different profiles of clusters.

The discretization method can have a significant effect on the profile logo. However, equal width binning with two bins and mean- or mid range-discretization do not differ that much in their effect on the data, that the interpretation is vastly different. Choosing custom thresholds or a high number of bins can vastly change the representation of the data. However, this allows profile logos to be adapted to a wide range of applications. In contrast to the other three methods, TSD has a different focus. Instead of summarizing the values, it summarizes the change of the values between time points. This is useful to identify inflection points (which would lead to a profile logo similar to “↑↑↑↓↓↓”) in profiles and monotonous (↑↑↑↑) or unstable (↑↓↑↓) profiles.

Profile logos are most successfully applied to single clusters, when the overall tendency is of interest. Then, a profile logo can provide a clean, concise summary of the cluster, which can easily be compared with other profile logos, as can be seen from the comparison of figures 5.1 and 5.2. Profile logos are not well suited to summarize a mixture of clusters or large unstructured sets of profiles. In a mixture of clusters, the clusters can interfere, which leads to very low information content per column. The shape of the individual clusters would be lost. The result would be a profile logo with very small columns that would be hard to interpret (see figure 5.3). Therefore, the application of partitioning clustering algorithms, like k-Means or QT clustering [121, 75]. greatly enhances the use of profile logos.

5.3. Chromogram and Tag Cloud

Chromograms and *tag clouds* are two visualization tools that display textual information. More precisely, key words, often calls tags, which are related to a dataset are shown. Chromograms display temporal patterns, while tag clouds directly display the frequencies of key words. With the abundance of meta information, many of which is textual in nature (see table 5.1 for an overview), these tools can be powerful additions to the

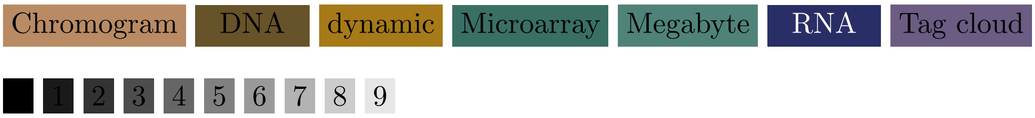


Figure 5.4.: Examples of the chromogram color transformation.

gene expression analysis toolbox. In this section, the two tools are described, and an implementation of them in Mayday is presented. Their application to gene expression data and notes on interpretability conclude this section.

5.3.1. Chromograms

Chomograms have been introduced by Wattenberg *et al.* [201]. The main idea is to use color to encode strings that summarize activities in a way that similar strings are encoded by similar colors. The application of chromograms is the visualization of long sequences of short texts. Wattenberg *et al.* applied this to visualize edit operations on Wikipedia, visualizing the titles of the edited articles as a time series [201].

The input to chromograms is an ordered lists of strings. Commonly, the strings are ordered by some temporal key, however this depends on the application. A lexicographical order is less useful in most cases. Lemmatisation of strings can be a useful preprocessing step to unify inflected forms of words. The string is then converted to lower case and encoded as a color, based on the HSB color model. For this purpose, only the first three letters are used: The first letter determines the hue of the resulting colors, second and third letters determine saturation and brightness, respectively, which are chosen from a limited range to ensure the hue remains easily recognizable. Let c be a lowercase alphabet character, and let be 'a'=1, 'b'=2, etc.; then $v(c) = \frac{1}{26}(c - \text{'a'})$ is the letter value of c. Then the color (h, s, b) of a word $w = w_1, w_2, w_3, \ldots$ converted to lowercase is given as $h = v(w_1)$, $s = 0.3 + 0.6 \times v(w2)$ and $b = 0.4 + 0.5 * v(w_3)$ [201]. Strings starting with a number are converted using a grayscale color gradient. Figure 5.4 shows some examples.

The encoded strings are then displayed in a block view, a sequence of colored boxes, which fill the screen in reading direction. In the original application, the chromograms omit the text and instead use tool tips to display it. For applications to time series, Wattenberg *et al.* also presented a timeline view that can highlight bursts of activity [201] in a stem-and-leaf style.

Chromograms can be applied to the visualization of meta data of gene expression studies. A multitude of annotation data is available, for which temporal and other patterns are of interest. For ease of exposition, we assume that there is a unique annotation for genes investigated in an expression study (see section 5.3.5 for a discussion of this). Then, depending on the ordering of the genes, chromograms can be produced as described above. For example, it could be interesting to see if genes with an overall high expression pattern tend to have similar annotations. This could be achieved by creating a chromogram sorted by mean expression and searching for runs of similar colors.

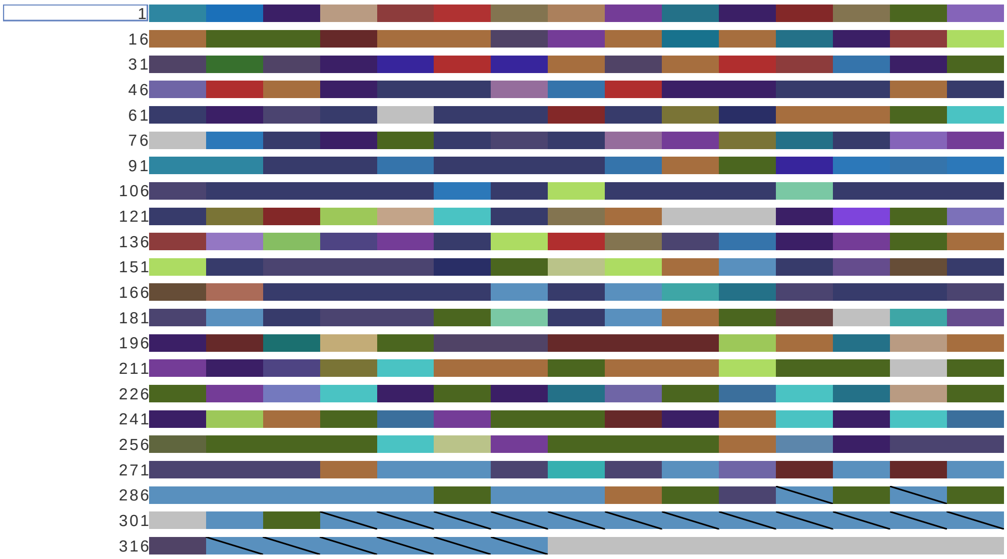


Figure 5.5.: Chromogram of Sanger annotations of variant genes in *Streptomyces coelicolor* [140]. The genes are sorted by the experiment in which they have the maximal expression. The stretch of marked genes at the end of the chromogram are annotated with "PKS" thus related to polyketide synthesis.

5.3.2. The Chromogram Plugin for Mayday

The chromogram plugin for Mayday is an implementation of the chromogram block view. Based on a table component, it uses annotations associated with sorted probes to produce chromograms as discussed above. The chromogram is created either from the the probe names, their alternative display names, or meta information. Either textual or lists of text-based meta information can be used. However, as chromograms display only one element per probe, currently only the first element of the list is used.

As genes have no unique natural order, and in general, no temporal ordering is available, the order of the genes has a significant effect on the resulting chromogram. There are many possible orderings, and the choice depends on the underlying question. Currently, the following ordering options are available in Mayday:

- *Probe name.* This function allows to exploit the fact that the probe identifiers can reflect structured information that has a useful ordering. This could for example be locus tags (which reflect the order of the genes on the chromosome), as often used for bacteria.

- *Probe display name.* The ordering by probe display names option is based on the same rationale as the ordering by probe name. However, this also supports non-unique names and is more configurable.

- *Number of probe lists.* This allows to prioritize the probes that are contained in a large number of probe lists. These might be key genes that are present in many pathways.
- *Experiment value.* Here, the user can select an experiment. The probes are then sorted by their expression value in this experiment.
- *Minimum, maximum.* Sorts the probes by their overall minimum or maximum across all experiments.
- *Mean.* This function can be used to sort the genes by their average expression value, allowing for example to identify runs of similar annotations in highly expressed genes.
- *Standard deviation, variance.* Allows to search for patterns related to the overall variation in the probes' profiles.
- *Meta information.* A configurable meta information can be used to introduce an ordering, for example significance values, feature selection scores or genomic loci.
- *Experiment with maximum / maximum.* This feature is useful for example in time series studies. Patterns of annotations that arise in probes peaking early or late in the time series can be found with this ordering.

The ordering can be performed either increasingly or descendingly. Note that the meta information used for sorting is not necessarily the same meta information as displayed in the chromogram. Figure 5.5 shows an example. It shows 322 genes in a fermentation time series of *Streptomyces coelicolor* [140]. We are interested in finding groups of genes that are highly expressed at the beginning or the end of the time series, therefore genes are ordered by the experiment which the maximal expression value occurs. We find several genes related to primary metabolism (ribosomal proteins, TCA cycle) to peak at the beginning of the time series. At the end of the time series, many genes from the secondary metabolism peak, especially genes related to polyketide synthesis.

Chromograms provide a condensed view of meta information. This view is most useful when combined with the underlying gene expression data. For this purpose, users can obtain a profile plot of each probe by clicking on them with the middle mouse button. As with all Mayday visualization plots, selections are communicated to and received from other open plots.

5.3.3. Tag Clouds

Tag clouds (also called word clouds) are a well-established tool to display the abundance of tags – key words associated with data. They are being used in many web-based applications, and are useful to highlight, for example, the major topics on a social web site. Tag clouds began to receive much attention as the photo sharing web site Flickr (`www.flickr.com`) used a tag cloud to visualize the frequency of the tags the site's users

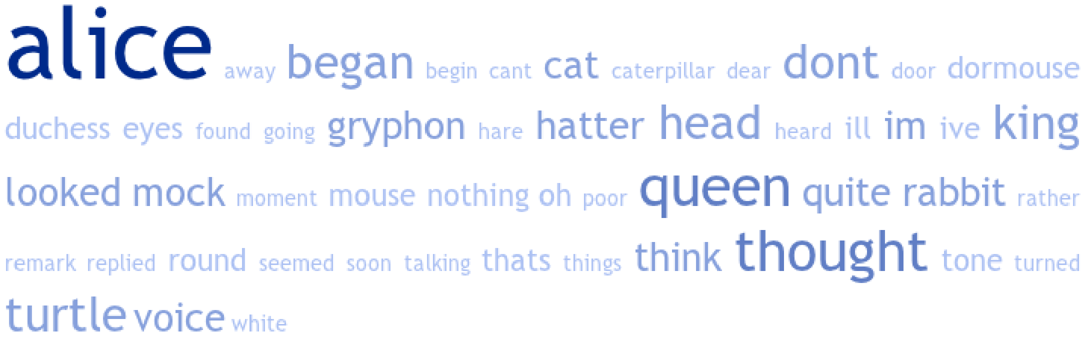


Figure 5.6.: Tag cloud showing word occurrences in "Alice's Adventures in Wonderland" [28] created by TagCrowd (`www.tagcrowd.com`).

added to the photos. Beside the display of such "folksonomies", tag clouds have been applied to text analysis. Figure 5.6 shows a tag cloud of the word counts in "Alice's Adventures in Wonderland" [28]. In addition, tag clouds have also been used extensively for illustrative purposes and infographics (see `wordle.net` for examples).

Tag clouds use the font size to display the frequency of a tag. This can be easily and efficiently implemented in HTML, which contributed to the popularity of tag clouds in social web sites. More formally, for a tag t with relative frequency $0 \leq f_t \leq 1$ the height h is given as $h \sim f_t$. A simple way of calculating the height of a tag cloud item is a scaling of the range of the tag frequencies to the range of font sizes. In the case of a larger range of tag occurrences, this may lead to unsatisfactory results. In this case, exponential transformation of the untransformed counts could highlight the most frequent tags, while log transformation is useful to highlight a larger number of tags.

The *layout* of tag clouds is subject of research. In general, the tags are embedded in the plane, in reading direction, in lexicographical order or by their frequency. Clustering of tags gives rise to alternative layout styles. A circular layout, in which the most frequent tags are placed in the center of the embedding is also used [122]. This has been performed using self-organizing maps [155], k-means clustering of similarity matrices [72] or automatic electronics design methods [94].

A user study [122] found that lexicographically sorted tags are best to find specific tags. A circular layout with the most frequent tag in the center optimally highlights this tag. Clustered layouts were found best to identify groups of tags belonging to a specific topic.

In the context of gene expression data, many annotations can be visualized by tag clouds. Functional gene annotations, for example each of the three parts of the Gene Ontology [6], or the Sequence Ontology [51] can be instantly used for this purpose. Also the membership of genes in biological pathways (e.g. from KEGG [90]) and structural properties (e.g. from Pfam [54]) are good sources of tags. Long textual annotations are less useful, as long tag names distort the tag cloud image.

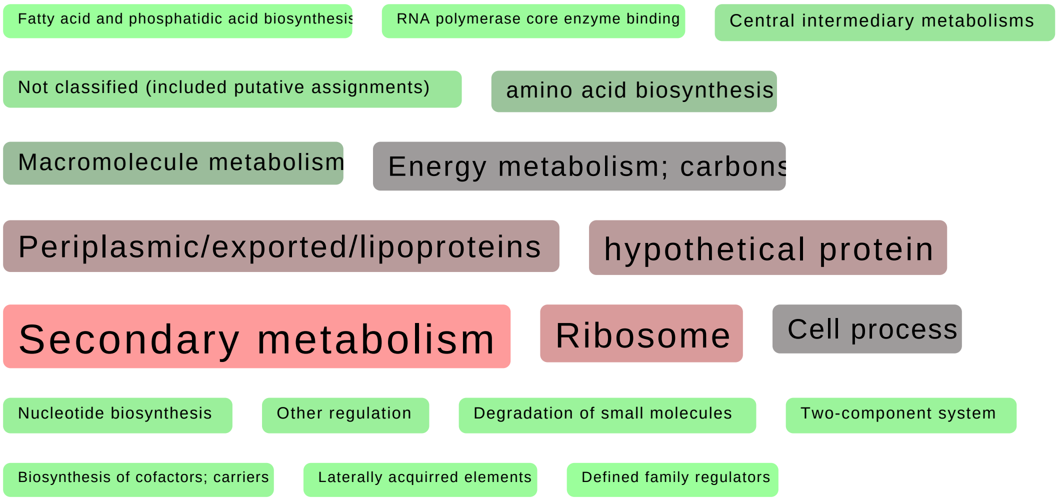

Fatty acid and phosphatidic acid biosynthesis
RNA polymerase core enzyme binding
Central intermediary metabolisms
Not classified (included putative assignments)
amino acid biosynthesis
Macromolecule metabolism
Energy metabolism; carbons
Periplasmic/exported/lipoproteins
hypothetical protein
Secondary metabolism
Ribosome
Cell process
Nucleotide biosynthesis
Other regulation
Degradation of small molecules
Two-component system
Biosynthesis of cofactors; carriers
Laterally acquirred elements
Defined family regulators

Figure 5.7.: Tag cloud showing TIGR/ JCVI annotations (`www.jcvi.org`) of variant genes in *Steptomyces coelicolor* created with the Mayday tag cloud plugin. The most frequent tags are placed at the center of the cloud.

5.3.4. The Tag Cloud Plugin for Mayday

Tag clouds are a useful tool to visually inspect textual or other discrete meta information in gene expression studies. The integration of tag clouds in Mayday is based on the Mayday graph framework (see section 6.6) and is implemented as a plugin for the Mayday visualization framework.

The initial view presented to the user shows a tag cloud calculated from the size of the selected probe list. Tag clouds can also be generated from meta information groups that can be selected from the menu. To transform the occurrence counts of the individual tags, linear, logarithmic or exponential mapping is available. The mapped counts are then transferred to the font size range, which is 10 to 28 pt by default.

Each tag is displayed on an individual component. The component is painted as a box colored according to the frequency of the tag (see figure 5.7 for an example). The color gradient can be configured by the user. Each tag is associated with the probes that have this tag. These probes can be visualized within the tag component using a profile plot. For better visual comparison, if tag is selected, the selection is exchanged with all currently visible plots.

The layout of the tag cloud is also configurable. The tags can be ordered lexicographically or by tag frequency. A radial layout is also available. In addition to these strategies, the expression values of the probes can be used to induce a layout of the tags. This is done by computing a matrix consisting of the centroids of each tag's probes and calculating a principal component analysis on this matrix. Each tag is then placed at the position of its centroid reduced to the first two principal components of its centroid.

When using meta information as tags, the resulting tag cloud can become large. The tag cloud plugin can display a histogram of the tag counts to give an overview of their distribution. Filtering is also possible to remove the least frequent tags.

5.3.5. Interpretation of Chromograms and Tag Clouds

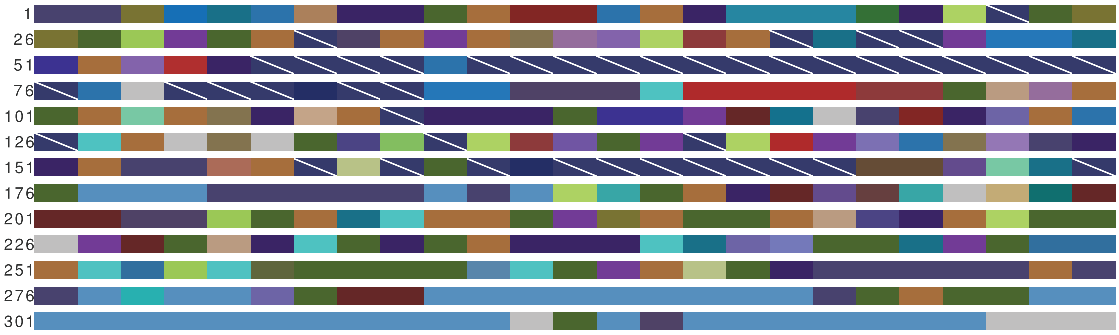
Chromograms and tag clouds both deal with textual annotations of objects, albeit in different ways. Chromograms visualize the annotations of (temporally) ordered objects, while giving no direct estimation of the frequency of a certain annotation. In contrast, tag clouds show the frequency of annotations while, in general, omitting any patterns present in the annotated objects. Especially with the interactivity possible in the Mayday implementations of both tools, they can be used together to perform powerful analyses of meta information and the primary data.

The combined use of tag cloud and chromograms is demonstrated on a set of variant genes in a time series in *Streptomyces coelicolor* [140], that were clustered using quality threshold based clustering [75]. In total 13 clusters of 4 to 147 genes and a set of 27 unclustered genes were computed. One chromogram and two tag clouds are produced. The chromogram is ordered by the maximum experiment criterion and shows the Sanger categories (`www.sanger.ac.uk`) of the genes. The order was chosen to find genes that have their expression maximum early or late during the time series. The first tag cloud represents the clusters sizes. The second tag cloud also shows the Sanger annotation. The resulting plots are shown in figure 5.8.

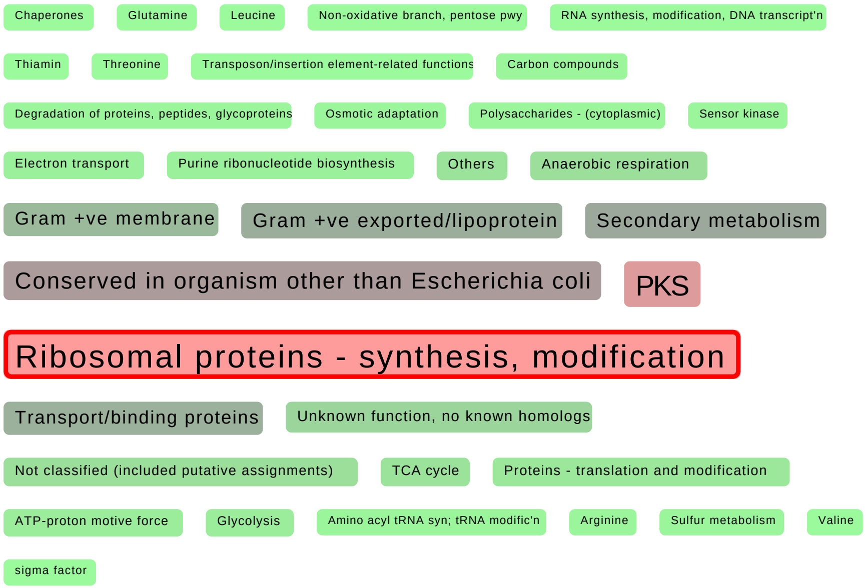
These plots allow to identify several interesting patterns. Genes related to ribosomal proteins all belong to the largest cluster and have their peaks at an early time point during the fermentation, as they are mostly located at the beginning of the chromogram in figure 5.8a (dark blue, also indicated by white diagonal lines). Further findings in this setup are that the PKS genes (related to polyketide secondary metabolites) are mostly expressed at the end of the fermentation. In fact, all the PKS genes that are represented by the long light blue sequences at the end of figure 5.8a are identical to cluster 10 (20 genes). Furthermore, most of them are genomic neighbors, as can be inferred from another chromogram ordered by locus tag (SCO5071 to SC05091 without SCO5082 and SC05083, not shown). Genes related to the secondary metabolism (scattered along the time series) and gram-positive exported/lipoprotein (maxima mostly at later time points) are often co-regulated with each other, or they are co-regulated with PKS genes, as can be inferred from the cluster memberships of the respective genes.

This demonstration used the experiment with maximum ordering for the chromogram. This is only one possible application. Genomic location or the overall expression level are also useful to identify interesting clusters here. In fact, with large or multiple displays it is no problem to display several chromograms and tag clouds in parallel.

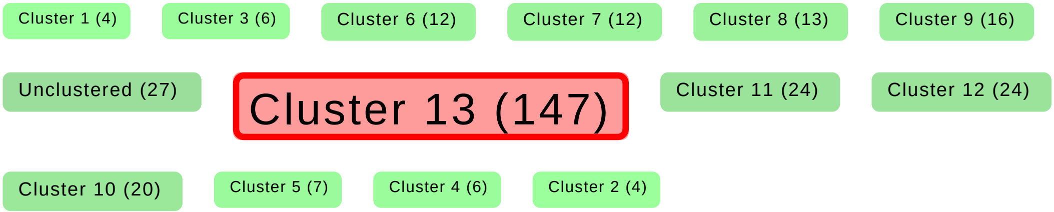
The above findings can be quickly inferred from the plots of figure 5.8 using interactivity. Users can browse tag clouds to identify tags that have interesting temporal (or other, see figure 5.9) patterns in chromograms. By selecting the genes in question, the respective tags are selected as well which allows, for example, to swiftly identify co-expressed genes. Vice versa, by selecting a tag, the chromogram shows instantly whether or not



(a) Chromogram of clustered genes


Chaperones
Glutamine
Leucine
Non-oxidative branch, pentose pwy
RNA synthesis, modification, DNA transcript'n
Thiamin
Threonine
Transposon/insertion element-related functions
Carbon compounds
Degradation of proteins, peptides, glycoproteins
Osmotic adaptation
Polysaccharides - (cytoplasmic)
Sensor kinase
Electron transport
Purine ribonucleotide biosynthesis
Others
Anaerobic respiration
Gram +ve membrane
Gram +ve exported/lipoprotein
Secondary metabolism
Conserved in organism other than Escherichia coli
PKS
Ribosomal proteins - synthesis, modification
Transport/binding proteins
Unknown function, no known homologs
Not classified (included putative assignments)
TCA cycle
Proteins - translation and modification
ATP-proton motive force
Glycolysis
Amino acyl tRNA syn; tRNA modific'n
Arginine
Sulfur metabolism
Valine
sigma factor

(b) Tag cloud of Sanger annotations



(c) Tag cloud of cluster sizes

Figure 5.8.: Chromogram and two tag clouds of 13 clusters of variant genes in *Streptomyces coelicolor*. All three plots are connected with each other. The highlighted tags represent tags that match the selected genes in the chromogram. (a) Chromogram with genes sorted by experiment with maximum, showing the Sanger categories of the genes; (b) Tag cloud of Sanger categories present in the genes; (c) Tag cloud showing the cluster sizes.

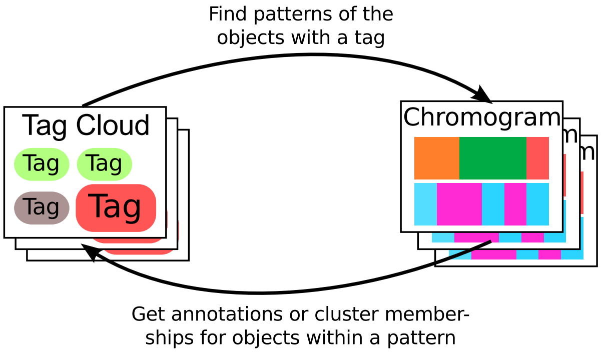

Find patterns of the objects with a tag
Tag Cloud
Tag
Tag
Tag
Tag
Chromogram
Get annotations or cluster memberships for objects within a pattern

Figure 5.9.: Interactions between tag clouds and chromograms that can be used to infer patterns in annotations and annotated objects.

there are any patterns in the annotated objects. This concept of connected views is a well established part of the visual analytics concept and is in no way limited to two plots. For example, to the view expression patterns of the genes along with chromograms and tag clouds, heat maps or profile plots can be used.

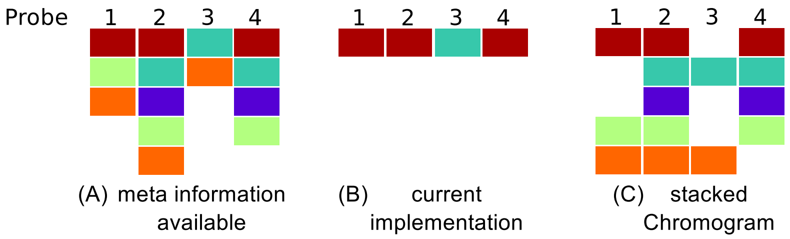
Chromograms exceed the capabilities of sorted lists of genes, which could also be employed for such a strategy, because they are far more compact. The above example shows 322 genes (see figure 5.8a), still distinguishable, even in such a small view. No current list implementation could perform this without converging to a vertical kind of chromogram.

The handling of string lists as meta information in chromograms remains a problem. The current implementation displays only the first elements of such a string list. The advantage of chromograms is that they visualize a long sequence of strings in a very compact way. Drawing multiple strings per probe might distort patterns. Drawing multiple horizontally stacked colors per probe would require more space per probe to be readable. This would remove a major advantage of chromograms. Using unevenly sized boxes to display would also reduce readability. For a limited number of meta information items per probe, a stacked view of constant size could be used for each probe. The order of the strings within the probes would have to be constant, which would also allow missing elements. Figure 5.10 illustrates the shortcomings of current chromograms and a stacked chromogram as a possible solution.

Further implementations of the tag cloud can be designed to exploit relationships between tags. The current design of the tag cloud plugin already allows edges between tags, e.g. for indicating relationships between tags to be displayed.

5.4. Term Pyramid

While tag clouds and chromograms, especially in combination, excel at exploring the landscape of gene or probe annotations, they cannot provide a comparison of the annotatons for two sets of genes. Tag clouds occupy too much screen area, and the ordering



(A) meta information available (B) current implementation (C) stacked Chromogram

Figure 5.10.: Illustration of a stacked chromogram showing up to five annotations for a set of probes. (A) lists of available meta information for 4 probes; (B) the current implementation of the chromogram shows only the first item for each probe; (C) a stacked chromogram showing all annotations for the probes.

of tags may be unstable. While chromograms are space-efficient, the ordering problem remains and individual annotations are cumbersome to compare due to the color encoding.

For comparative purposes, bar charts have been successfully used for more than 200 years ([188] discusses examples of early examples of such visualizations). Furthermore, it has been found that bar plots, that by design encode numbers using different points on the same scale, use one the most efficient and readable strategies for visual encoding data [205]. For comparative purposes, groups of pairs of bars can be used to display pairwise comparative data. *Population pyramids* use another approach. Besides providing an overall summary of the population structure of a country along with the age distribution, it also shows the differences of in size of the male and female population in a certain age quantile. This all is reached by showing two bar charts with the same base line, but with the bars pointing into two different directions (by convention, male to the left, female to the right). Similar charts are in general used for comparative purposes. The usage of two equal, but non-aligned scales has also been found to be a readable visual encoding for data [205].

The concept of a pyramid style bar chart for pairwise comparison of quantities can be used to compare the frequency of annotations in two sets of genes. The objects of comparison are annotation terms, with their frequency calculated from the count of terms for all genes in the respective set. The resulting plot is therefore called *term pyramid*. In the following, an implementation of the term pyramid concept is presented and a few notes concerning the interpretation of term pyramids.

5.4.1. The Term Pyramid Plugin for Mayday

The term pyramid concept is implemented in the term pyramid plugin for Mayday. It is based on the visualization framework for Mayday. As an input, the plugin takes a list of probe lists and a meta information group (of string or string list type). From this data, a configurable tabular view is created that shows the term pyramid.

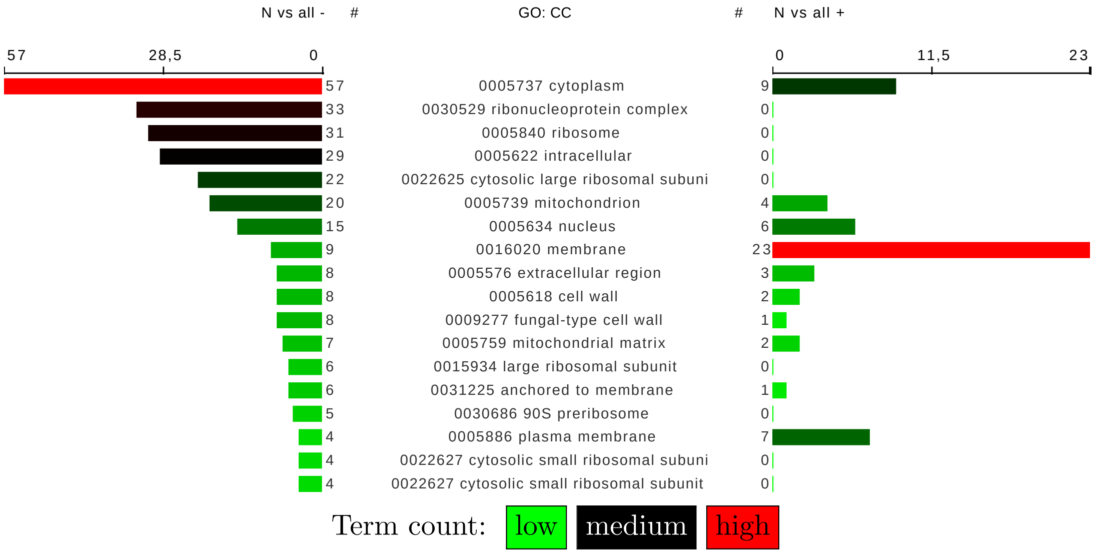


Figure 5.11.: Term pyramid showing the Gene Ontology cellular component (GO:CC) annotations in two sets of nitrogen-regulated genes in *Saccharomyces cerevisiae*. The figure only shows the upper part of the plot. The rows are sorted by the count of annotations in the left set. The coloring of the bar charts is based on the relative count of terms for each bar, using a green-black-red color gradient. Data from [103]

The term pyramid table has 5 columns (see figure 5.11). The central column shows the annotation terms, the outer two columns list the number of genes with this annotation and a bar chart that is scaled to the most frequent annotation. The probe lists used to generate the left and right columns can be configured via the main menu. Each column can be used to sort the term pyramid.

The rendering of the term pyramid can be configured in many ways. The coloring of the bar chart can be adjusted to match the color of the probe list, to the chromogram color encoding or using a green-black-red color gradient to encode the count of annotations. As an alternative view, the centroid profile of the probes having an annotation can be viewed using a heat map view (see figure 5.12a for an example). Several sparkline based visualization [190] are available. For interactive analyses, the term pyramid plugin communicates all selections to other open plots and sends and receives data transformations.

5.4.2. Interpretation of Term Pyramids

Term pyramids allow the swift comparison of the annotations of two probe lists. Similar probe lists with similar annotations would result in a very symmetrical term pyramid, with bars of the same length at each row. Dissimilar annotations give rise to a sequence of ordered bars on the one side, and an unordered set of bars on the other side. Example with somewhat dissimilar annotations can be found in figure 5.12b. Completely disjoint annotations give rise to a term pyramid with only one non-zero value in each row.

Term pyramids further extend the capability of analysing the annotation content of a set of genes. Comparisons of the amount of annotations of two gene sets are cumbersome using lists, and also the advanced tools discussed in the previous section have their strength in exploring the annotations of one set of genes. In contrast, term pyramids are specifically built for this purpose and use, beside a variety of options for visualizing either the abundance of terms, symmetry for the comparison of the annotations. Symmetry has been shown to be easily detected by human perception [13].

5.5. Probe Rank Plot

The concepts of the previous plots concentrates on discrete meta information, like gene annotations. As discussed at the beginning of this chapter, numerical meta information is also important. Tag clouds and chromograms rely on aggregation of textual meta information and perform poorly on numerical values. Discretization methods as discussed in section 5.2.1 can transform the numerical values to discrete bins, but a direct visualization of such data is often more useful.

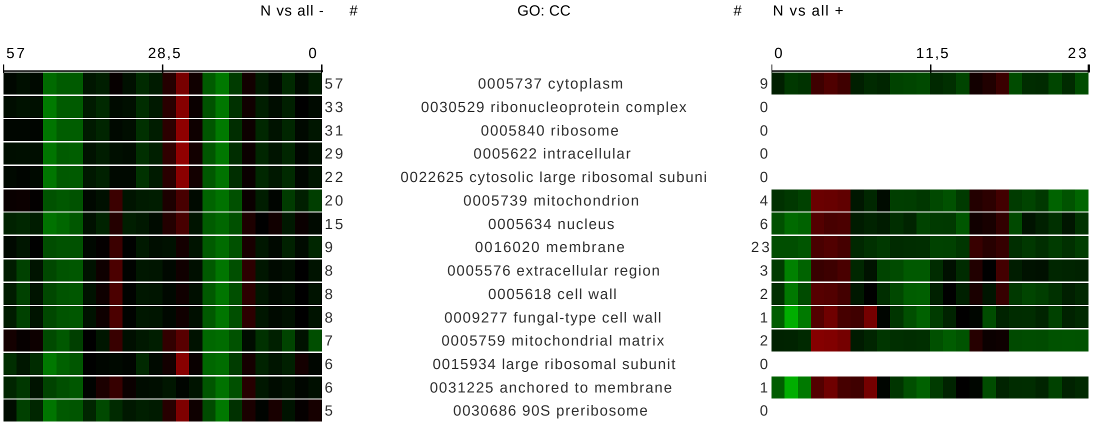
Standard visualization tools like histograms, bar plots and scatterplots can be used to visualize numerical values reasonably well. For the integration of numerical meta data with gene expression data, the enhanced heat map [63] exists.

For comparisons of numerical meta information horizontal bar plots can be employed, in a way similar to the term pyramid. Given a probe list and two different sets of numerical meta information, for each set the ranks are of the probes are calculated. Then, the ranks are visualized as two horizontal bar plots, called *probe rank plot*. The bars of both plots are arranged in the order of the first (left) bar plot. Alternatively, the ranking can be visualized in a tabular form. This allows to show the names of the probes and the actual numbers. A gradient can be used to strengthen the visual comparability of both plots. If the order of the probes with respect to the different values differ, this can be visually detected: The bars in the right bar chart are not sorted ascendingly, and the color gradient is not continuous.

This concept has been used for many applications, including the visualization of sorting algorithms (see `www.sorting-algorithms.com` for an interactive example). For gene expression analysis, the comparison of quality measures, summaries and variance measures and feature selection values can be of interest.

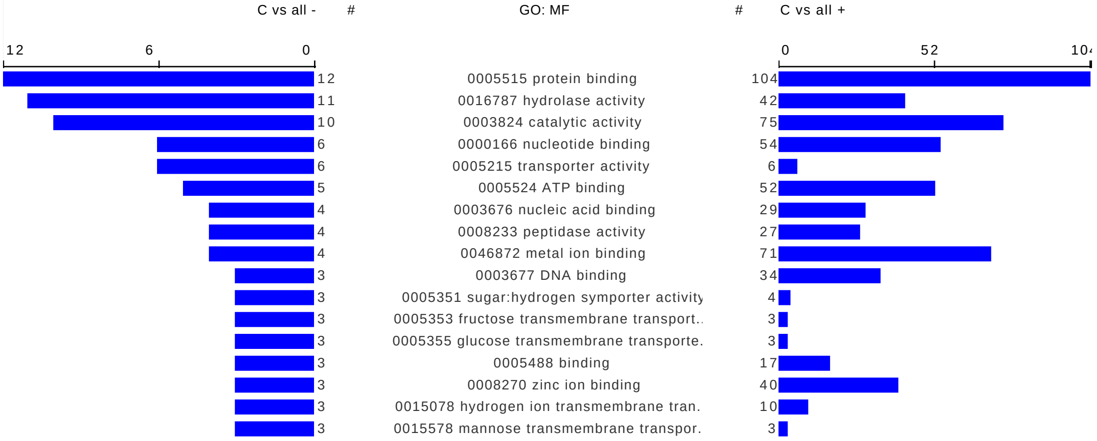
5.5.1. The Probe Rank Plot for Mayday

The probe rank plugin for Mayday implements the probe rank concept for expression data. It is based on the Mayday visualization framework and uses a customized `JTable` for displaying the bar plots. The plugin accepts as input one or more probe lists (only one can be shown at one time), and any numerical meta information. As described above, the left probe list is remains constantly sorted, while the order of the right probe list depends on the left meta information set. The alternative, more verbose, list view visualization



(a) Term pyramid showing annotations and centroid profiles of the annotated genes

Expression level: low medium high



(b) Term pyramid comparing clusters of carbon related genes

Figure 5.12.: Two term pyramids showing the Gene Ontology cellular component (GO:CC) and molecular function annotation in two sets of nitrogen (a) and carbon (b) regulated genes in *Saccharomyces cerevisiae*. Only the upper part of the whole plots are shown. The rows are sorted by the count of annotations in the left set. (Data from [103])

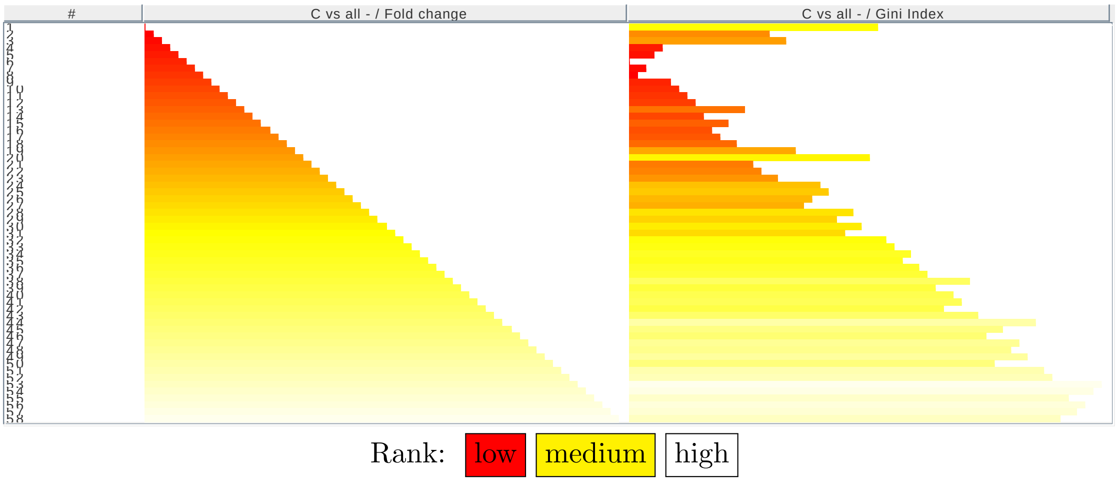


Figure 5.13.: Probe rank plot showing a comparison of the fold change and the Gini Index (discriminating between anaerobic and aerobic samples) of cluster of carbon regulated genes in *Saccharomyces cerevisiae*. A heat colors gradient is used to color the bar plot. Data from [103]

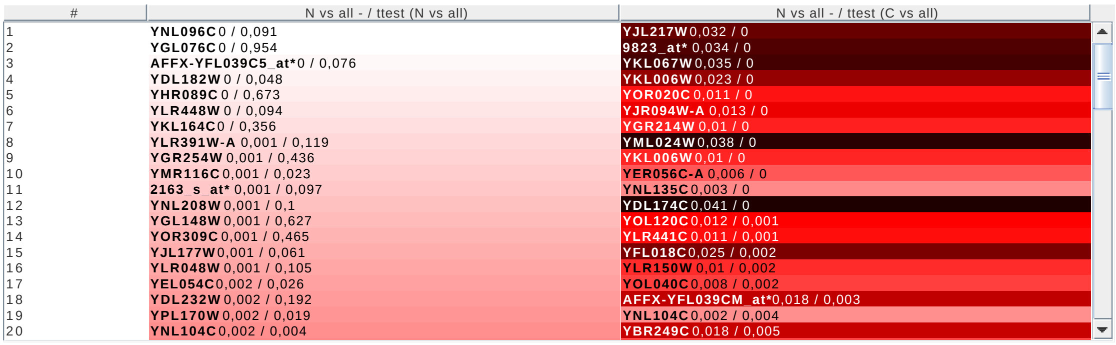
of the probe ranks with explicitly stated probe names and values is also implemented (see figure 5.14a). In both cases, a configurable color gradient can be used, with a heat colors gradient (red - yellow - white) as the default. Figure 5.13 shows an example, in which for a set of genes the fold change and the Gini index are compared. The example shows that there is in general a high correlation between these measures, however some outliers can be seen.

Comparisons of annotations between two probe lists are also possible (see figure 5.14b). In this case, only probes that occur in both probe lists are drawn as bars in the right bar plot. The other probes are not drawn explicitly.

The probe rank plot works well in combination with the GeneMining plugin for Mayday [158] and with the statistical test plugins. These plugins produce numerical meta information as an output. For further analyses of genes, the probe rank plot is interactive, selections in one of the bar plots are communicated to the other bar plot and to all other currently shown plots in Mayday.

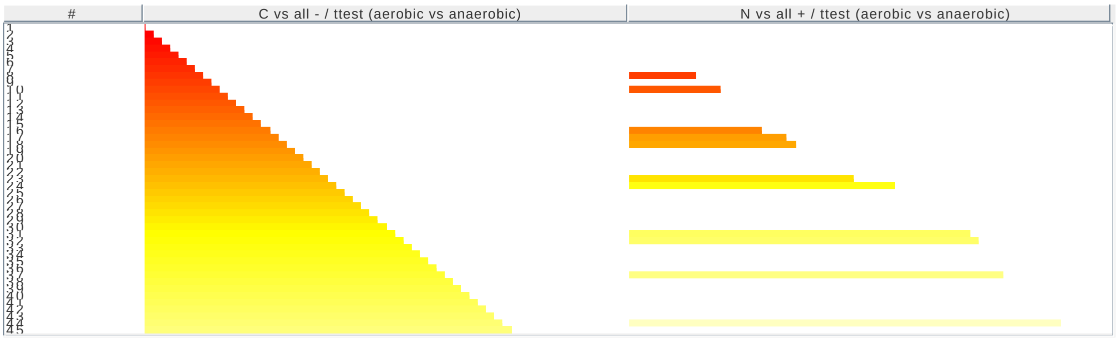
5.5.2. Interpretation of Probe Rank Plots

The interpretation of probe rank plots is straightforward. A mostly monotonously increasing bar length in both plots indicate that both meta information values have a very similar ordering. An example can be found in figure 5.13. Deviations from a similar ordering can easily be detected, as they introduce out-of-the-order bars in the right plot. Completely different meta information sets give rise to an unordered bar plot on the right side, which is also visible in the color gradient. An example can be found in figure 5.14a.



#	N vs all - / ttest (N vs all)	N vs all - / ttest (C vs all)
1	**YNL096C** 0 / 0,091	**YJL217W** 0,032 / 0
2	**YGL076C** 0 / 0,954	**9823_at*** 0,034 / 0
3	**AFFX-YFL039C5_at*** 0 / 0,076	**YKL067W** 0,035 / 0
4	**YDL182W** 0 / 0,048	**YKL006W** 0,023 / 0
5	**YHR089C** 0 / 0,673	**YOR020C** 0,011 / 0
6	**YLR448W** 0 / 0,094	**YJR094W-A** 0,013 / 0
7	**YKL164C** 0 / 0,356	**YGR214W** 0,01 / 0
8	**YLR391W-A** 0,001 / 0,119	**YML024W** 0,038 / 0
9	**YGR254W** 0,001 / 0,436	**YKL006W** 0,01 / 0
10	**YMR116C** 0,001 / 0,023	**YER056C-A** 0,006 / 0
11	**2163_s_at*** 0,001 / 0,097	**YNL135C** 0,003 / 0
12	**YNL208W** 0,001 / 0,1	**YDL174C** 0,041 / 0
13	**YGL148W** 0,001 / 0,627	**YOL120C** 0,012 / 0,001
14	**YOR309C** 0,001 / 0,465	**YLR441C** 0,011 / 0,001
15	**YJL177W** 0,001 / 0,061	**YFL018C** 0,025 / 0,002
16	**YLR048W** 0,001 / 0,105	**YLR150W** 0,01 / 0,002
17	**YEL054C** 0,002 / 0,026	**YOL040C** 0,008 / 0,002
18	**YDL232W** 0,002 / 0,192	**AFFX-YFL039CM_at*** 0,018 / 0,003
19	**YPL170W** 0,002 / 0,019	**YNL104C** 0,002 / 0,004
20	**YNL104C** 0,002 / 0,004	**YBR249C** 0,018 / 0,005

(a) Probe rank list view



(b) Probe rank comparison of two probe lists

(a) Rank: low medium high

(b) Rank: low medium high

Figure 5.14.: Probe rank plots showing different numerical meta information and probe lists in *Saccharomyces cerevisiae*. (a) shows the list view of a cluster of nitrogen regulated genes, annotated with different statistical test results. Here, an inverse heat color gradient is used; (b) shows a cluster of carbon regulated genes and the nitrogen genes from (a). Only a subset of genes from the left probe list is present in the right probe list, as carbon and nitrogen related genes have a small intersection. Data from [103]

When comparing two probe lists with respect to numerical meta information, only the probes that are in the intersection of both probe lists are drawn as bars in the right plot, however all the probes of the right probe list are used to sort the right bar plot. This shows both the intersection of the probe lists and the ranks of the common probes. Here, the comparison of the two rankings must take the possible gaps between two common probes into consideration.

It can be argued that the left bar plot is redundant, because it is always sorted. However, the left plot serves an important purpose, because it provides a standard for comparing bar length and color. Furthermore it sends and receives selections from and to the right bar plot, which helps navigating the plot.

Probe rank plots compare the ranks of numerical meta information sets. When however, the overall distributions of the values should be compared, then one should use

QQ-plots, scatterplots or histograms. Nevertheless, the ranks of the probes are very important in practice. Examples are settings in which the top 100 probes with respect to a feature selection criterion are to be found. Probe rank plots can give an overview of how the genes selected with respect to the first method would perform with respect to the second one. This can be applied when searching for genes that are stable with respect to different scoring functions.

5.6. Summary and Discussion

In this chapter, I have described and discussed five different plots that visualize gene expression data at several levels of abstraction.

The least abstraction is used for the profile logo plot. The profile logo plot uses a summary of an indirect representation of the primary expression data. This allows to compare analysis results like partitioning clusterings. It is also useful to inspect profile logos of genes grouped together by gene annotations. The different discretization strategies allow to pursue different aims when using profile logos, for example find experiments in which most genes are either up- or down-regulated. In addition, the TSD method allows to identify time points or series of time points with interesting features. The interpretation of profile logos is sometimes not straightforward, and heavily depends on the chosen binning strategy. Still, the examples show that they provide a very concise, but still readable summary of a set of gene profiles, especially when profile plots or heat maps become illegible due to the high number of probes. Furthermore, they present data in a very similar way as sequence logos, which are widely adopted in the research community.

Tag clouds and chromograms have been introduced as tools for text analysis and visualization. They both visualize short summaries of the texts, data or activities. These plots complement each other. Tag clouds show a quantitative summary of the tags. This is not done in the most space efficient way, but the different two dimensional embeddings of the tag cloud allow to adapt to several questions, as for different tasks, different embeddings are optimal. For most tasks, straightforward layouts are unsurpassed [122]. Chromograms provide a very concise summary of an ordered list of tags. This order is, in the case of gene expression data with annotations, not unique and may be customized to fit different purposes, but can make interpretation more complicated, if a non-trivial sorting pattern was chosen. Users with impaired color vision will have trouble dealing with chromograms. However, chromograms have the great advantage of being able to fit thousands of genes onto the screen without over-plotting and the necessity of scrolling. Together, chromograms and tag clouds can be used to interactively browse the landscape of annotations, and inspect their patterns with respect to various sorting criteria.

For comparisons of meta information between two probe lists, neither tag clouds nor chromograms are very useful, especially when temporal or other patterns are not of interest. In contrast, term pyramids provide bar charts of the frequency of annotations present in two sets of genes. This allows to answer questions like: “How do two clusters of up-regulated genes differ in the annotations of their genes?” Term pyramids can, by the way of sorting, find the most frequent annotations in both probe lists. By the way of scanning over the annotations, an overview of the similarity of both groups and terms that are differentially abundant between both gene sets can be found. For quick comparison of the expression levels of commonly annotated probes, several ways of adding expression summaries are available. For a through analysis of genes of interest, however, a full plot, like a profile plot or a heat map should be used.

Textual meta information is only a part of the whole meta information available in gene expression experiments. Therefore, the comparison of numerical meta data is also of interest. None of the above methods are particularly useful for this purpose. The comparison of the ranks of the meta information is visualized by the probe rank plot. This plot allows to compare the ranking of a gene with respect to two different meta information sets. Also the ranking of probes in two probe lists can be viewed. This somewhat extends the concept of the term pyramid plot, as it makes use of two bar plots placed next to each other. The fixed left bar plot serves as an anchor and makes finding specific probes easier.

Interactivity and communication between plots is crucial. The joint analysis of several sets of meta data is only possible if several plots can be viewed simultaneously, which is nowadays possible as large inexpensive screens and fast computers supporting them are available. However, the approach is more intuitive and potentially faster than devising complex filtering schemes for the same purpose.

All of the plots discussed in this chapter have not yet received much attention in the context of gene expression data visualization. This is especially true for the profile logo, and the tag cloud/chromogram combination. I am not aware of other implementations of this set of tools, at least not within a single working environment. Their implementations presented herein now allow an evaluation on their single and joint capabilities. The notes on the interpretation of the plots highlight some key assets and liabilities of the individual approaches.

A free and open source implementation of all five plots is available in Mayday. The implementation of the plots are all based on the very versatile Mayday visualization framework (see section 3.5). This provides interactivity, interoperability of different plots and high quality data export features with very little additional effort. Extensibility was prime concern during the implementation of the plots. Several extension points can be used to add features. Still, the implementations differ in some important points. The profile logo is based on an internal plot library used in several visualization plots in Mayday [12]. Extension points here are the binning strategies and the rendering strategy for the bin representation. The tag cloud is implemented based on the Mayday graph framework (see section 6.3), which provides a large number of extension points (see chapter 6 for details). For sake of simplicity, only a few of them have been used. The chromogram, the term pyramid, and the probe rank plot are based on the `JTable` component of the Swing library (Oracle, Redwood Shores, CA, USA). This component employs the model-view-controller design pattern and allowed a swift implementation of all the features required by the plots. In general, this strategy requires a specific table model, a table cell renderer that displays the information as desired, and a selection handling component that can communicate with the selection model of the Mayday visu-

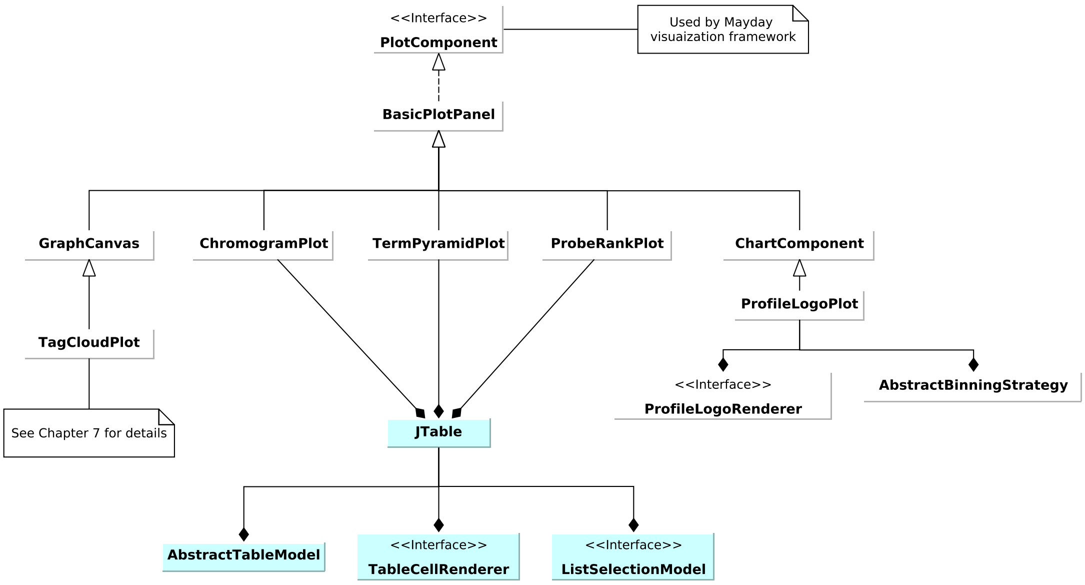


Figure 5.15.: UML class diagram of the five context-based visualization plots presented in this chapter. The diagram shows the overall implementation of the plots, as well as their principal extension points. Components that are part of Mayday are drawn in white. Light green indicates Swing components.

alization framework. In principle, the rendering components are exchangeable, however they are specific for their respective purpose and would not introduce useful features to other plots. Figure 5.15 shows a UML diagram with the overall structure of the plots, the means of their implementations and the principal extension points.

These different implementations have been chosen because they made the task at hand as easy as possible. Especially the table-based components were useful as the table component provides the overall structure of plots and is designed with several extension points in mind. Furthermore, they work fairly efficient, which is important when dealing with tens of thousands of genes.

Further development of these tools is possible, based on the extension points exist that allow to add new features like summarization or rendering methods. In this work however, this approach has not been pursued. While the five plots discussed here provide, especially in combination, a wide range of possibilities for data analysis incorporating meta information in the broadest sense, they are specialized tools for a limited purpose. The joint visualization of meta information and gene expression data should however be approached in a broader sense, with more general tools, that also provide a more direct view of the primary expression data. One such approach based on graph visualization is presented in the next chapter.

6. Graph Based Visual Exploration of Gene Expression Data

The field of visual data exploration has provided a set of powerful tools for data analysis. These tools are now routinely applied to gene expression data, which has lead to a large number of new insights. The complexity of gene expression data, with data from up to three different levels (transcript, protein, metabolites), and additional meta information, which can be equally complex (see section 5.1), requires strong analysis and visualization tools.

Several ways of dealing with meta information combined with expression data from a single source have been introduced in the previous chapter. The tools presented there proved to be useful, especially when combined. However, the entire complexity of meta information, especially of pathways and other biological models, is not covered by them.

To overcome these limitations, an approach based on graph visualization is applied in this chapter. The well-established field of graph drawing and the emerging field of visual analytics provide the background for this strategy. This chapter presents specialized and generic solutions for the visualization of graphs in combination with expression data and meta information. New applications and topics extending those presented in chapter 5 are discussed.

This chapter first discusses the combination of graph based visualizations with the visual analytics concept. Related work in this field is summarized in section 6.2. My approach to this problem is described in the following sections, starting with the description of the Mayday graph framework in section 6.3. The various options for displaying gene expression data on graphs are described in section 6.4. Tools based on these concepts are presented in section 6.5 where specialized pathway viewers are presented, and in section 6.6, where the concept, implementation and extensions of MGV – a generic graph viewer for Mayday – is presented. An important feature of MGV is the capability of combining gene expression data and meta information from different sources (datasets, studies, “omics”), which is presented in section 6.7. The key features are illustrated with various different experiments, along with notes on interpretability. A discussion of the approach, the implementation and the examples is given at the end of the chapter.

6.1. Exploratory Analysis of Biological Networks

The topic of this thesis is the visualization of gene expression data. The theoretical foundation, the methods and principles of data visualization, as well as their application for gene expression data were summarized in chapter 3. Based on these principles, in chapter 5 several methods were described which can be used to integrate gene expression data with meta information. The focus of these context-based methods were certain classes of categorical and numerical meta information. While these tools are appropriate for their purpose, they are specialized and generalize poorly. The visualization of both meta information and gene expression data is possible only using connected views, which requires a lot of screen space.

In every step of the analysis of gene expression data, visualization is used: it allows to identify quality issues, supports decisions on methods, summarizes the results and helps to structure the data and hypothesis. In higher level analyses, it becomes necessary to integrate meta information into visualizations. Meta information is often structured. Examples are biochemical pathways and interaction networks, or ontology based annotations. Neither of those can be adequately visualized in a table or a bar plot. Most common visualization plots cannot display network information. Network information, however, is important, as the full structure of pathways or networks contains more information than a list of their components. It shows the context in which gene expression occurs and in which a protein is active, especially for multi-omics data, where it is especially useful to visualize the process combined with the activity measurements.

Biological networks (see table 6.1 for an overview of common types) are of major interest. Research in systems biology yielded a large and extending body of knowledge rich both in breadth and depth. This knowledge is condensed and accessible in databases which describe organisms, processes, genes and chemicals in detail. This is the foundation on which systems biology works toward its ultimate aim: to comprehensively model biological systems. Therefore, especially biochemical pathways are of interest, as they provide a conceptual representation of biological knowledge, which is highly condensed and often aggregates many studies. Pathways are the elements of our understanding of basic biological processes. Their visualization is helpful for various purposes. While gene set enrichment analysis tools can identify the overall activity of a pathway, a drawing of the pathway enriched with expression data can give an overview as well as the details of the process.

The natural representation of biological networks are *graphs*. Graphs directly model interaction networks, where nodes represent proteins and edges represent interactions. A similar graph can be defined for reactions and pathways. Splice graphs [73] use this concept to represent different transcripts of a gene. In general, every association of two objects can be represented in a graph, for example a gene and associated objects of meta information. However, it has been shown that a Petri net representation [145] is a more useful way of describing biological processes. Here, place nodes represent metabolites, and transition nodes represent reactions. This concept is used for example in the SBGN standard [115].

Graphs represent a wide range of biological information and graph theory is a solid foundation based on which theoretical questions can be addressed. The field of graph drawing provides methods for visualization of graphs which can be seamlessly used for biological applications.

The visualization of gene expression data in this context is possible via charts on the nodes in a graph visualization or representing single values via node color and size (see section 6.2 for a summary of related projects). This can be achieved because in most

Type	Description	Sources	Formats
Pathways	Summary of a biological process	KEGG [91], MetaCyc [29], PathwayCommons [30], WikiPathways [144], Reactome [130]	KGML, BioPax
Interactions	Network of interacting biological components	UniProt [184], STRING [181]	PSI-MI
Reaction Models	Detailed model of a single process	BioModels [113]	SBML, CellML
Gene Models	Model of different transcripts of a gene	Ensembl [55]	GFF
Co-expressed Genes	Genes with similar expression profiles	Gene expression studies	

Table 6.1.: List of different types of networks occurring in systems biology. For each type of network, a short description of the content, as well as the most common sources and file formats are stated.

cases, the nodes represent measurable entities, like transcripts, proteins or metabolites. The same visualization tools as described in section 3.2 or chapter 5 can be potentially used for this purpose.

The feasibility of presenting multiple small plots in greater contexts has been demonstrated by *spark lines* [190], which show a small profile plot, or bar plot in an often textual context. Only a single data series is plotted in each instance. Applications are common in finance (stock rates), health and exercise. The technical foundation of spark lines is a medium that allows high resolution graphics. In general, a distributed visualization of gene expression data can be seen as an implementation of the *small multiples* concept, which is also well established. Commonly, a matrix of small profile plots, or bar chars (i.e. spark lines) are used to display multiple data series. Each individual plot displays only one, or very few individual data series using the same scale.

The aim is to allow the user to explore biological data, detect interesting patterns and find hypotheses. To this end, tools for data visualization in various levels of detail, for graph layout and for data integration are required. In general, biological networks are a tool for exploratory data analysis [97]. Statistical and visual summary tools and methods for extending and summarizing the graph can be applied. This is an application of the visual analytics concept which involves the usage of interactive visualizations combined with statistical methods to enhance human perception. Mining more details usually ensues, often including additional data or models. Integrative bioinformatics aims at combining data from multiple domains. For this purpose, rich data integration features are necessary.

A framework for visual analysis and exploration of high dimensional gene expression data and meta information in the context of networks will be presented here. The framework will be based on interactive graph visualization and will allow rich visualization of gene expression data and meta information presented on the nodes, as individual nodes, and in the form of graphs. Data integration – of expression data and networks – and clustering and summary tools for a variety of purposes will support this.

A further advantage of using graphs as the basic model of visualization is that this framework allows a large amount of flexibility. This will be exploited via associating gene expression values with nodes, which then can be used to induce an embedding of the graph. For an unconnected graph, this is a special case of a scatterplot. Thus, using probe information and meta information, as described in section 6.4 allows to produce high dimensional data visualizations. Furthermore, data from different sources can be included: Applications will be the enriched visualization of general biological networks, biochemical pathways, gene models and clustering stability.

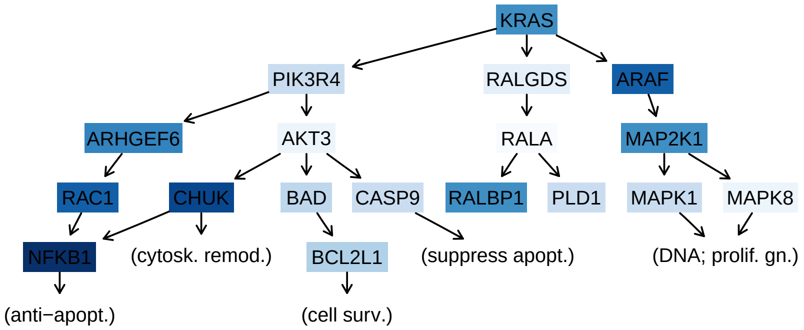
6.2. Related Work

With the theoretical foundation of this chapter presented in section 6.1, this section summarizes previous efforts in this field. Various concepts and tools have been proposed for the visualization of biological networks and gene expression data. Common applications are multiple purpose visualization tools, gene expression software analysis tools, biological graph analysis tools and pathway visualization tools.

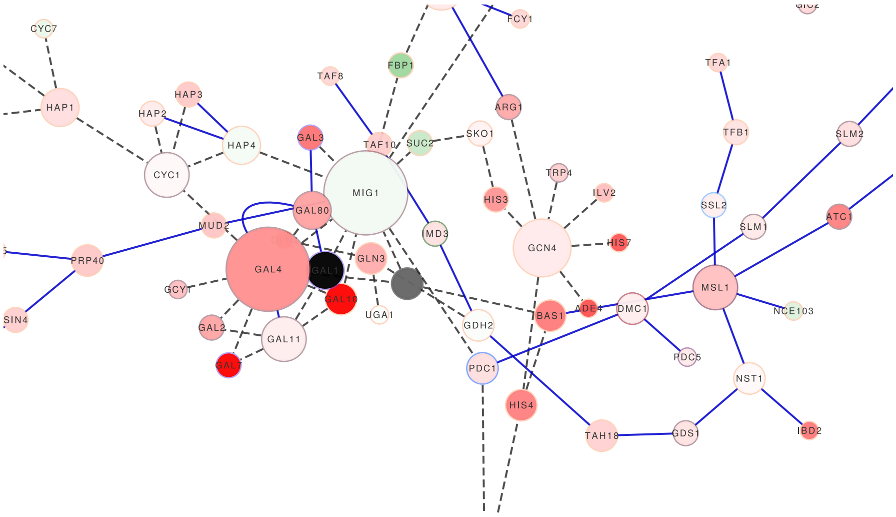
In principle, every visualization tool can be used for the visualization of biological processes. *Spreadsheet tools* can be used for this purpose. A better choice in most cases would be the BioConductor [64] suite for R [147], which features a vast choice of analysis and visualization methods. Analysis tools like sigPathway, and visualization tools like pathRender or KEGGgraph (all available via `bioconductor.org`, see figure 6.1a for an example), are based on the layout capabilities of GraphViz (`graphviz.org`). They provide powerful statistical methods, but are not interactive and visualization options are limited.

Graphical applications in this field, like GGobi [177], Mondrian [185] and others excel at the visualization of high dimensional data. They feature general visualizations like box plots, scatterplots, histograms and profile plots. These can be used to explore the expression profiles of genes that are part of pathways. This however comes at the price that the network information is lost, and as a consequence can only be inspected using another application.

Further into the direction of biological applications, *gene expression data analysis tools* have all necessary data integration features and provide specific data analysis and visualization features. Examples are MEV (Multi-Experiment Viewer, part of the TM4 suite [153]) or Chipster (`chipster.sourceforge.net`) which provide a wide range of visualization tools. The integration of network data still requires other tools. Partially, this is addressed by MapMan [186], a Java gene expression data visualization application. It is focused on plant model organisms, features some general purpose visualization tools and integrates pathway visualization by means of showing plots that summarize the gene expression or metabolite concentration of pathway components.



(a) Regulatory pathway displayed using the pathRender package for R.



(b) Regulatory network in Yeast shown in Cytoscape. Data: Cytoscape sample data

Figure 6.1.: Visualization of biological pathways in R and Cytoscape. (a) shows a small sample signaling pathway controlled by KRAS; (b) shows a sample interaction network from *Saccharomyces cerevisiae* in Cytoscape.

Biological *pathway analysis tools* are available in a wide range of functionality. A commercial product is the Ingenuity Pathway Analyzer (`ingenuity.com`). It is supported by extensive manually curated annotations and provides custom designed pathways, though it is limited to selected standard organisms. Cytoscape [168] is a popular free application for the analysis of biological networks. It allows the analysis of large graphs and provides many data integration features. Charts can be added to nodes by means of the Google Chart API (`code.google.com/apis/chart/`). Numerous plugins are available for analysis purposes. Data analysis is, however, no core feature of Cytoscape. VisANT [79] is a similar tool, however with more limited analysis and visualization features. The integration of experimental data is possible, but visualization tools are limited to profile plots. Vanted [85] integrates expression data and KEGG pathways and allows to build custom networks from them. Several graph-based analysis tools and layout options are available, as well as several node visualization tools and SBGN export. BiNA (Biological Network Analyzer) [98] provides high quality graph visualizations of biological networks. It stores networks and annotations in a database and can integrate expression data and visualize it via node properties. ONDEX [104] allows to create and integrate networks from multiple sources. It focuses on data integration from heterogeneous data sources and on using such data in a graph layout. GenMapp [154] is a Windows program that integrates expression data with community-created pathways. Expression data can also be integrated and visualized as small one-line heat maps. PathVisio [196] is an extended reimplementation of GenMapp in Java. Both tools allow to create user-defined pathways.

Web based tools for visualization of biological pathways are quite common. ProMeTra [138] visualizes biological pathways and other networks as annotated SVG files. These files are rendered with a visualization of measured data on annotated SVG objects. All elements to be displayed need to be prepared before visualization. KaPPA-View [187] is a tool for the visualization of metabolical networks of plants. It features limited visualization features and allows user-provided pathway maps. PathwayProjector [107] is a web application based on the Google Maps API (`code.google.com/apis/maps/`) that provides an overview of all pathways in the KEGG database. It concentrates on visualizing pathways. No data visualization or analysis is possible. Paintomics [62] is a web service that allows to map expression and metabolome data to KEGG pathways using a heat map based visualization. Several other tools use this strategy, see also section 6.5.1 for applications that specifically work with KEGG data.

In Mayday, all important visualization tools for gene expression data are included and can be used in a coordinated way. Aside from these, an implementation of a KEGG pathway viewer was available [44], which was used to inspect pathways resulting from gene set enrichment analysis. The pathway viewer was capable of rendering single expression values on nodes and featured a information bar that shows KEGG annotations of selected components.

It must be duly noted that the visualization of gene expression data within biological networks is nothing revolutionary new. Applications like Vanted and VisANT are to be credited for providing such features. There is a great range of visualization options in the pathway visualization tools, but even the applications with most options miss important visualization features. A complete visualization of gene expression and meta information is missing in all of the tools mentioned above. Such information, though, is helpful to make full use of the data. Nodes representing many genes are also rarely addressed (also only to a limited degree in Vanted). Furthermore, data integration methods are often lacking, or applications are limited to certain organisms. Concepts like exploratory data analysis rely on a multitude of visualization tools, multiple views of the same data and powerful analysis tools. Dynamic manipulation of graphs is also an important feature, still many of the above applications are restricted to fixed graphs. These features are generally rarely provided by the web-based applications, and neither are interactivity and summary functions.

6.3. The Mayday Graph Framework

This section describes the technical implementation of the graph visualization components for Mayday. Incorporating graph visualization tools into Mayday has advantages as well as requirements that arise from the architecture of Mayday.

Mayday already has a rich visualization framework, which allows to display gene expression data in many different ways, including those presented in chapter 5. It has proven to be powerful enough to support applications ranging from simple scatterplots to genome browsers that can display large eukaryotic chromosomes on single base pair resolution [180]. Mayday additionally offers a variety of tools for statistical and clustering analysis of such data, that could aid in graph based visualizations. On the other hand, a graph framework included in Mayday could be used for other applications.

An external graph framework that is to be used within Mayday must fulfill certain requirements. As a technical requirement, it must be compatible with the Mayday visualization framework and should be lightweight and implemented in native Java code for portability. Visualization features of such a framework are less important, as the compatibility with the Mayday visualization framework and the focus on gene expression and meta data visualization would require vast customization. Furthermore, the licensing must be compatible with the free and open source licensing used for Mayday.

For the implementation of the graph based visual exploration of gene expression data, Mayday has therefore been extended by a custom lightweight framework for graphs and graph visualization. The overall concept of the Mayday graph framework is described in the rest of this section. Specific extensions for gene expression data visualization are discussed in the following sections.

6.3.1. Overall Design

The conceptual basis of the *Mayday graph framework* is a directed graph $G = (V, E)$. To every node $v \in V$, edge $e \in E$, and G itself, a name $n_\bullet$ can be assigned. For the nodes and edges, additionally roles $r_\bullet$ that describe their biological meaning are stored. Edges can be weighted, w_e denotes the weight of edge e.

The core class of the Mayday graph framework is the class `Graph`, which models a directed graph G as defined above (see figure 6.2 for an overview of the model). A `Graph` object stores the graph name n_G and keeps an adjacency list $L : v \mapsto E_n \subseteq E$, with E_n being the set of directed edges that start at node $v \in V$. A list $L' : n \mapsto E'_n \subseteq E$ that contains the incoming edges of node v can be used for added performance in some cases. The use of an adjacency list is motivated by the observation that most biological networks, especially pathways and reaction models are sparse. The `Graph` class allows access and modification of nodes and edges and offers some convenience functions for searching nodes, calculating the degrees of nodes and removing unconnected nodes.

Both nodes and edges are explicitly modeled, because in biological networks both nodes and edges can be associated with explicit meaning and data. Nodes are modeled in the class `Node`. A node object stores n_v and r_v. Defaults for commonly used roles are defined in the utilities class `Nodes`. For convenience, if associated with a graph, a node can calculate degree and neighborhood functions directly. It is possible that a node is a part of more than one graph, which is useful for dealing with subgraphs. In this case, functions querying degrees or neighbors may return different results, depending on which graph is used as the basis. Several subclasses of `Node` exist: `DefaultNode` extends the basic node by the handling of a map of properties. It is further extended by the class `MultiProbeNode` that keeps a list of probes, which represent expression data. Edges are implemented in the class `Edge` $e = (s, t)$. They keep the source and target node s, t, the edge role r_e and have a weight w_e ($w_e = 1$ by default) and an optional name. Edges can keep a map of attributes that can be used to denote biological or processing details. Similar to the `Nodes` class, an `Edges` class exists that defines constants used by edges.

This design is complemented by a number of graph algorithms that are implemented in the class `Graphs` and an algorithms package. Currently available methods encompass, among others, the identification of connected components, strongly connected components, cycles and smallest cycles (via the Floyd-Warshall algorithm [56]). Standard operations like restricting the graph to a set of nodes or identifying root and leaf nodes in tree-like graphs can also be found in the `Graphs` class. A set of iterators allow to traverse the graph in depth-first, breadth-first, and, for acyclic graphs, in topological order.

Graphs can be natively serialized to GraphML [20]. Import of graphs is also possible from GraphML. Further import and export features are delegated to the applications and are discussed in the respective sections, especially in section 6.6.4.

The overall implementation of the Mayday graph framework is located in the Mayday core, and is used by the applications presented in this chapter as well as by other plugins for Mayday, namely the association mining plugin [165] and the graph based enrichment analysis plugin [99] for both analytical and visualization purposes. The code base is lightweight, encompassing about 4100 lines of source code. This generic graph data structure is the basis of the graph based visualization tools for Mayday.

6.3.2. Graph Visualization in Mayday

The Mayday visualization framework provides the base classes for data visualization in Mayday. This encompasses data structures, tools for online data transformation, color gradients and image export as well a common graphical user environment. The graph visualization tools presented in this chapter are all based on this general framework.

A *model/view* design was chosen to integrate graph drawing into Mayday. The view part of this approach is called `GraphCanvas`, which relies on models that extend the `GraphModel` class. `GraphCanvas` is the base class for graph visualization in Mayday. It is responsible for drawing the graph, receiving user interaction and communicating with the Mayday visualization framework. While it is used in some basic applications within Mayday, for more sophisticated applications it is subclassed and extended to meet special requirements.

The `GraphModel` class is the abstract basis of the models serving the `GraphCanvas`. For visualization and interactivity, each node $n \in V$ is represented by a `CanvasComponent`, a lightweight Java Swing component. The components are children of the `GraphCanvas` object. They also represent the screen embedding of the graph. This allows to have

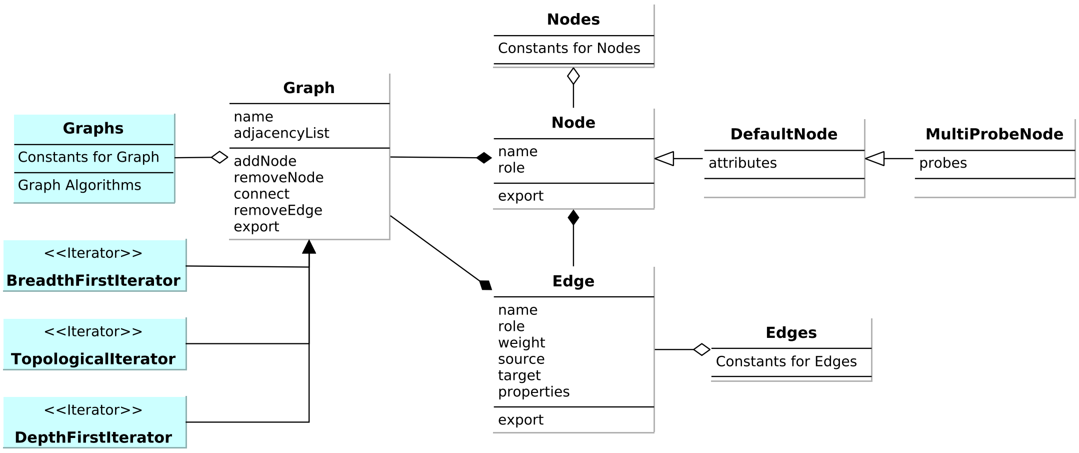


Figure 6.2.: UML Diagram of the core graph classes of Mayday. Key components are the classes `Graph`, `Node` and `Edge`. The subclasses of `Node` exist for different purposes and can keep annotations and expression data. The algorithms package (colored in blue) implements some of the most important graph algorithms.

several drawings for the same graph. The `CanvasComponent` class is also responsible for handling interactions with the node it represents. The `GraphModel` keeps track of the `CanvasComponent` instances and maps them to nodes.

A model serves as a front end for the graph it represents. All changes to the graph that arise during visualization, e.g. introduction of edges or additions of new graphs must be performed through the model, which informs by means of the listener pattern, all `GraphCanvas` implementations that represent it. Several `GraphModel` implementations exist for different purposes, for example mapping probes to components.

The layout of the graph is taken care of by a `CanvasLayouter`. The classes which implement this interface use graph drawing methods like force-based or hierarchical layout, or naïve methods to arrange the components on the canvas.

Drawing a `GraphCanvas` to the screen consists of four basic steps. For drawing, the default Java two-dimensional graphics framework is used. This provides a `Graphics2D` object that can be used to paint on. The following steps are involved:

1. Clear the drawing area and draw the edges. This is done by drawing the shape of each edge. The shape of the edges is determined by an `EdgeRouter` instance. Several styles for the stroke of the edge, and the color are available. Finally, arrow heads, edge names and weights are rendered.

2. Draw the nodes. The nodes are represented by `CanvasComponents`, which are drawn by the library method `paintChildren()`. The way the component is drawn is controlled by the `ComponentRenderer` instance responsible for it. See section 6.6.1 for details.

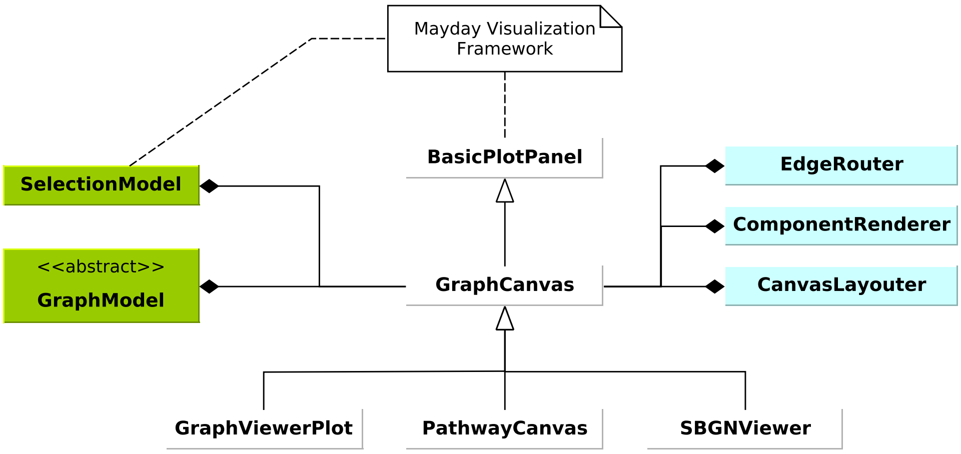


Figure 6.3.: Overview of the graph visualization classes in Mayday. The subclasses of `GraphCanvas` may use their specialized models and can handle rendering in a custom fashion.

3. Draw the node labels and other external decorating items for the nodes. This is done independently of drawing the nodes, to allow node labels that are not within the bounding box of the node.
4. If necessary, draw the selection rectangle, and an ellipse that highlights an important node or edge. The former is triggered by a drag selection event on the canvas, the latter is triggered by a call to a highlighting function.

This process guarantees that nodes are drawn over edges, and that labels and the selection rectangle can overlay both nodes and edges. The key components of edge rendering (`EdgeRouter`) and of node rendering (`ComponentRenderer`) are discussed in section 6.4. Many extension points exist in this framework. `ComponentRenderer`s, `EdgeRouter`s and `CanvasLayouter`s can be added via the Mayday plugin interface. Modular interfaces, as well as a rich library of supporting functions allow to swiftly extend these and other components. Based on this framework, three implementations of the Mayday graph framework are presented in this chapter.

6.4. Graph-based visualization of expression data

In this section, the options available in the Mayday graph framework for the visualization of gene expression data and meta information of various types within graphs are described. Note that graphs and networks representing biological concepts are also a form of meta information. Based on this observation, the tools presented herein are basically tools for displaying high-dimensional data.

This is the basis for presenting rich visualizations of biochemical pathways (the pathway viewers described in the following section however no not use all of the below

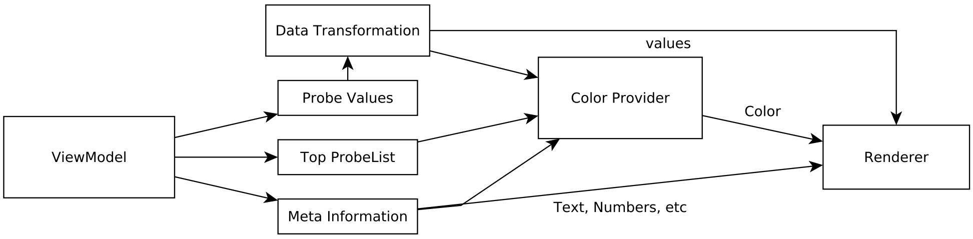


Figure 6.4.: Data preparation for node rendering. The view model represents the expression data including meta information. All information can be prepared by a color provider, which also handles color gradients. Expression data and meta information can also be rendered directly. Figure simplified for brevity.

options), and rich visualizations of arbitrary graphs, as provided by the Mayday Graph Viewer, which can use all of the options described below.

These tools intend to make full use of data visualization and graph drawing principles. The most important tool is the display of primary and meta information on nodes, which are discussed in the first and second part of this section, respectively. A summary of the edge rendering and graph layout options, which are also used for adding expression data and meta information to graphs concludes this section.

6.4.1. Options for visualizing nodes

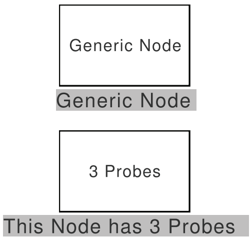
As systems biology data is diverse, many different types of nodes, which may or may not be associated with probes are necessary. The different strategies for displaying nodes and the renderers that implement them are discussed in this section. The renderers are grouped by the data typically represent.

In general, three types of data can be viewed in Mayday. Experiment (expression) data, which can undergo a data transformation, meta information, which is represented by color or text and the top priority probe list, whose color can be viewed (see figure 6.4 for the structure). The renderers that use probe information can use these three types, the details depend on the individual renderer. The color gradients used for this are configurable. In general, a single color gradient is used for probe data visualization.

Drawing Nodes

All renderers for drawing general nodes can be seen in figure 6.5. Nodes for which no experiment data is to be displayed, or that are without other properties that direct the rendering, are displayed using the default renderer. This renderer displays the number of probes present at the node, or the label of the node on a rectangular shape.

Some informal representations of the data require distinguishing nodes of different types. For this purpose, a set of different shapes can be used. If probes are associated with nodes rendered as a shape, optionally, an overall mean of the probes values can be displayed via the node color. Otherwise, a single, user defined color can be used.



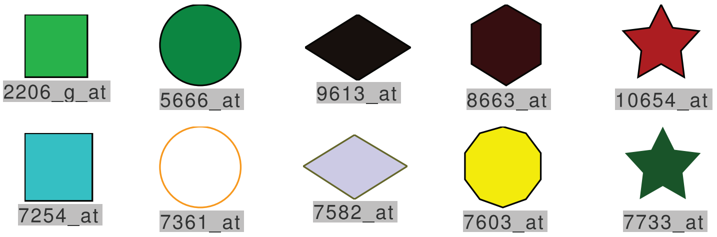


Figure 6.5.: Basic node renderers for different purposes. The nodes on the left are drawn with the default renderer that only draws a brief summary of the node. On the right side, a set of nodes with probes having different expression levels are shown with different renderers. The bottom line shows custom colored shapes.

Single Probes

Often, nodes are associated with a single probe, that can be visualized in different styles based on general plots for displaying numerical data (see figure 6.6). In the simplest case, a solid color can be used to display a single value of a profile (i.e. from one sample). The gradient renderer uses the selected experiment for this purpose. This concept seamlessly generalizes to visualizing all the probe values, giving rise to a “one-dimensional heat map”. Instead of encoding values by color, they are encoded by height in the bar plot renderer. It uses the bar color as an additional visual cue. Pie charts encode data using angle. While this concept is frowned upon in the visualization community, it has been notably used in the microarray analysis tool GeneSight (`biodiscovery.com`). Radial bar plots, which are also available for display, convey information better than pie charts.

Often, summary values are of interest, for example the fold change *fc*, as defined in section 2.5.2. The “Fold Change Renderer” uses a class partition to calculate the fold change of a gene between two conditions. It visualizes this data by means of an arrow, either upward or downward-pointing, flanked by the individual class means.

While small plots of data are useful, a minimum of space is required for probe data. Furthermore, a rectangular shape is preferable for most plots (except pie charts or radial bar plots). For this reason, the default size of most of these renderings is 80x50 pixels. This is sufficient for most visualizations, as can be seen in figure 6.6. The ratio of both sides is also close to the golden ratio. However, radial plots require a larger size to be effective.

Multiple Probes

Multiple probes at a single node can occur, for example, when working with probe lists or as a result of aggregating nodes. Basic visualizations of multiple probes can be created with renderers showing single probes. These renderers display the column mean of the expression value. To directly view multiple profiles, heat maps and profile plots are

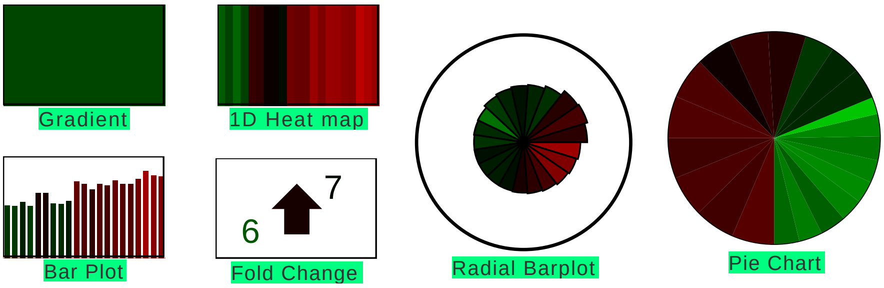


Figure 6.6.: Single Probe renderers available in the Mayday graph framework. The gradient renderer shows a solid color gradient, and thus only a single, configurable value. The fold change renderer summarizes the fold change between two conditions. The other renderers show all data points of a single expression profile.

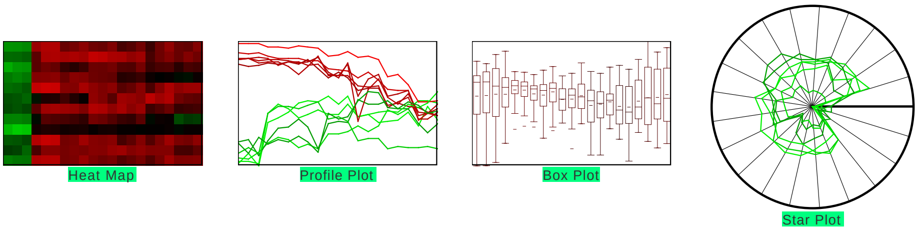


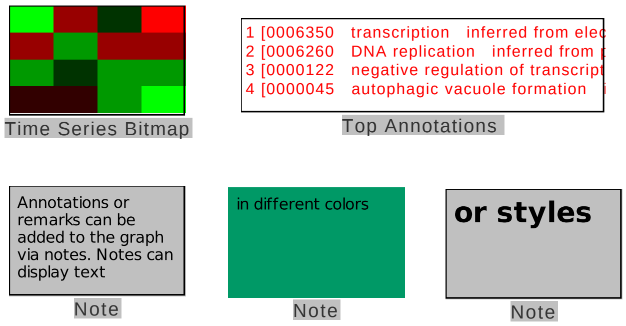
Figure 6.7.: Renderers capable of visualizing multiple probes. The heat map, profile plot and star plot renderers can also display single genes, while the box plot is optimal for visualizing the distributions of a large number of values.

available, as well as a star plot (also called radar chart), which extends the concept of radial bar plots to multiple genes. The renderers for this purpose are shown in figure 6.7.

For most applications, the efficiency of these plots is limited by the number of profiles. For large numbers of genes, where the individual profiles are of no concern (or are inspected in a connected plot), the profiles can be visualized as a box plot, which concentrates on displaying the overall distribution. However, it is not always desirable to visualize all data. Therefore, in the Mayday graph framework, it is possible to filter the probes: Instead of all probes, only a centroid (mean, median) or the median and the first and third quartile of the data can be displayed. This is transparent to the renderer.

Further Options: Time Series, Tags, Notes

Displaying summaries and either structured or unstructured meta information is the purpose of the renderers shown in figure 6.8. *Time Series Bitmaps* [110] are a method for summarizing and visualizing time series, based on the SAX [120] representation of the data combined with a chaos game representation of the resulting data. This technique allows to summarize long and complex time series to shorter sequences of symbols, which



1 [0006350 transcription inferred from ele
2 [0006260 DNA replication inferred from
3 [0000122 negative regulation of transcript
4 [0000045 autophagic vacuole formation

Annotations or remarks can be added to the graph via notes. Notes can display text

in different colors

or styles

Figure 6.8.: Renderers for additional information. The time series bitmap summarizes the profile of a time series experiment. The Tag list renderer shows the top annotations associated with genes. The nodes at the bottom show free-text annotations that can be displayed in different styles.

are represented as matrices drawn as squared heat maps. Each field in a time series bitmap represents a pattern in the time series, and the color indicates the frequency (bright green=0, bright red=1). The resolution of the heat map indicates the length of the patterns: a heat map with n rows represents patterns of length $\log_2(n)$. This feature allows to compare and visualize profiles with specific temporal patterns.

The meta information associated with a set of probes may of primary interest. The tag list renderer summarizes meta information associated with the probes of a node, and displays them as an ordered list, with the most frequent term on the top. This allows to include parts of the tag cloud concept (see 5.3) into graph visualizations, however it does not summarize the overall tags, but summarizes the tags present at one node.

Nodes with a textual note (a short description of the graph, of adjacent nodes, etc.) are displayed by the Note Renderer. This renderer draws plain text on the component that represents the node. Nodes can be configured to have different text color, background color and fonts. This is achieved by editing the node properties.

SBGN Process Diagram

Glyphs defined by the SBGN process diagram standard (see section 3.4.2) are rendered using the `SBGNGlyphRenderer` class. This class inspects the role of a node and renders the corresponding glyph as defined by the standard. This is applied for all entity pool nodes, and process nodes. Non-SBGN nodes are drawn by this renderer are as blue circles. The size of the nodes is chosen to be proportional to the other nodes. See section 6.6.2 for the display options for compartments and submaps.

6.4.2. Adding Meta Information

Displaying meta information together with primary experimental data is a prime concern of this work. For this purpose, the default rendering tools described above can be used, however, it is often necessary to view meta information as well as the primary

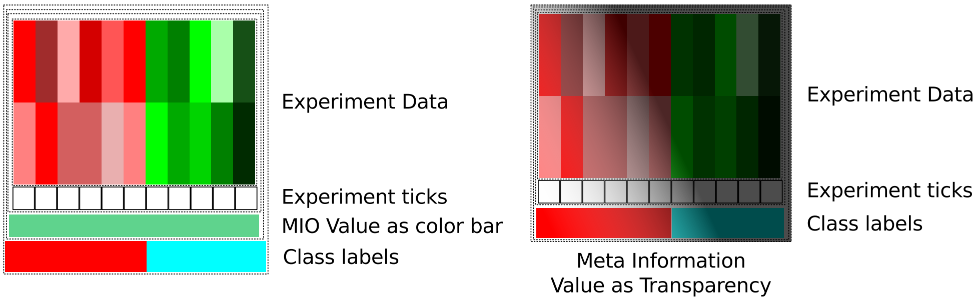


Figure 6.9.: Concept of renderer decorators. The left example shows a node associated with two probes and decorators showing experiment ticks, a meta information (MIO) value and class labels. The right example shows the same probes, without the meta information; but instead using transparency to display a relevance meta information.

data. This is achieved here using a concept based on the *decorator* design pattern [61]: Each of the renderers described above can be extended by one or more renderer decorators. These decorators have the same interface as the primary renderers, and keep a pointer to the next renderer or decorator. The painting is done recursively, with the first decorator drawing the outermost layer. Each decorator paints its part of the image, crops the drawing area, and recurses to the next decorator or finally the primary renderer, as illustrated in figure 6.9. Renderer decorators can convey different messages (see figure 6.10): They can add indicators to plots (ticks for experiments, or highlights for the currently selected experiment), show meta information and properties of nodes and probes.

Decorators for Displaying Meta Information

The meta information decorators available can display probes-based meta information as colored bars, or numerical values printed below the primary plot. Multiple probes are handled in the former case by displaying a horizontal array of colored boxes and in the latter case by averaging. The color bar approach is also chosen for the indication of class labels and the dataset membership of the node's probes. Textual annotations can be added that show rendering information, or probe and node properties.

For relevance information (e.g. statistical values), the nodes can be shaded using a decorator according to a meta information group that contains their variance. One possible strategy is overlaying nodes of low relevance with the opaque background color of the graph and highly relevant nodes with a nearly transparent color.

Auxiliary Items

Additional information, which can be added *ad hoc* or by filtering (see section 6.6.3) are displayed using auxiliary items. *Auxiliary items* can be added to any node, and a variety of shapes shown in figure 6.11 is available. These symbols add small visual

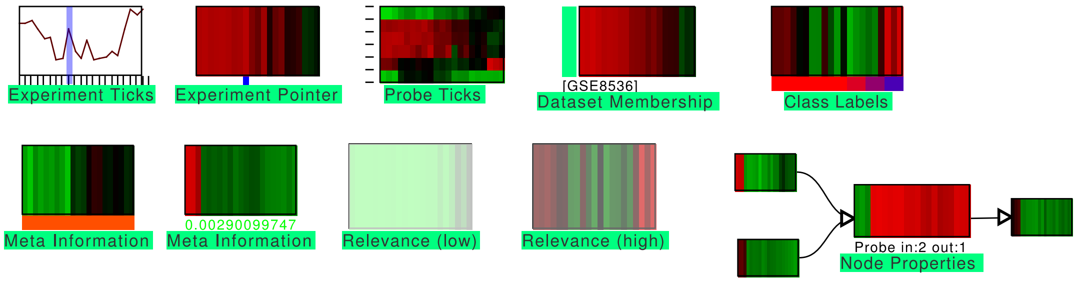


Figure 6.10.: Decorators add additional information to nodes. This figure shows decorators for adding tick marks to both axes of the renderings, as well as information about the dataset, the node and sample classes. Meta information objects can be displayed as numbers, colors, or used for shading the node.

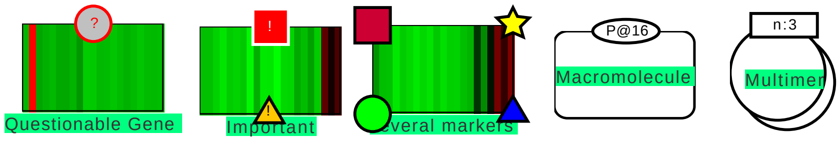


Figure 6.11.: Auxiliary items add small visual cues to nodes, like for expressing uncertainty, importance or generic signals. For SBGN state flags or units of information can be added to a node.

cues that express uncertainty ("?"), indicate important genes ("!") or add information encoded in four different shapes with configurable colors. State variables, which indicate e.g. modifications in macromolecules, and units of information, like the cardinality of multimers, which annotate SBGN glyphs are also rendered in this way.

6.4.3. Visualizing edges

The style in which an edge is drawn conveys the nature of the connection between adjacent nodes. For this purpose, the edge rendering can be configured in the Mayday graph framework according to roles of the edges. Edges can be characterized by three properties: The stroke used for drawing the edge, the arrow heads on both ends of the edge, and the shape of the edge. Examples of possible configurations are shown in figure 6.12.

The stroke and arrowheads usually define the type of the edge. Strokes can carry two messages. The type of the stroke (solid, dotted, dashed, see figure 6.12) indicates the meaning of the edge, while its width indicates the weight of the edge. The weight of an edge can also be indicated using the amplitude of a zig-zag stroke. The stroke can also

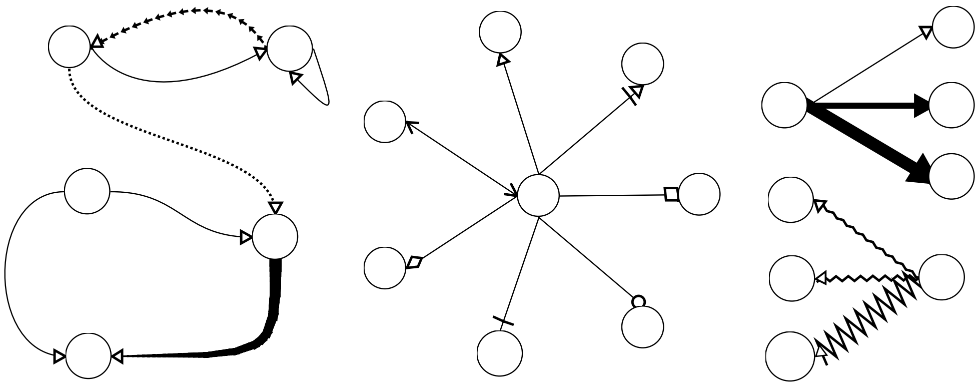


Figure 6.12.: Options for rendering edges. The left graph shows several options for drawing edges as quadratic and cubic Bezier curves, with different strokes. The center graph shows all options available for drawing arrow heads. The right graphs show two ways of representing edge weight: line width and zig-zag amplitude.

be used to indicate the direction of an edge, for example the tapered edge (bottommost edge of the left panel in figure 6.12), which has been found to convey this information very well [77]. Many different arrowheads indicate different interpretations for an edge, for example in SBGN, in which connecting arcs have defined arrow heads, while using a uniform stroke.

The shape of an edge is determined by a `EdgeRouter` instance. `EdgeRouters` create a path, either a straight line, or a quadratic or cubic Bezier curve. Figure 6.12 shows several examples for all three cases. The supporting points of the Bezier curves drawn are determined only by the relative position of the nodes. The rules for placing the support point(s) are encoded in the `EdgeRouter`. Bezier curves can be used to heuristically bundle edges.

6.4.4. Graph Layout

Much research has been done on the problem of graph drawing. Many methods have been proposed, an overview of them can be found in section 3.3.2. Both standard methods as well as naïve methods are available in the Mayday graph framework, the latter often using properties of the probes assigned to edges.

Common methods for graph layout are available in basic implementations, including a force based layout using the Fruchterman-Reingold method and the Sugiyama hierarchical graph drawing framework. From the JUNG (`jung.sourceforge.net`) graph package for Java, several methods including the Kamada-Kawaii algorithm, a balloon layout, a spring-based approach and a SOM-based layout are integrated. Interfaces for methods from the GraphViz and JGraph package are also available

A common task is the layout of graphs with $E = \emptyset$. In this case, common graph layout algorithms lead to unsatisfactory results. On the other hand, naïve methods often lead to good results. Aligning all nodes on a rectangular grid allows to employ sorting in order to arrange the nodes. Sorting criteria can be the properties of the probes or nodes. In a simple circular layout, in which all nodes are placed on the circumference of a circle, the nodes are automatically ordered by their in-degree or by probe properties. Linear graphs can be drawn either as rows, columns or as snakes. Some basic biological processes denoted in SBGN can be drawn based on simple rules, as they are linear or have specific topologies. These layouts are used for some data integration features (see section 6.6.4) as an initial layout.

Krzywinski *et al.* recently suggested for complex networks the so-called "Hive Plots" (`mkweb.bcgsc.ca/linnet/`), which use internal and external properties of nodes to build so-called "axes" of nodes. Axes are then are sorted and drawn, omitting the node shapes. Only edges between nodes from different axes are displayed. This method is extended here to building groups from nodes using different criteria for the creation and sorting of the groups and the sorting of the nodes within the groups. For this purpose, mostly the properties of the nodes' probes are used. Groups are created based on probe list or dataset membership, meta information and expression values. The role of the node and the degree (sources, sinks, inner nodes, unconnected) as in [210] can also be used. Sorting the nodes within the groups is possible by various criteria related to nodes and probes (for sorting of probes see section 5.3.2). The groups can be ordered by either their name or their number of nodes and are laid out using four different methods. Krzywinski *et al.* suggested the axis layout. Alternatives are as laying out groups circles, which can be arranged either on a grid, or as concentric circles, or in the form of rectangular groups, or as heuristically placed components laid out with the Fruchterman-Reingold algorithm.

The expression values of probes associated with nodes can be used to induce an embedding. This is done by calculating a principal component analysis of the expression values of the probes, using the the centroid if several probes are present at one node. Nodes are placed on the position of their probes, reduced to the first two principal components and scaled to the screen dimensions. The resulting embeddings often reflect the structure of partitioning clusterings.

As finding the layout with minimal edge crossings in a bipartite graph is NP-complete, heuristics are commonly used to arrange the nodes in each partition [49]. Here, for this purpose, the node degree or the number of probes can be used. The ordering can also be specified during the creation of a graph and therefore follow arbitrary criteria. Some of the data import features described in section 6.6.4 employ this strategy.

6.5. Pathway Viewers

Pathway visualization is a major tool in the analysis of gene expression data. Pathways condense much of the available biological knowledge of a process. Therefore, pathway viewers are important tools for understanding data, hypothesis generation and teaching.

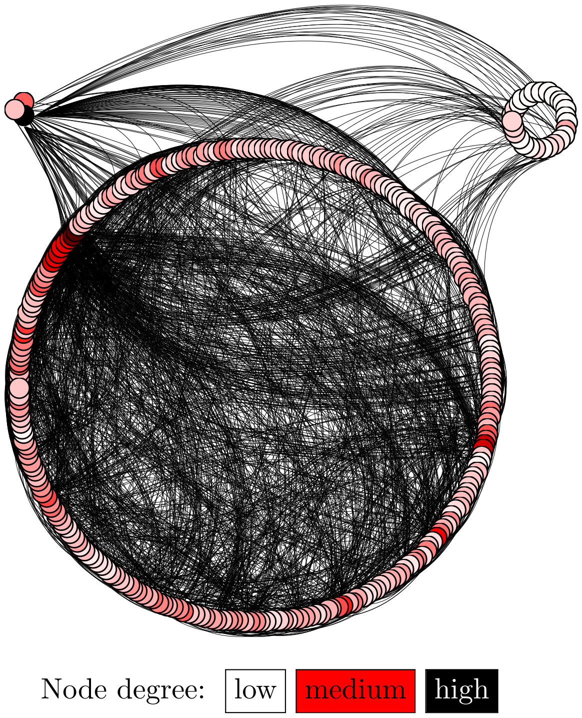


Figure 6.13.: Group based layout of the neural network of *C. elegans*, taken from [202]. Groups are input, inner, and output nodes. The nodes are colored by the overall degree, using an inverse heat colors gradient.

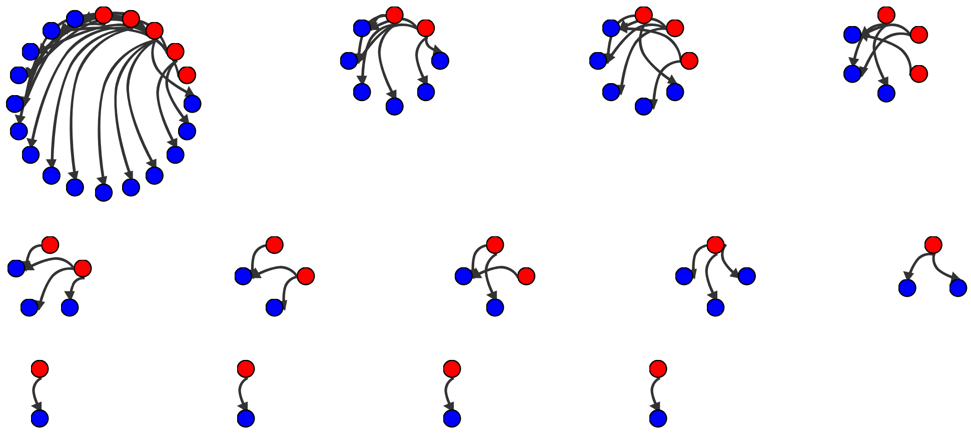


Figure 6.14.: Group based layout of transcription factors and controlled genes in *Saccharomyces cerevisiae*. Transcription factors (red) control clusters of genes (blue). Each connected component of the graph is laid out as a circle, ordered by the node degree. Data from [103]

This section describes two implementation of the options for the visualization of biological pathways described above, focusing on combining biological pathways with measured data and meta information. As the related work section indicated, there are limited options for visualizing biological pathways enriched with additional data. This context however is important to make sense of the data as well as of the pathway. For example, in an otherwise highly expressed pathway, a single key gene is down-regulated under some condition, causing the pathway's final product not to be produced. In order to detect this, neither single gene expression profiles, nor all the profiles related to a pathway are sufficient. The information necessary to explain the phenotype lies in both the pathway and the profile. Their joint visualization has the highest probability of triggering the discovery of the key gene.

Pathways are available in different styles and formats. In this section, tools for visualizing pathways from KEGG and pathways in the BioPax format, a standard format for biological pathways, are presented. This distinction is necessary, because these pathway sources are highly different. The tools presented in [180] and the following sections are optimized to the different properties of their data bases. While they serve a similar purpose, the resulting plots look different and the usage of both tools is different.

6.5.1. The KEGG Pathway Viewer

The *Kyoto Encyclopedia of Genes and Genomes* (KEGG) is a major source of pathways, which covers all aspects of a biological process. The KEGG database covers a wide range of topics, including genomic annotations, enzyme annotations (called "ko", KEGG Ontology), metabolites ("compounds") and reactions. This information is stored in user and machine readable files.

Using this information is common in bioinformatics and pathway visualization. Tools like KGML-ED [102], which concentrates on the manipulation of KGML files, Caleydo [174], which uses a 3D-box view to visualize several pathways and other plots in order to compare different datasets, and Dragon View which uses overlay graphics for displaying expression of enzymes [19] exist. Often such applications lack complete visualization features, or are restricted to a single organism. Furthermore, some tools do not exploit the rich database provided by KEGG.

In the application presented here, KGML files (see section 3.4.1) are parsed, represented as graphs and laid out based on the given coordinates. The resulting layout can be manipulated by the user and allows for larger nodes that represent enzymes and metabolites. The fact that KEGG provides highly organism specific pathways is used in the implementation presented herein: only the specific pathway components are shown.

Implementation

The *KEGG pathway viewer for Mayday* is implemented as a plugin for the Mayday visualization framework, and a subclass of the `GraphCanvas`. The implementation has all features of a Mayday plot, including communicated selections, configurable color gradients and online data transformations.

The KEGG pathway viewer requires as input a directory which contains KGML pathways and data files (ko, compounds). Furthermore, a taxon identifier (“hsa” for human, “mmu” for mouse, etc) is required to select organism-specific pathways. The download of pathways and database files is possible, but time consuming.

From this input, a configurable default pathway is imported from a KGML file. For easier mapping of probes to enzymes or metabolites, the ko and compounds files are parsed and organism-specific identifiers are extracted. Pathway components for which a mapping could be provided are annotated with probes. These probes are then displayed with one of the renderers listed above, allowing to display meta information as well. The pathway components are laid out as defined in the KGML file, with the components size adjusted to allow for rendering of probe values.

In the KEGG pathway viewer, nodes can be moved and resized freely. The nodes representing pathway links are interactive: a double click on such a link opens the target pathway of the link in the current window. This allows to interactively browse the landscape of pathways. As an alternative way of switching pathways, a menu listing all available pathways can be used.

Rendering options available allow to configure the rendering style of reactions and relation edges. In order to reduce the complexity of a pathway, the graph can be filtered for connectivity in different settings: either all nodes, the backbone nodes of the pathway or only key hubs can be displayed.

The nodes are marked according to their type: “E” denotes enzymes; a flask denotes a chemical compound; a link symbol denotes a pathway link. Tool tips are available that give more information about the nodes. A look up function allows to directly search for a node in KEGG, the result is shown in a web browser.

Examples

The KEGG pathway viewer for Mayday can be applied for two general purposes: gaining an overview of the entire pathway, and concentrating on the details of a pathway. Figure 6.15 gives an overview of the phenylalanine, tyrosine and tryptophan biosynthesis pathway of *Streptomyces coelicolor* as generated by the KEGG pathway viewer. The precursors of these amino acids are produced in several carbon metabolism related pathways (penthose phosphate pathway, glycolysis). The products and intermediates are themselves part of other processes, which highlights the importance of this pathway and gives a good starting point for exploring the many pathways referenced from it. Zooming in allows to study the expression levels for this pathway. For displaying gene expression data, the time series data from [140] was used.

Figure 6.16 shows the details of the inositol phosphate metabolism. The figure shows a switch of means of inositol production after the depletion of phosphate in the medium. The expression level of the *inositol-3-phosphate synthase* is decreased, while the *phospholipase C* expression is increased. Inositol is related to many metabolic processes, including streptomycin biosynthesis and other secondary metabolism.

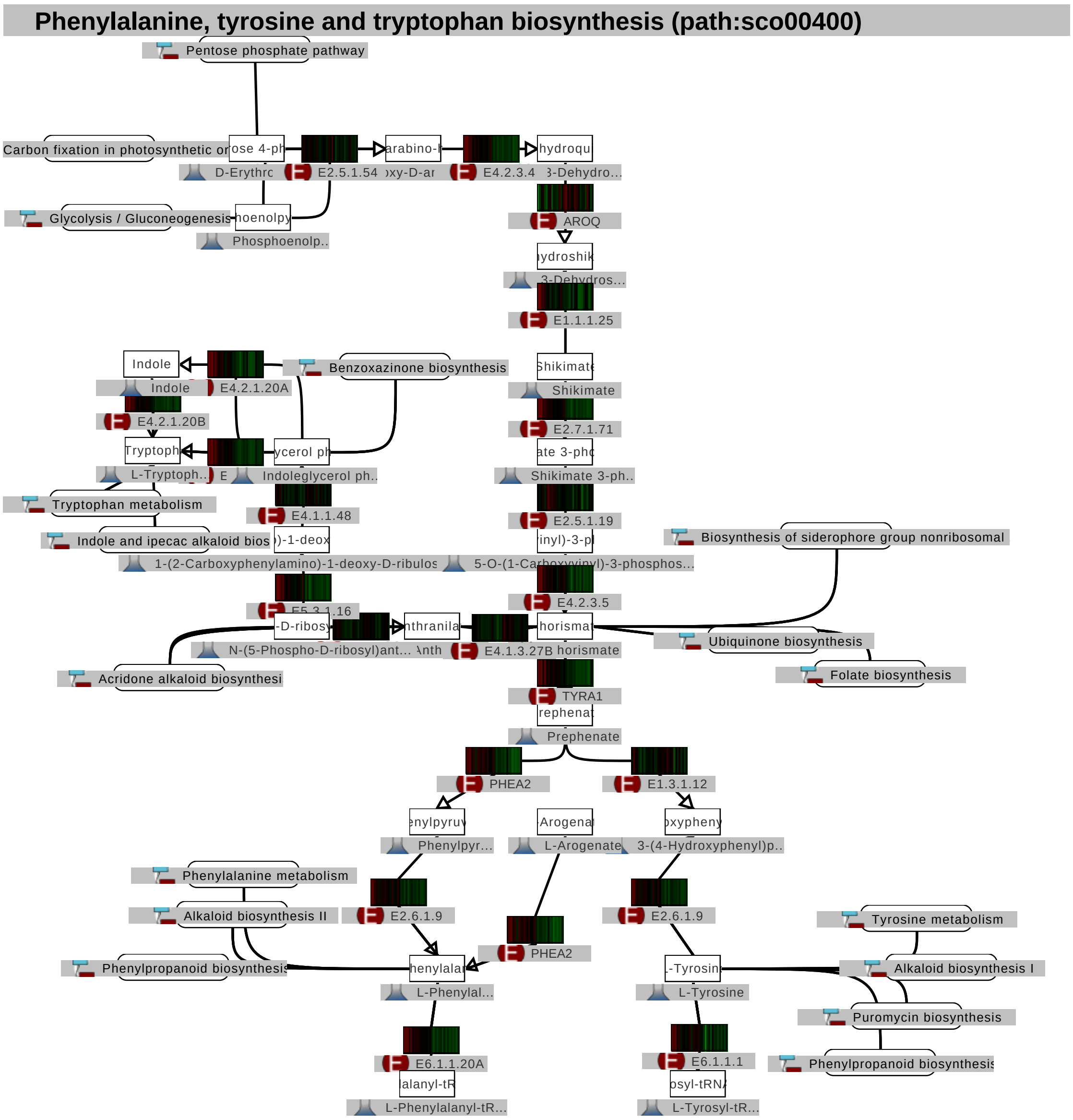


Figure 6.15.: Overview of the phenylalanine, tyrosine and tryptophan biosynthesis pathway for *S. coelicolor* from KEGG. The pathway is shown for *S. coelicolor*. Nodes are marked according to their type: "E" denotes enzymes; a flask denotes a chemical compound; a link symbol denotes a pathway link. The layout as defined by KEGG. Expression data from [140]

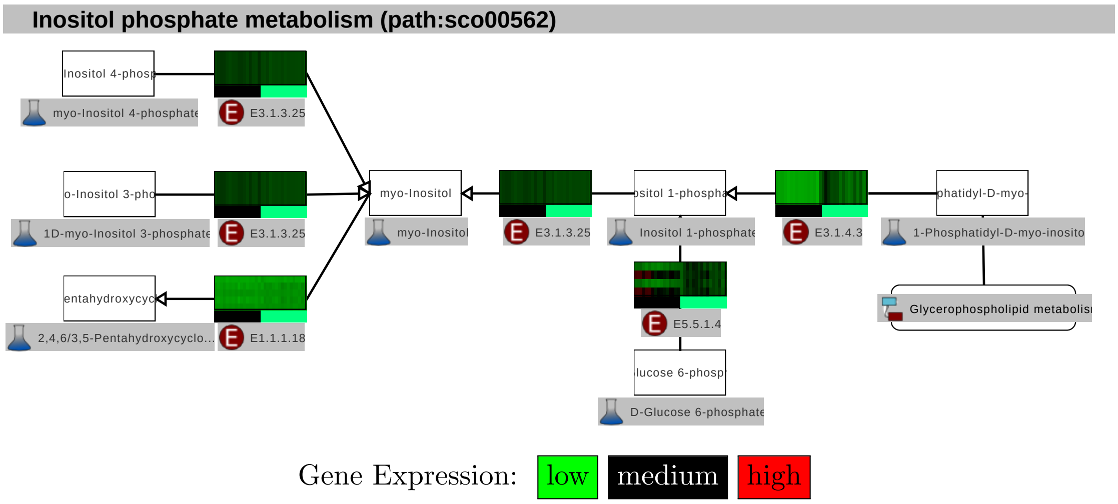


Figure 6.16.: Detail of the inositol phosphate metabolism of *S. coelicolor*. The gene expression is shown as heat maps rendered at each node. The data is taken from a phosphate limitation study in [140]. Nodes are marked according to their type: E denotes enzymes; a flask denotes a chemical compound; a link symbol denotes a pathway link.

6.5.2. The SBGN Pathway Viewer

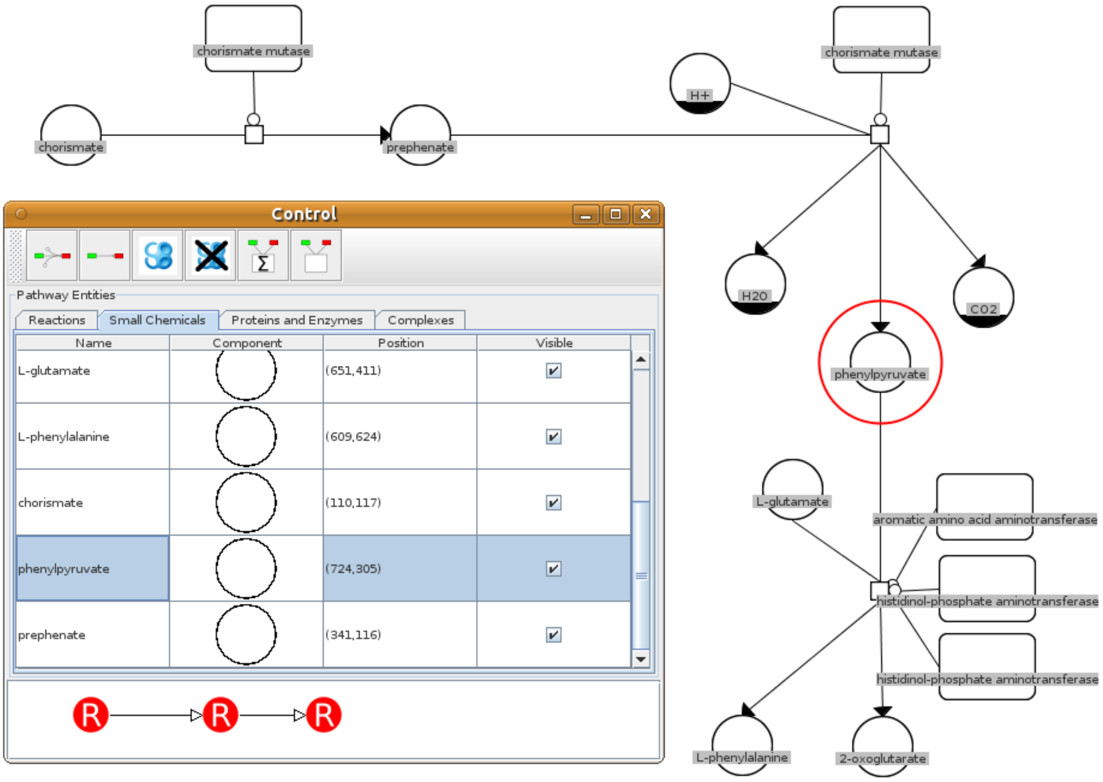
The visualization of pathways in Mayday is not restricted to KEGG pathways. Many databases provide pathways and other data in the BioPax format, which is modelled based on controlled vocabularies. Complement to BioPax is SBGN – the systems biology graphical notation (see section 3.4.2 which defines a set of rules for drawing pathways in an unambiguous way. The combination of both standards seems useful for conveying biological meaning.

Previously, this concept was mainly used for creating and exploring pathways, for example using CellDesigner [59], Athena [31], Cytoscape, and many other tools, a list of which can be found at `sbgn.org`. Biochemical modeling software packages have routinely adopted SBGN for their visualizations. Few further approaches exist for integration of gene expression data into such visualization. Furthermore, neither SBGN nor BioPax define rules for drawing expression values or metabolite abundances on pathways, which gives freedom to propose new methods.

To fill this gap, the SBGN pathway viewer [180] for Mayday has been developed. It complements the KEGG pathway viewer and allows to simultaneously view pathways in both formats.

Implementation

The implementation of the *SBGN pathway viewer* for Mayday is based on the Mayday visualization framework with all its advantages. As an input, it accepts a BioPax level 2 (`.owl`) file that contains pathway annotations. This file is parsed using a lightweight



Name	Component	Position	Visible
L-glutamate	○	(651,411)	☑
L-phenylalanine	○	(609,624)	☑
chorismate	○	(110,117)	☑
phenylpyruvate	○	(724,305)	☑
prephenate	○	(341,116)	☑

Figure 6.17.: SBGN pathway viewer showing the phenylalanine biosynthesis in *P. aeruginosa*. The figure shows the pathway diagram in SBGN, along with the control panel. The control panel shows a list of small chemicals, one of which (phenylpyruvate) is selected in both views.

BioPax parser which is converted into a graph. The components of the BioPax files are converted into nodes and edges with roles according to the SBGN standard. Annotations stored in the BioPax files are transferred to the nodes.

The layout of the graph is based on the recursive algorithm of Karp *et al.* [93]. In a first step, the backbone of processes and metabolites is determined, which is then recursively laid out as follows: Primitive graphs (linear, circular, branched) are drawn according to their type. For complex pathways, strongly connected components are identified and treated separately and reduced to a single node. The overall layout of the reduced graph is then performed recursively on the reduced graph. If a graph could not be further dissected without belonging to a primitive type, it is drawn using a force-based method. Based on this layout of the pathway backbone, side components are then placed heuristically around the reaction nodes. Alternatively, most of the layout methods described in section 6.4.4 can be applied.

By default, the pathway is drawn as a SBGN diagram. Nodes are freely movable and resizeable. Figure 6.17 shows the chorismate degradation pathway for *Pseudomonas aeruginosa*. A control panel allows to inspect the lists of the chemicals, proteins and

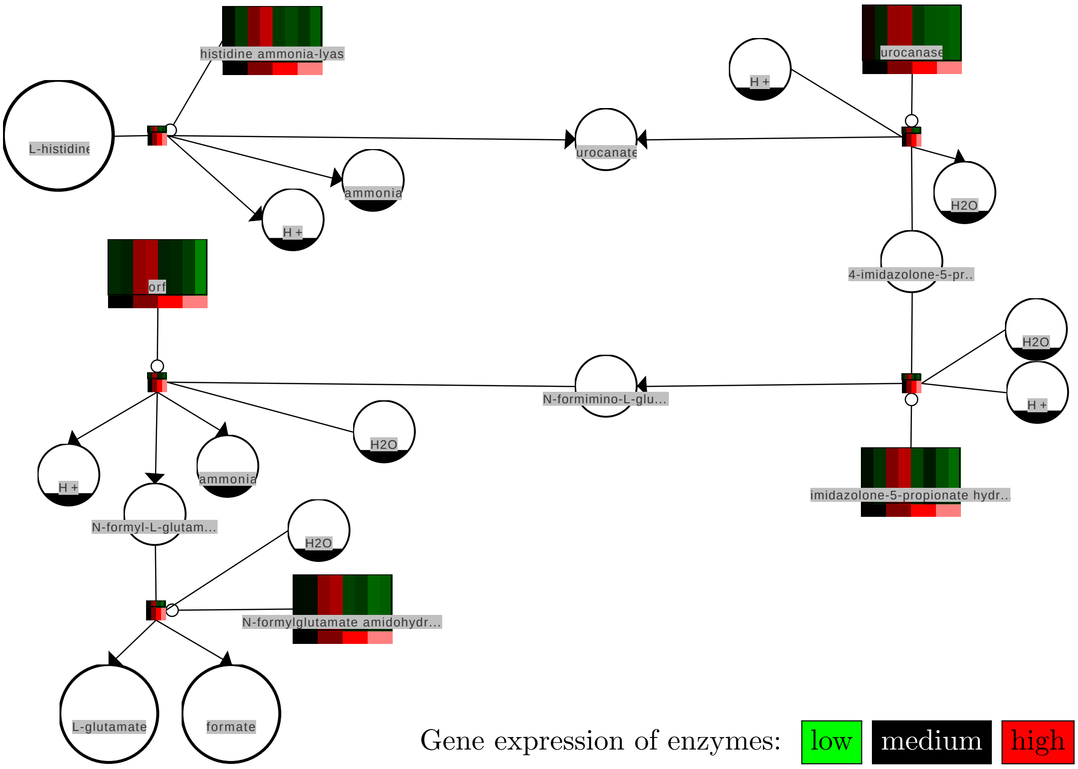


Figure 6.18.: SBGN pathway viewer showing the histidine degradation in *P. aeruginosa*. The gene expression levels for the enzymes are shown using a 1D-heatmap. The remaining components are drawn as SBGN glyphs. Principal input and output substrates are drawn enlarged. The growth conditions are color-encoded: black. no oxygen; shades of red, 0.4% (darkest), 2%, 20% (lightest). Data from [3]

reactions, and shows an overview graph of the pathway. The lists and graph are interactive and can be used for navigation in the pathway. In order to concentrate on the key aspects of the pathway, users can deactivate all side components (e.g. enzymes). To view expression data, the SBGN view can be deactivated. This allows all renderers described above to be used, including decorators for adding meta information. Often, reactions are catalysed by multiple enzymes. To give a summary of the reactions activity (estimated via the expression values of the enzymes), a mean profile of all enzymes can be shown at the reaction.

Application

The SBGN pathway viewer has been used to investigate metabolic effects of anaerobic and aerobic growth of *Pseudomonas aeruginosa* [3]. A first step would be exploring a pathway. Figure 6.17 presents the tools provided for this. The pathway can be viewed on the conceptual level (in SBGN process description style). The controls window assists in finding relevant elements and shows the overview of reactions. The pathway displayed is the chorismate metabolism of *Pseudomonas aeruginosa*, as found in MetaCyc [29].

Inspecting the details of a pathway, including enzyme expression levels, is the next step. As an example, 6.18 shows the histidine degradation pathway in *P. aeruginosa*. The visualization of class labels along with expression values allows to swiftly see that the expression of all enzymes is lowest in the anaerobic samples, and highest in the microaerobic (0.4% oxygen) conditions. Further investigations could include finding similar patterns and attempting to explain this pattern. Possible causes can be low overall metabolic activity in anaerobic conditions, high demand for glutamate or formate in anaerobic conditions and high demand for histidine in aerobic conditions.

6.6. MGV - The Mayday Graph Viewer

The pathway viewers are a first implementation of the principles described at the beginning of this chapter. However, a more generic application to visualize gene expression data in the context of biological networks is provided in this section. Here, *MGV* – the *Mayday Graph Viewer* – is presented, a generic graph viewer for Mayday [179]. It aims at integrating a wide range of biological networks, both induced by gene expression data in Mayday and from external sources. The full range of expression data and meta information visualization tools described above can be used. To provide an implementation of the concept described at the beginning of this chapter, extensible tools for editing, expanding and condensing graphs are provided, in addition to data integration methods for a wide range of biological models. Figure 6.19 illustrates the scope of MGV.

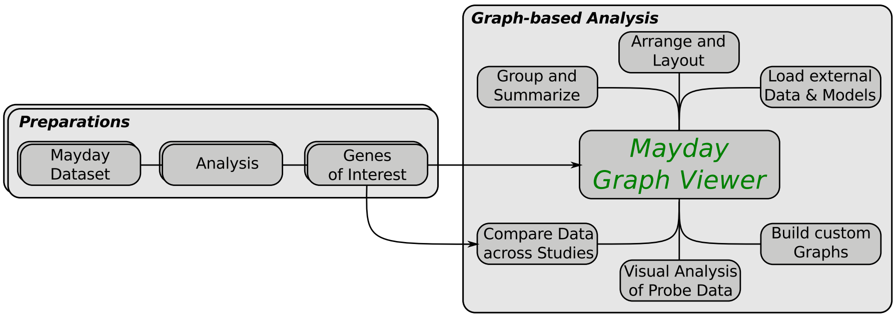


Figure 6.19.: Scope of MGV. MGV is a tool for the higher level analysis of data and integrates many high-level data visualization features.

6.6.1. Implementation

The Mayday Graph Viewer is implemented as a plugin for the Mayday visualization framework. It allows to visualize generic graphs, with the nodes annotated with gene expression data and meta information. MGV can be used to create, view and edit graphs and can perform filtering and analysis operations on them.

MGV is extensible by plugins which implement methods for data integration, for extending, structuring and summarizing the graph, editing its layout, and other actions on the graph. All plugins extend the class `AbstractGraphViewerPlugin` and have access to the `GraphModel` as well as to the graph canvas. Several categories of plugins exist for different purposes. Depending on the plugins, they can be run from the main menu of MGV or be invoked by a key short cut. For graph layout and arranging nodes, the tools described in 6.4.4 or plugins for arranging and sorting nodes by various criteria can be used.

MGV is designed to have a rich user interface for easy handling of graphs. Drag and drop of probe lists from the Mayday main window into MGV is supported, as well as copying and pasteing of nodes, also between different MGV instances. For swift working with graphs, selected features can be invoked using key shortcuts (see appendix A). MGV provides three overview functions for the graph. The node inspector gives a list of the nodes and edges in the graph. The structure inspector shows a hierarchically structured view of strongly connected components, connected components and nodes. The edge weight histogram displays the overall distribution of the weight of all edges.

For the easy integration of external information in MGV, a number of options are available to look up objects in external databases. If a node is annotated with a database entry, it can be automatically looked up via the “Look Up” function.

MGV supports graphs that represent probes from multiple datasets (see section 6.7 for details of the cross-dataset visualization and analysis features). Visualization of probes from different datasets is possible without preparations. Many plugins are designed to handle heterogeneous data, but some are limited to probes from single datasets.

For displaying time series data, MGV can use animation. Two plugins, called “Expression Movies”, are available for this purpose. Expression movies are visualizations of probe data in a serial way. They display one experiment after the other for a certain amount of time, either adjusting the node rendering or the node size for each experiment in the dataset. The former case is best used with a solid color gradient for probe data visualization, or with a highlighting decorator. The latter case scales the nodes in a defined range according to their value. The frame rate of the movie can be adjusted. Manually controlling the movie is also possible.

Rendering of Nodes and Edges

MGV implements all features for data visualization described in section 6.4. A node is rendered using a primary component renderer, which dictates the shape of the node and is responsible for data visualization. Additionally, decorators can be used that add additional information.

The assignment of *rendering strategies* to nodes requires multiple steps. MGV keeps track of a list of roles to which a certain rendering strategy is assigned. Such a rendering rule is defined as a the combination of a target role, a primary renderer and an ordered list of decorators (see figure 6.9). Rendering rules can also be assigned to nodes individually.

When a node is to be rendered, the correct renderer is chosen by first checking whether there is a customized rendering rule for this node. If not, the role of the node is used to render the node. If no such role is defined, a default renderer is employed to paint the node. If a decorator should be used at all nodes, it can be added to the list of overall decorators. They are applied to all nodes, in addition to their normal rendering. strategy. If a node is too small to be rendered in a legible way, the node is drawn as an ellipse, regardless of the rendering rule. A number of roles, especially for SBGN glyphs and Mayday objects are predefined, and custom roles can be added to adapt to new data.

A short cut for switching between different rendering strategies can be used. By pressing the space bar, the next of a series of predefined, but customizable rendering strategies (different plots and SBGN only) can be activated.

The same role based strategy with customization and a fall-back solution is used for edges. A set of predefined rendering rules exist that adapt to common use cases of MGV, including SBGN arcs. Overall settings include the rendering of the edge names and of the edge weight as labels for the edges.

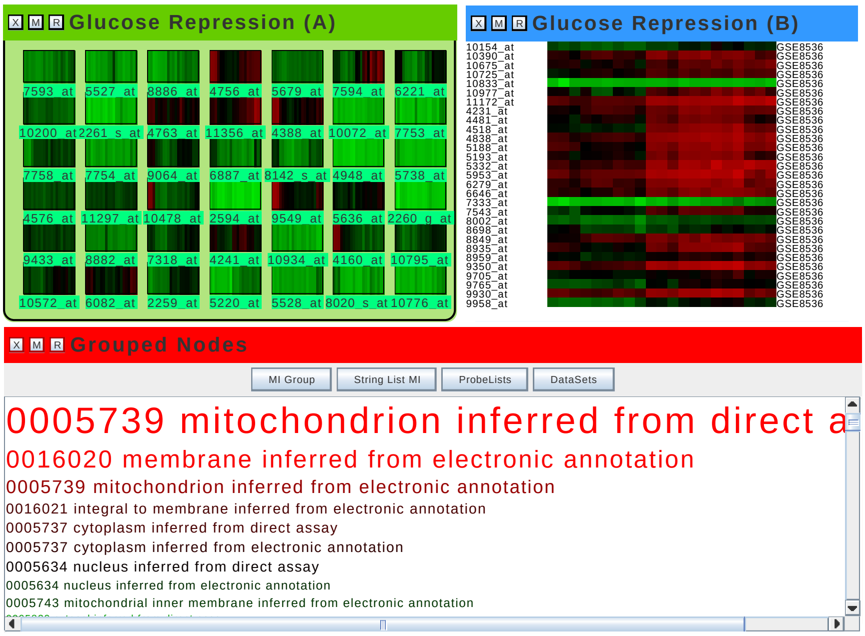
Edges in biological networks are often weighted, however the order of magnitude of the weights can differ greatly. Often, the inverse (for distance measures) or just the order of magnitude may be of interest. MGV can, for this purpose, transform (invert, $\log_2$, $\log_{10}$) or scale edge weights. Furthermore, a filter can be applied to remove edges that have a weight below a threshold.

6.6.2. Node Groups

It is often necessary to group a number of nodes. This can be done in order to indicate that these nodes share a certain property or result from a clustering method. It can also be used to organize the nodes. MGV allows all nodes to be grouped together. This can either be done manually, by selecting nodes and grouping them via the “Create Group” plugin, or as the result of a plugin that creates groups of nodes. Node groups in MGV have two properties, a name and a color.

Moving a group automatically moves all nodes that are within the group. This is useful for swiftly organizing nodes in the graph. There are different options for displaying the size and the content of a node group. In the default view, a node group shows all nodes it encompasses, their layout is defined by the current layout of the graph. Nodes within a group can be freely moved, however the bounding shape of the box will by adjusted to fit the nodes within. The default bounding shape is rectangular. Alternative shapes are the convex hull of the nodes, or an ellipse. For SBGN submaps and compartments, standard-conforming shapes can be chosen.

The content of the group can be displayed in several ways (see figure 6.20 for an example). Apart from the actual nodes, a brief summary of the nodes and probes present can be displayed. For functional analyses, the group can display meta information which is associated with the probes, summarized by their frequency. It is displayed in the form of a tag list. Alternatively, a tag list can be created from the source datasets and probe list. A sortable list of the probe names, origin, mean, and standard deviation, can be inspected. A heat map view shows an overview of the expression values of all probes.



0005739 mitochondrion inferred from direct a

0016020 membrane inferred from electronic annotation

0005739 mitochondrion inferred from electronic annotation

0016021 integral to membrane inferred from electronic annotation

0005737 cytoplasm inferred from direct assay

0005737 cytoplasm inferred from electronic annotation

0005634 nucleus inferred from direct assay

0005634 nucleus inferred from electronic annotation

0005743 mitochondrial inner membrane inferred from electronic annotation

Figure 6.20.: Grouped nodes in MGV. All groups show genes related to glucose repression in *Saccharomyces cerevisiae* [129] . The upper left shows a group displaying the nodes, the upper right shows a heat map of the probes, and the lower group displays a tag list showing Gene Ontology cellular component annotations.

Both the heat map and the probe table send their selections to the graph viewer and other visualization plots.

Groups of nodes can be created via grouping of nodes according to their properties, by probe list, by source dataset, based on common meta information or via a threshold binning method on the expression profiles. Creating groups via clustering algorithms is also possible. For this purpose, k-means clustering or quality based clustering can be used. Using modularity clustering [139], the graph can be clustered based on the nodes' connectivity. For this purpose, MGV makes use of implementation by Noack [141]. To display groups that are the result of modularity clustering, the convex hull style is useful.

The examples in figure 6.20 show how node groups allow to structure nodes, and how they introduce new views of the nodes enclosed by them. The number of nodes per group is not limited, and the minimize and restore function allow to handle large numbers of nodes. Thus, the tag lists and heat maps shown in the figure can represent a large number of nodes and help to reduce clutter in this way.

6.6.3. Working with Graphs in MGV

Graphs can be created in MGV from probes, probe lists and other components in Mayday. New elements can be added in different ways, as described below. The basis for graph creation in MGV are probes in user selected probe lists. These probes are structured by a component called `GraphProvider`, which transforms them to a graph. Three `GraphProvider`s are available. The default option creates a node for every input probe. The selected probes can also be used to create a compressed graph, that joins all the probes of an input probe list into one node. A third option uses the hierarchy of the GraphProviers to induce edges between the probelist nodes. Note that the first two `GraphProvider`s create graphs without edges. From this initial input, the graph can be extended by adding nodes and edges, filtered and condensed.

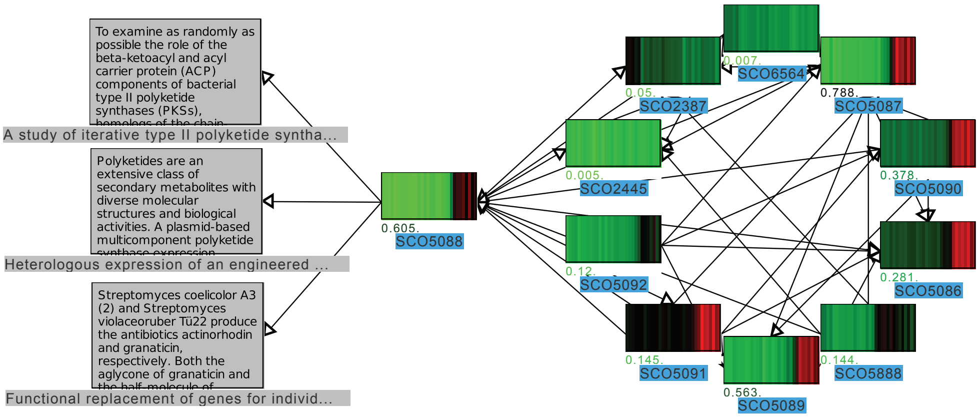
Extending the Graph

New nodes can be introduced into MGV by different means. Generic nodes, probe nodes and nodes with SBGN roles can simply be added at the mouse cursor position by invoking a key stroke (see appendix A). Via drag and drop, probe lists from the Mayday probe list manager can be imported. They can either be added as a single node that contains all probes, as an individual node for each probe, and a node that carries either a Mayday profile plot, heat map or box plot.

In addition to these manual import options, tools exist that allow to extend the graph in an automated way. These tools are implemented as plugins in the “Extend” category. The plugins in this category allow to add nodes created from Mayday objects, including probes, probe lists and meta information. Nodes that summarize the probes present at all adjacent nodes can also be added. To introduce nodes with similar expression profiles to a selected probe node, a plugin can be used that supports more than 10 distance measures. A similarity cutoff and a maximal number of results allow to restrict the number of added probes. Given genomic coordinates for the probes in the current dataset, the “Genomic Neighborhood” plugin allows to add probes in the genomic neighborhood of the probes of a node. Parameters are maximal distance, strand and relative orientation with respect to the query probe.

Edges are introduced into the graph using a mouse gesture (Ctrl+drag from the source to the target node). Furthermore, edges can be added automatically between nodes that share probes, or have probes annotated with the same meta information. Edges can also be added based on similarity of the probes in the two adjacent nodes, with respect to some distance measure. Existing edges can be annotated and weighted based on the same principle.

External data sources often contain important extensions to the information present locally in Mayday. For this reason, three external data sources can be queried to extend the current graph in MGV with different information. These plugins are usually used on one query node associated with at least one probe, from which a query for the webservice API of a web site is created.



To examine as randomly as possible the role of the beta-ketoacyl and acyl carrier protein (ACP) components of bacterial type II polyketide synthases (PKSs), homologs of the chain-

A study of iterative type II polyketide syntha...

Polyketides are an extensive class of secondary metabolites with diverse molecular structures and biological activities. A plasmid-based multicomponent polyketide synthase expression

Heterologous expression of an engineered ...

Streptomyces coelicolor A3 (2) and Streptomyces violaceoruber Tü22 produce the antibiotics actinorhodin and granaticin, respectively. Both the aglycone of granaticin and the half-molecule of

Functional replacement of genes for individ...

Expression values: 🟩 low ⬛ medium 🟥 high

Figure 6.21.: Abstracts and interaction partners of SCO5088. Abstracts were retrieved from PubMed, and interaction partners are fetched from STRING. A high correlation of interaction partners genomically located near SCO5088 to this gene can be observed. The numbers below the heat map denote the regularized variance of the probe.

The STRING database [181] contains protein/protein interaction data. This database can be queried by the "String Interaction Partners" plugin. The plugin contacts the database via the String Web API and parses the results that are returned in the PSI-MI-Tab format. Resulting interactor names are mapped to probes, which are added to the graph as new nodes and connected to the query node (see figure 6.21).

Information about genes and their effect is often stored in unstructured form in journal publications. In the biomedical community, PubMed is the default search engine for such information. The "Abstracts" plugin queries PubMed, with an optional preprocessing step that finds relevant PubMed ids in STRING. Abstracts matching the query are added to the graph as notes (see figure 6.21). These nodes are annotated with PubMed ids and can be opened in a web browser using the "Look up" function.

The UniProt database [184] contains a vast amount of annotations for numerous proteins from all organisms. This data can be queried using the "UniProt" plugin, which accesses UniProt and retrieves an XML file containing the annotations. Currently, alternative names, structural similarities, interaction partners and literature abstracts like generated with the above plugin are added to the graph.

Joining Nodes

Visually analyzing a graph might require to collapse a number of nodes to a single node. For this purpose, the plugins in the "Shrink" category allow to systematically reduce the number of nodes within a graph. Relevant options allow to merge nodes with identical name, nodes that adjacent, or that have probes with similar expression profiles. Merging all selected nodes is also possible. To generate a coarser view of the data, it is possible

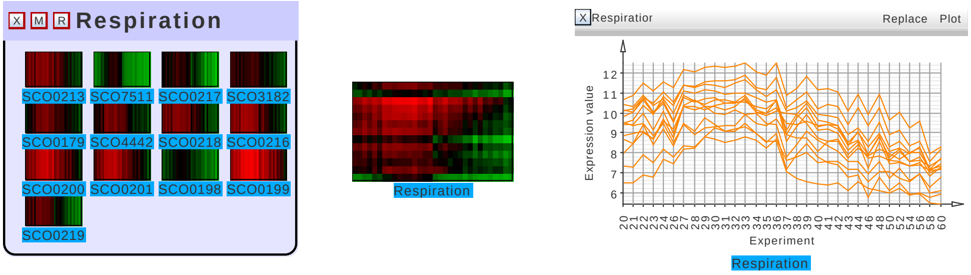


Figure 6.22.: Methods of joining nodes in MGV. The original nodes are grouped on the left. The center shows a single node carrying all probes. On the right, a Mayday profile plot displays the same data.

to replace selected nodes by the probe lists they are contained in. The resulting nodes represent the entire respective probe lists. Edges starting or ending at removed nodes are redirected to the resulting joint node.

To remove nodes while keeping the probes that are associated with them in the plot, they can be replaced by a joint node with all probes, or by a Mayday visualization plot (see figure 6.22 for an example). This allows to include a large plot to the visualization that gives an overview of a number of nodes.

Filtering Nodes

Filtering nodes allows to remove unwanted content from the graph. For this purpose, several filtering plugins are available.

In general filters are formulated as following: An operation is selected, which is applied to all nodes that *do not* match the filtering criteria. It is possible to invert this condition: then, nodes that *do* match the criteria are processed as selected.

Filtering can be done on Node criteria (name, role, degree) and probe properties (the dynamic probe list interface is used for this purpose, cf. [12]). In addition, the dataset origin or the number of probes present at a node are criteria for filtering. Nodes that are processed when not matching a filter are subject either to be set to minimal size, be hidden, deleted, merged to a single node, grouped together, or they can be assigned a selectable role or auxiliary item.

6.6.4. Data Integration

Loading and saving Graphs

MGV can save and restore its graphs in a native format based on GraphML [20], which preserves all rendering settings, groupings and viewer settings. For general graphs from other applications, support for several graph data formats exists. Graphs in GraphML, DOT (GraphViz), GML and XGMML formats can be imported. These formats support graphs for all applications. Currently, the topology of the graph, and the names and properties of nodes and edges are imported. Not all properties are used to direct the display of the nodes. For export options, the graph can be exported into GML, GraphML and GraphViz DOT formats.

Integrating Biological Data

Biological graphs are in general not limited to the general purpose formats mentioned above. Several formats for biological models exist, including KGML and BioPax [8], which is used by multiple providers of biological pathways. Biochemical models are denoted in SBML [81] or CellML [39], while interactions are often exchanged in the PSI-MI format [100].

Creating a graph based on these formats is not generally straightforward. The domain-specific models employed for all of the above file formats do not directly give rise to a useful data structure. Furthermore, different graphs can be extracted from a file format. This is especially true for data in BioPax files.

The interface that MGV used to address this problem is the `GraphModelProvider`. Components that implement this interface provide the details for an auto generated dialog that directs the user through the import of the data. This dialog has five steps:

1. *Method of data import.* These options are described in this section.
2. *Import Settings.* This allows to select the input files and set parsing options.
3. *Select probe lists.* In this step, the expression data to be viewed with the graph is selected from one or more datasets.
4. *Configure Probe Lists.* This allows to configure how probes from different datasets map to each other or to identifiers in the input files.
5. *Additional configuration* depending on the input format. This allows for example to select individual pathways from BioPax files.

The selected probe lists are mapped by the format-specific graph creation method to the probes, using the mappings selected in step 4. `GraphModelProvider`s also define a layout method, which is used by default to lay out the imported graph. This can either be a standard method, or a data specific layout.

Data Mining of BioPax Files

BioPax files contain large amounts of data which are by no means limited to pathway definitions. This information can be used to characterize the pathway and reaction landscape of an organism. It can also give an overview of the overall metabolic activity, when combined with expression data. BioPax is a format based on RDF, the Resource Description Framework (standardized by the W3C). An observation about this framework is that each entry can be seen as a triple of format $(subject, predicate, object)$. An example for such triples is shown in table 6.2. This concept is used by triple stores, which are optimized for storing and querying RDF data. A simplified version of this principle here is used for parsing and mining the BioPax files. A component called `BioPaxMiner`

`smallMolecule618210`	`OBJECT-TYPE`	`smallMolecule`
`smallMolecule618210`	`XREF`	`unificationXref618215`
`smallMolecule618210`	`SYNONYMS`	`H2O`
`smallMolecule618210`	`SYNONYMS`	`hydrogen oxide`
`smallMolecule618210`	`SYNONYMS`	`water`
`smallMolecule618210`	`STRUCTURE`	`chemicalStructure618216`
`smallMolecule618210`	`NAME`	`H2O`
`smallMolecule618210`	`MOLECULAR-WEIGHT`	`18.015`

Table 6.2.: RDF triples describing the water molecule in a BioPax file. The triples were created by the `BioPaxLoader` of MGV. BioPax files contain many cross-references (`XREF`) to database entries. Some triples are omitted here for brevity.

parses a BioPax file and stores all triples of information in a single table in an in-memory SQLite database. SQL queries can be used to extract the required data.

This procedure has some advantages. The database can, to some degree, optimize the queries used for mining the SQLite files. The current parser works for BioPax level 2, and due to the fact that no assumptions about the format are made, also for BioPax levels 1 and 3, however for those the queries need to be adapted. The overall implementation, without the SQLite jar file itself, is lightweight (1400 lines of Java code). This approach requires a large number of triples stored in a unoptimized database schema. Many of these triples might be unnecessary for most tasks. However, SQLite scales well even for large tables and allows fast queries even for large files.

Based on this component, several options for extracting data from BioPax files are implemented for MGV. Complete pathways from BioPax files can be loaded via the "BioPax (Pathway)" option. They are laid out and drawn in the same style as in the SBGN pathway viewer (see above, section 6.5.2). Furthermore, it is possible to import graphs that arise from the *interconnections of pathways and reactions*. These graphs can be imported using the BioPax `GraphModelProvider` option. In this case, from a BioPax file and a mapping to identifiers, the user can select the pathways which are to be investigated from a list. This could be all pathways annotated in an organism, or a subset.

For the intersections of pathways, the resulting graph $G = (V, E)$ has as set V the selected pathways, and the set of edges $E = \{e = (V_1, V_2) : I_{V_1,V_2} \neq 0\}$. I_{V_1,V_2} is defined as the number of chemicals shared by the two pathways and is used as the weight of the respective edges. The role of each edge is "undirected". The graph for the intersections of reactions is defined analogously. Ubiquitous chemicals, such as water, oxygen, CO_2 and several co-enzymes induce a large number of connections which are of limited biological interest. Therefore, I can optionally be calculated ignoring these chemicals. A force-based graph layout is used to embed G. If possible, the nodes that represent pathways or reactions are associated with the probes for the enzymes that catalyze the respective process. In the resulting graphs (see figure 6.23 for an example), cross references and

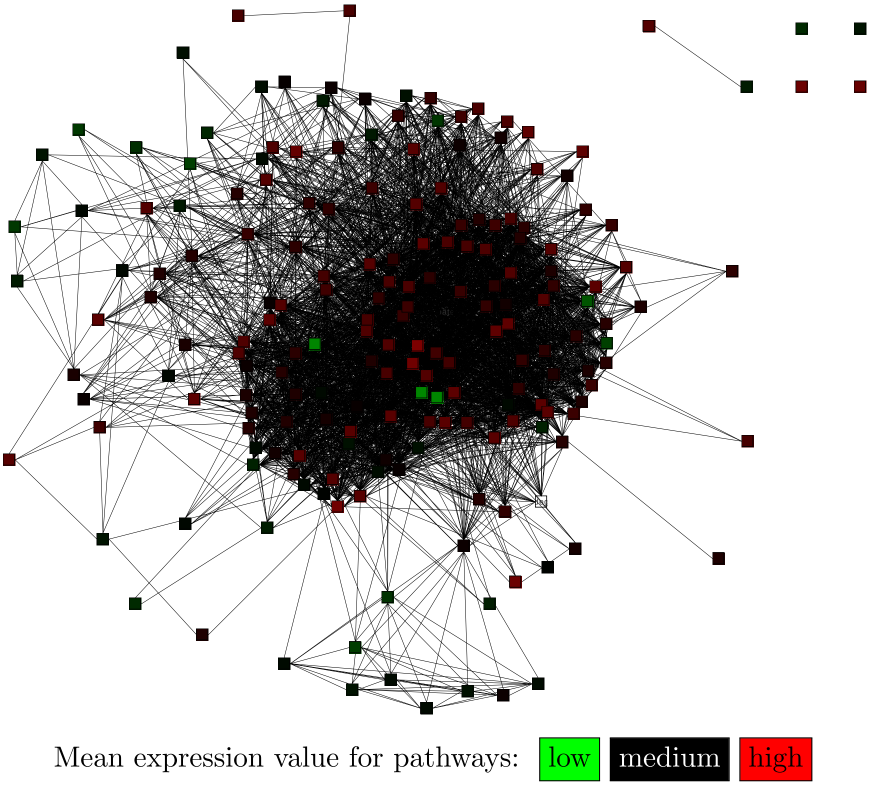
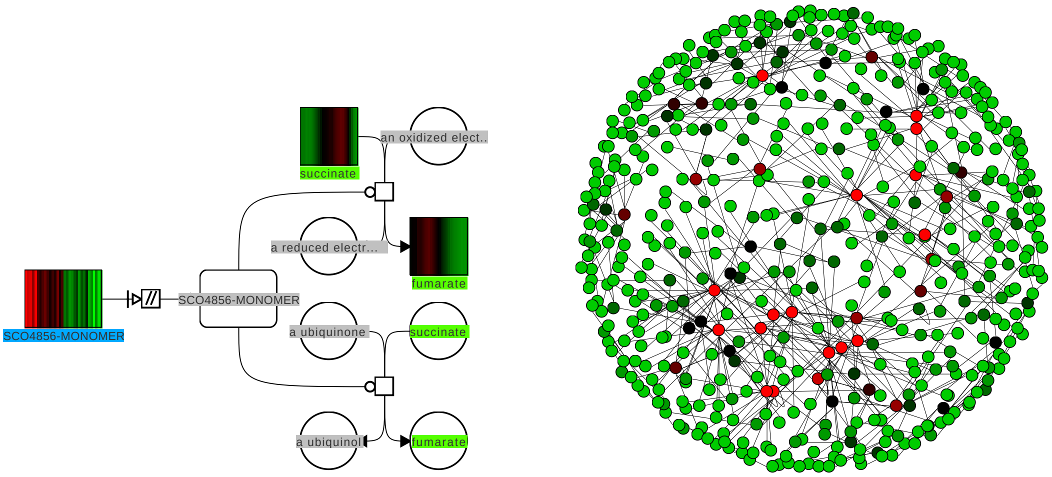


Figure 6.23.: Network of biochemical pathways in *S. coelicolor*. Nodes represent pathways, edges represent shared metabolites between two pathways. The color of nodes shows the mean expression values over a time series [140].

other information about physical entities that are stored in the BioPax file are added as node properties that can be used for web lookup.

In a multiple omics setup, it is useful to include data from different levels, especially when a comprehensive dataset is available. In this case for example, the probes for transcripts and for metabolites can be viewed with their functional relationship by the import function “Transcript - Protein - Metabolite”. This function uses a mapping from transcripts to proteins and a BioPax file to establish which proteins act as enzymes and which reactions they catalyze. An SBGN process diagram is used for this purpose. The graph has multiple connected components, one for each protein. Each component consists of a linear part, which denotes translation (the transcript is drawn as a nucleic acid feature, the protein as a macromolecule, connected via an omitted process - the translation). An optional, potentially branched part shows the reactions catalyzed by the protein. For each reaction, the substrates are on top, a process node in the middle, and the products on the bottom of the graph (figure 6.24a shows an example). This graph allows to inspect the correlation between transcript and protein and gives an estimation of the metabolic activity of the protein.



(a) Detail of transcript-protein-reaction view for the gene *sdhA* in *P. aeruginosa*. (Data from [33])

(b) Interaction network from PSI-MI file in *Saccharomyces cerevisiae*. The nodes are colored by degree centrality.

(a) Expression value and (b) degree centrality: low medium high

Figure 6.24.: Examples for data integration in MGV.

To concentrate on the relationships of transcriptomics and proteomics data, a simple component is available that aligns the transcripts and proteins, and displays them in a reduced view omitting the metabolites. This component is called “Transcript - Protein”, and lays out the graph as described above, but without the catalyzed reactions and the metabolites.

Importing Biological Models

A number of different biological models can be imported into MGV. For this purpose, a number of parsers for relevant file types, all of which are based on XML, have been implemented. Each of the following four formats can be imported in MGV via a `GraphModelProvider`.

SBML files contain biochemical reaction models, which are defined by the reacting components and a mathematical model that defines the kinetics of the reaction. SBML requires to define the type of each component, i.e. substrate, reaction, etc. using the Systems Biology Ontology (SBO) [113]. This allows a direct mapping to SBGN, as SBGN is defined as a set of visualizations for SBO terms. For this purpose, a parser for generic SBO files is available that also allows to check transitive is-a relationships within an ontology and that can map SBO terms to SBGN roles. The current implementation imports the topology of the model, but not the reaction kinetics, as there is currently no use for it in MGV.

Further formats that can be imported are CellML, PSI-MI and KGML. CellML is an alternative to SBML. CellML works by defining variables and components, which are connected. The resulting network of connections can be imported into MGV. PSI-MI is a language for denoting protein interactions, an example of a PSI-MI interaction network displayed in MGV can be found in figure 6.24b. KEGG issues KGML files as described above to describe single pathways for a single organism.

Comparing Clusterings

Cluster analysis is an important step in the analysis of gene expression data. Often different clusterings are produced during the analysis. Evaluation and comparison of clustering outcomes has been subject of research [183]. Several quality measurements, together with visualizations like silhouette plots are used for this purpose. Furthermore, it could be interesting how stable clusterings are when performed within different studies. This is related to the question of whether genes are co-expressed in two studies. Generalizing from clusterings, lists of differentially expressed genes, or genes found by prioritization methods can also be subject of comparison.

The “Compare Clusterings” function of MGV addresses this problem using a graph-based method. The input are probe lists from two or more different studies on genes for which a mapping between the probes in all datasets exist. From this input, a bipartite (in the case of two studies), or multipartite (in the case of $n > 2$ studies) graph $G = (V_1, \ldots, V_n, E)$ is created. $V_i = \{V_{i1}, \ldots\}, i < n$ is the set of clusters from the ith study. Node that all V_i are disjoint. The set E of edges is defined as follows: An edge e is introduced in G if clusters V_{ia} and V_{jb} share at least one gene, i.e. $V_{ia} \cap V_{jb} \neq \emptyset$. Edges are introduced in this way for all studies $i \neq j$ and all clusters $a \in V_i, b \in V_j$.

To reduce the number of intersections, it is possible to modify the criterion for edges to $|V_{ia} \cap V_{jb}| > t_e$ for some threshold t_e. Likewise, only clusters V_{ia} can be regarded if $|V_{ia}| \geq t_c$, i.e. if the cluster V_{ia} represents at least t_c genes. Furthermore, it is possible to remove all unconnected clusters from the graph.

If the overall set of genes in the clusterings is identical for all samples, then a significance test based on the hypergeometric distribution can be performed to find intersections between clusters that are significant [60]. Let n_i, n_j be the number of genes in cluster V_{ia} and V_{jb}, respectively and let $m = |V_{ia} \cap V_{jb}|$ be the number of genes both clusters share. Let N be the the total number of genes. Then

$$p(m) = \frac{\binom{n_i}{m}\binom{N-n_i}{n_j-m}}{\binom{N}{n_j}} \tag{6.1}$$

is the probability (based on the hypergeometric distribution) of observing an intersection of size m when randomly sampling n_i genes and independently randomly sampling n_j genes from a total of N genes [60]. Based on equation 6.1, the probability for all $k, m \leq k \leq \min(n_i, n_j)$ is given as

$$p = \sum_{k=0}^{\min(n_i,n_j)} p(k) - \sum_{k=0}^{m-1} p(k). \tag{6.2}$$

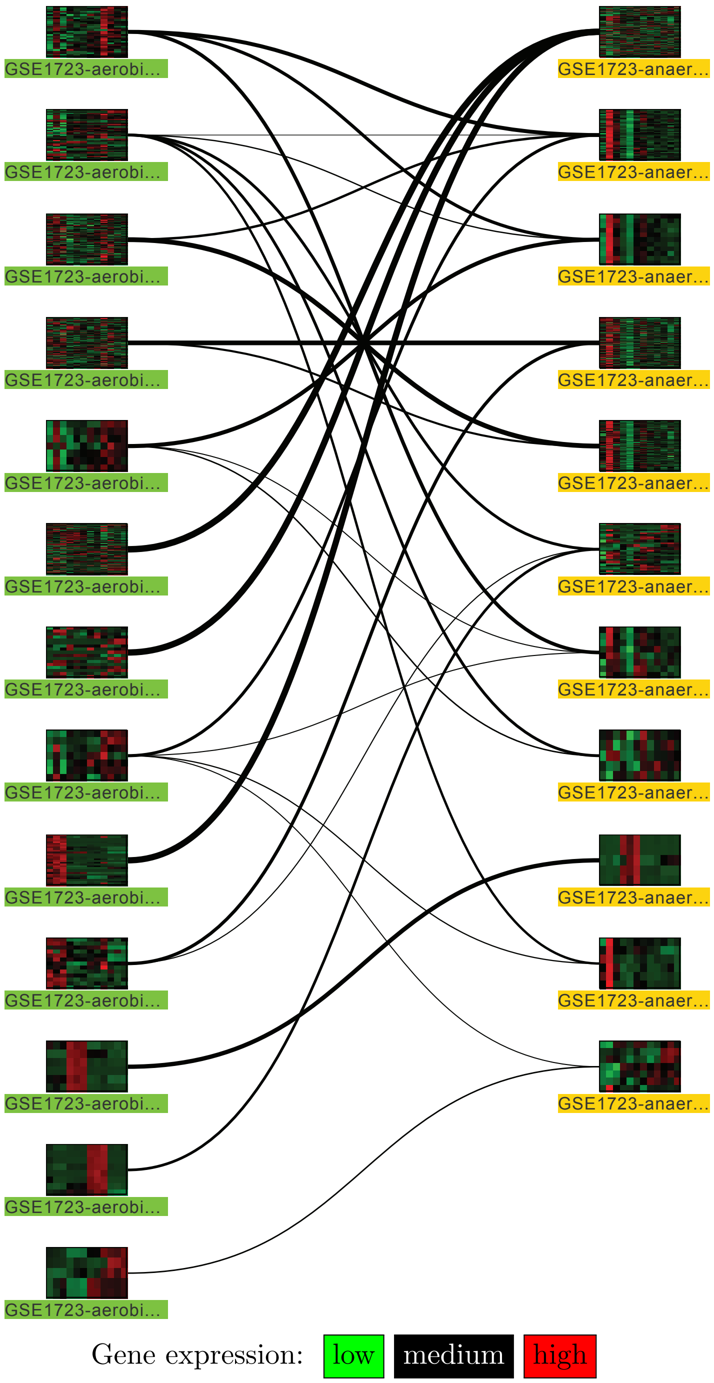


Figure 6.25.: Comparison of clusterings between aerobic (left) and anaerobic conditions (right) in *Saccharomyces cerevisiae*. Edges represent significant overlaps between clusters ($p < 0.05$).

Using this p-value, we can conduct a statistical test. The null hypothesis is that the intersect of size m of two independent random samples of size n_i and n_j out of n objects, is a random event [60]. For the calculations of the distribution function of the geometric distribution, the Apache Math library (`commons.apache.org/math`) is used.

The example in figure 6.25 compares clusterings of genes in two studies on aerobic and anaerobic nutrient limitations in *Saccharomyces cerevisiae* [103]. The conditions in both datasets can be considered comparable. The graph only shows significant ($p < 0.05$) overlaps. The example indicates that there is limited concordance between both clusterings: The largest cluster in the anaerobic conditions intersects with three smaller clusters in the aerobic study, two of which however have different expression profiles. In contrast, several cluster intersections exist, in which both clusters have very similar profiles.

A similar view can be used to compare different *statistical measures*. Comparisons of statistical values, feature selection scores or other gene prioritizations can be made using the statistical values comparison tool for MGV. This tool creates a bipartite graph based on the ranking of two numerical meta information groups. The probes are ranked and binned independently for each meta information and visualized as a bipartite graph. For each pair of groups, an edge is added to the graph if the groups have an non-empty intersection. The weight of each edge is set to the number of intersecting genes.

Visualization of Alternative Splicing and Differential Expression

Differential splicing is a common process in eukaryotes and is a major source of genomic variation and adaption. It is known that different transcripts that arise from the same gene locus can have different lengths, functions and, naturally, differences in expression levels. The visualization of differential splicing as a graph has been proposed by Heber *et al.* [73], who coined the term *Splice Graph*. In combination with visualization of expression on the nodes, a splice graph can represent differential activity on two levels: differential splicing and differential expression. These measurements became feasible using RNA-Seq both on transcript and exon level. The concept of MGV allows the straight-forward presentation of the splice graph. Let a transcript T be defined as a sorted set of exons e_i, defined by begin and end coordinates e^b and e^e. Sorting of all exons is done ascendingly by e^b. We assume that for all genes that have more than one transcript that $T_a \cap T_b \neq \emptyset$ for all pairs of transcripts T_a, T_b.

The transcripts of each gene can be transformed to a directed acyclic graph $G = (V, E)$, keeping a mapping M of nodes to exons, $v \in V \to \{e_i, \ldots\}$. Introns are represented by edges between two nodes representing exons. Several ways of transforming a gene to a gene model graph are considered here:

1. *Verbose*: Each transcript T is considered separately, with each exon being represented by a node. In M, each node maps to a single exon from one transcript. The resulting graph G consists of t connected components (with t being the number of transcripts).

2. *Simplified*: Starting from the 5'-most exon, identical exons from different transcripts are combined to a single node, as long all transcripts contain the same exons. When different exons occur, the process is repeated for each unique set of identical exons. G can contain one or more connected components.

3. *Compressed*: For each unique exon, exactly one node is produced. Therefore in M, one node maps to one or more exons from different transcripts. G consists of one connected component. This is equivalent to the original splice graph suggested by Heber *et al.* [73]

The implementation of this concept in MGV uses genomic coordinates parsed from Generic Feature Format (GFF) files with genomic annotations. This is based on the framework described in [11]. In MGV, a `GraphProvider` is available that allows to create splice graphs of all three types from a set of annotated input probes.

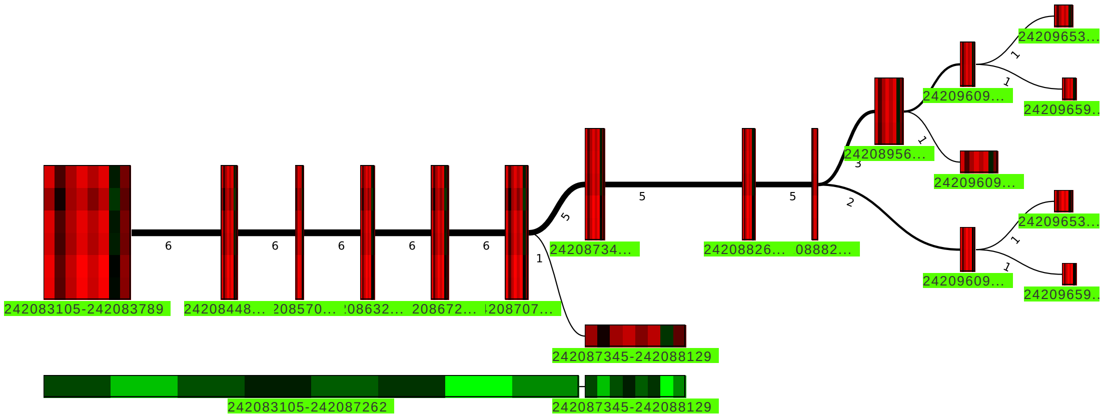
Drawing the splice graphs (see figure 6.26) is done representing the order of the exons, i.e. the start and end coordinates of the exons determine the horizontal position of each node as well as its width. The height can be chosen arbitrarily. Long transcripts need to be scaled to fit to the screen: exons and introns overlapping exons are drawn to scale while other introns are scaled to a fixed width.

For the verbose case, the layout is straightforward, as each component of the graph is completely linear. Simplified splice graphs can be laid out in a cladogram-like fashion, placing the parent node at the mean height of the children. Compressed gene model graphs could be placed by sorting overlapping nodes by degree and thus placing the most commonly used exons on the top of the layout. This potentially reduces the overlap of the edges with the highest weight.

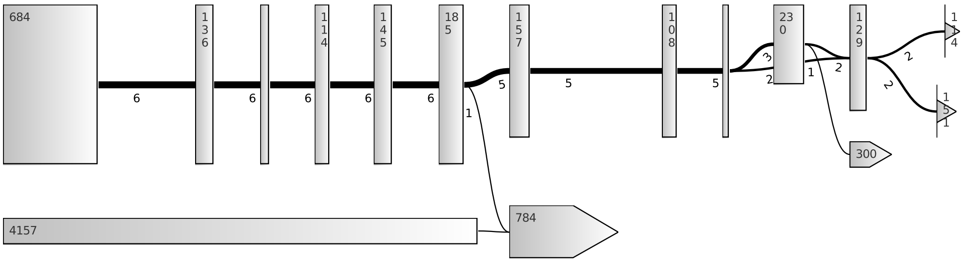
Classically, gene models are visualized with the exons as boxes. Often, arrow heads indicate the 3' direction on the last (3'-most) exons. Introns are usually represented as lines. This kind of view is well-established and is not changed here. The nodes representing exons, however, can be used to represent additional data, e.g the length of the exon, the number of transcripts it occurs in, or the genomic coordinates of the exon. Renderers for visualizing transcript abundance (i.e. gene expression) can also be used. However, the different length of the exons might introduce a bias in the perception of parallel coordinate plots.

Visualization can then be used to study differential expression and splicing. The examples in figures 6.26 show the different transcripts of the gene *STK25* (serine/threonine kinase 25) of *Homo sapiens* in several cancer cell lines (data from Illumina, Inc, unpublished). From figure 6.26a we can infer that the bottom-most transcript is only marginally expressed and retains 5 introns compared to the six other transcripts, which share a common 5'-end. Figure 6.26b gives an overview of the splice variants, highlighting the four different 3'-ends.

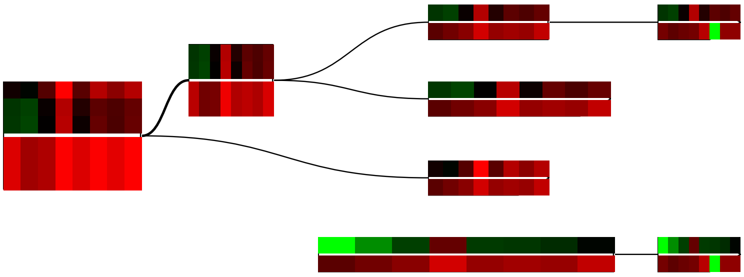
In the case that exon-based values are also available, it would be of interest to compare the transcript-based and exon based values. For this purpose, the exon renderer in MGV can be configured to display two heat maps: one displaying the transcript totals, one displaying the exon-specific values. This is illustrated in figure 6.26c, where the difference



(a) Simplified view of the transcripts of gene *STK25* in *Homo sapiens*. The heat maps on the nodes indicate differential expression in several cancer cell lines (columns of the heat map).



(b) Compressed splice graph for *STK25*. The number in each node denotes the length of each exon.



(c) Visualization of both transcript and exon values for a short gene model in *Homo sapiens*. The upper heat map represents the transcript, the lower one represents the exon.

Color gradient for gene expression in (a) and (c): low medium high .

Figure 6.26.: Differential splicing and gene expression displayed in MGV using the simplified (a,c) and compressed splice graph (b).

of exon and transcript RPKM values can be inspected. In the example, in most cases a high correlation between transcript and exon values can be observed.

Comprehensive Gene Annotations

Often, it is interesting to aggregate all information available about one single gene. For this purpose, the “One Probe” import function aggregates all information available to MGV on a selected probe. For this probe, internal and external information is queried. Internal information encompasses the Mayday probe lists the gene is contained in, as well as meta information, and genomic neighbors. The expression profiles in other dataset can also be queried. External sources are BioPax files, from which reactions and pathways the gene’s product takes part in are extracted. Interactors can be queried from STRING, and further annotations are fetched from UniProt. The resulting nodes are grouped by origin. The entire graph is laid out in a star-like topology. Figure 6.27 shows an example for the gene *pgi* (glucose-6-phosphate isomerase) in *S. coelicolor* with five groups of annotations. In this view, the position of the gene in the primary metabolism of this gene can be inferred.

6.7. Cross Dataset analysis

MGV is prepared to visualize and analyze data from different datasets. This is necessary in order to properly integrate data from different omics. Different experimental platforms lead to different ranges and distributions of values. Even if the same technology is used, data integration is not trivial (for microarrays see e.g. [164]). Combining all data in one expression matrix is therefore usually not helpful. Still, combining data from different platforms is important, as this allows to explore the data on different levels. Furthermore, different time points or experimental conditions investigated in different studies can be compared. The graph data structure used in MGV makes, in general, no assumptions on which dataset a probe comes from. Therefore, MGV can be used to visually compare data from different sources if some adjustment are made to the applications. For visual comparison, this is straightforward and often automatically used in several data integration methods (see for example figures 6.25 and 6.24a). However, MGV supports further tools for integrating data from several datasets.

6.7.1. Cross-Dataset Visualization and Statistics

Mayday can work on different datasets in parallel. Operations on *multiple datasets* are already implemented for time series alignments [12]. MGV adds the possibility to jointly visualize and analyze any data from two or more datasets.

This is achieved by keeping different `ViewModel` instances for each dataset. This ensures that all nodes are rendered correctly and allows independent value ranges and color gradients for each dataset. This is sufficient for visualizing multiple datasets within a single MGV plot, provided no nodes exist with probes from more than one dataset. The joint visualization of probes from different datasets is possible via node groups. Node

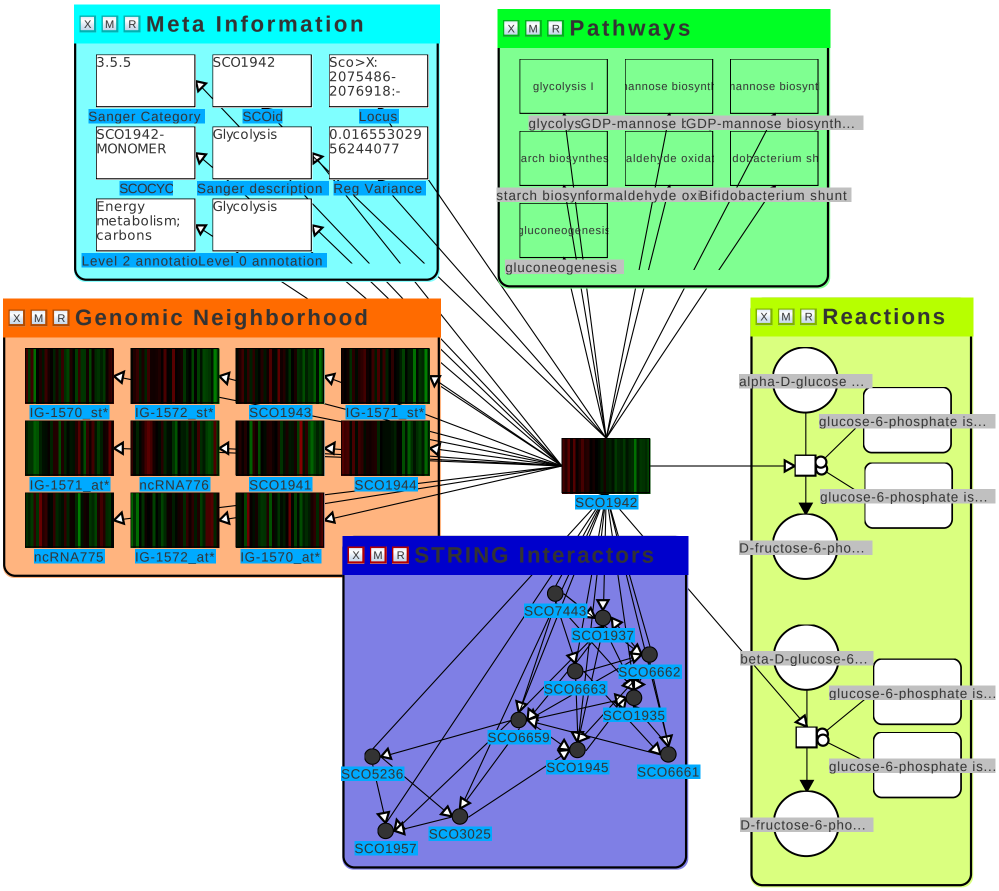


Figure 6.27.: Annotations graph for the gene SCO1942 (glucose-6-phosphate isomerase) in *S. coelicolor*. The graph contains meta information, related pathways and reactions parsed from a BioPax file, interaction partners retrieved from STRING and the genomic neighbors. Expression data from [140]

groups allow the joint visualization of probes from multiple datasets via their heat map function.

When dealing with different datasets, probes and experiments may differ between them. Therefore, a mapping of what is common between datasets is necessary. A mapping of experiments common to the datasets is kept, along with a mapping of equivalent probes. The probe mapping can be inferred from probe names or meta information. The experiment mapping can be set up from predefined settings, e.g. based on common names or time points. All mappings can be manually altered. The probe mapping can be viewed in a mapping tree in which each set of equivalent probes is summarized. Based on these mappings, statistical comparisons and visualization can be created.

A supporting view of multi dataset graphs is the correlation heatmap (see figure 6.29b). The correlation heatmap provides a pairwise comparison of probes from different datasets.

$$\left\{\begin{array}{c}\text{Distances (between)}\\ \text{Distances (within)}\\ \text{Distances (star)}\\ \text{Correlation (between)}\\ \text{Correlation (within)}\\ \text{Correlation (star)}\\ \text{RV Coefficient}\end{array}\right\}\left\{\begin{array}{c}\text{Dataset}\\ \text{Node Name}\\ \text{Node Role}\\ \text{Probe Name}\\ \text{Probe Display Name}\end{array}\right\}\left\{\begin{array}{c}\text{12 Distance}\\ \text{Measures}\\ \text{— or —}\\ \text{4 Correlation}\\ \text{Measures}\end{array}\right\}$$

Figure 6.28.: Overview of the options for cross dataset analysis. From left to right, the statistical operation, the grouping strategy, and the parameter for distances and correlation.

It shows three columns: the left and right columns display the expression values of the probes in different datasets. The central column shows one of three correlation measurements. Either the Pearson correlation coefficient, Spearman's rank correlation coefficient or Kendall's τ rank correlation coefficient can be used. For the calculation, only the mapped experiments are used. The correlation heatmap reflects data transformations and communicates selections to the graph. The columns can be sorted by probe name or correlation coefficient.

Node groups can be formed from nodes with probes from any dataset. They can also be used to calculate and visualize basic statistics from probes from multiple datasets. The "*Calculate Statistics*" function allows to calculate distance and correlation measures within or between groups of probes. Alternatively, the correlation or distance to a central element can also be calculated. Groups can be defined in various ways, including using the dataset and node name. A full view of the options is given in figure 6.28. The calculated statistics are stored at the node group and can be displayed as heat maps, either within the node group or in a separate window. Correlations and distances between groups containing multiple objects are summarized using the mean or median. No summarization is necessary for the *RV* coefficient [149]. Let $\mathbf{X}_{i\times j}$ and $\mathbf{Y}_{i\times k}$ be matrices, then the *RV* coefficient is defined as:

$$RV(\mathbf{X},\mathbf{Y}) = \frac{\operatorname{tr}(\mathbf{X}\mathbf{X}'\mathbf{Y}\mathbf{Y}')}{\sqrt{\operatorname{tr}((\mathbf{X}\mathbf{X}')^2)\operatorname{tr}((\mathbf{Y}\mathbf{Y}')^2)}} \tag{6.3}$$

where $\operatorname{tr}(X) = \sum_i X_{ii}$ denotes the trace of a square matrix. A modification was suggested by Smilde *et al.* [166] that addresses the overestimation of correlation by the above formula. Both methods are implemented in MGV.

6.7.2. Example

Explorative analysis of data from different sources can reveal common features and differences. Figure 6.29 shows an example using two studies investigating aerobic and anaerobic growth of *Saccharomyces cerevisiae* under four nutrient limitations [103]. In the aerobic dataset, a number of clusters were identified, and we are interested in how the genes of each cluster are regulated under the anaerobic conditions. In MGV, the clusters are shown using profile plots. This gives a first overview of similarities and differences (see figure 6.29a). Cluster 2 (6 probes) and cluster 5 (8 probes) are very similar in both conditions. Often, only a subset of the genes differ. In the correlation heat map (figure 6.29b), we can see that the correlation is, in general very high. However, we find some probes that have a strong anticorrelation between both datasets. Investigating these probes yields that the most highly anticorrelated ones belong to a cluster of 30 probes, many of which are related to respiration processes. The four nearly perfectly anticorrelated genes are annotated with biological processes related to oxygen reduction. In general, the mean euclidean distance of clusters (see figure 6.29c) to all others is especially high for two clusters with 7 and 31 genes. Many genes in these clusters are related to ribosomal proteins.

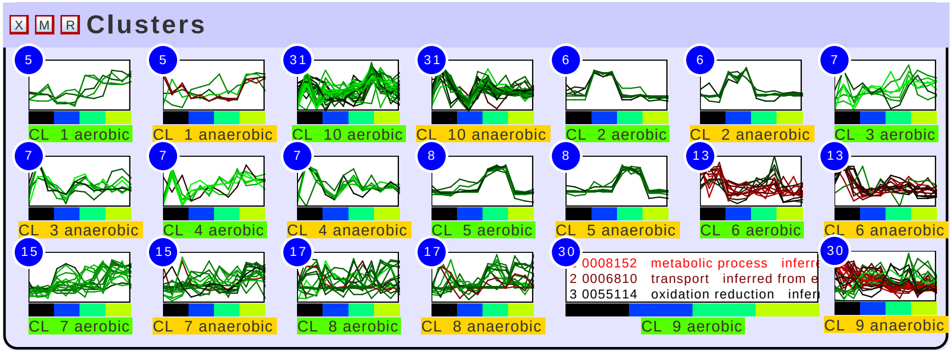
This example shows how to search for expression patterns between different datasets. In a first step, the different display options of MGV are used to get an overview of the data. Using the correlation heat map, can be used to verify cartain hypothesis. Overall similarities and differences can be searched for using the distance heat maps. These two heat map implementations complement the visualization tools provided by MGV.

6.8. Summary and Discussion

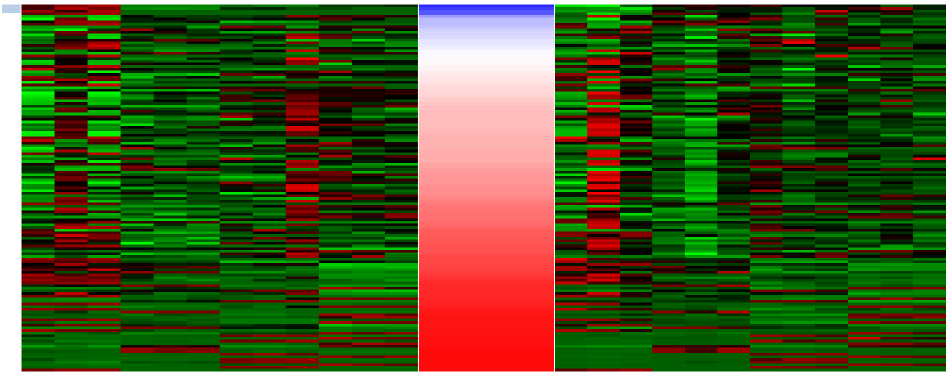
In this chapter, a generic concept for the joint visualization of gene expression data, meta information and biological models was described and implemented. Based on the related concepts in data visualization and integrative bioinformatics, the question of how to display both data and meta information was already addressed, for special cases, in the previous chapter. This question is addressed in this chapter in a much more generic way, based on graphs and the massive usage of small multiple plots of various kinds. From the existence of numerous related, though often lacking programs in this field it can be concluded that this approach is feasible.

The Mayday graph framework serves as a lightweight foundation for the implementations. The framework was designed to ensure extensibility and to allow for swift integration of new methods for many purposes related to graph and data visualization via plugins. A number of visualization tools allow to display many different types of data in different styles. Generic nodes, single expression profiles, multiple profiles, with or without meta information and manual annotations can be visualized within this framework.

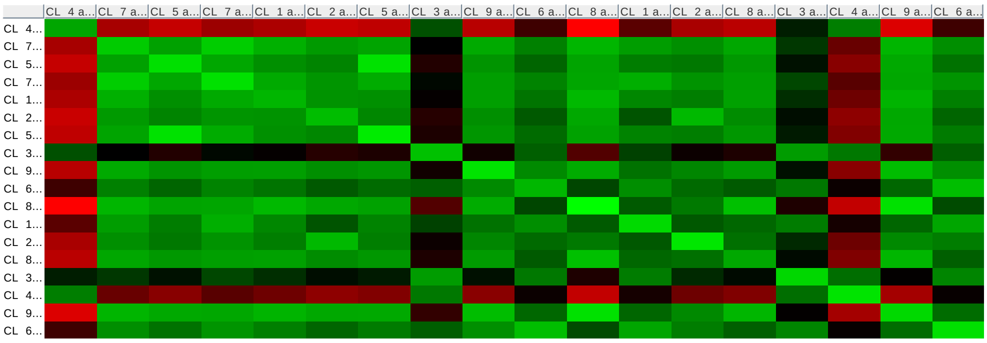
Applications of this framework are twofold. The specialized pathway viewers, the KEGG and SBGN pathway viewers integrate the data/meta information combination into standard tools in biomedical research. The two pathway viewers employ many of the strategies for visualizing gene expression data and meta information. Being included in the Mayday visualization framework, they can further rely on the tools presented in chapter 5 to view additional data in parallel. The KEGG pathway viewer attempts to be a user-friendly front end for exploring and browsing pathways. The SBGN pathway viewer is a lightweight alternative to many extensive tools which are available. Some



(a) Clusters in yeast chemostat study. The values are z-score transformed for better comparison of the profiles. For cluster 9 (30 genes), the gene ontology (biological process) is shown.



(b) Correlation heat map of the chemostat study



(c) Distance heat map showing the euclidean distance using a green-black-red gradient.

Figure 6.29.: Comparative visualizations of 10 clusters in the aerobic and anaerobic datasets. The correlation heat map in (b) shows a number of highly anti-correlated genes. The distance heat map in (c) identifies four clusters with vastly different genes.

helpful tools for visual exploration are implemented, like summaries of reactions. The host of different pathways available in the BioPax format and the stringent visualization of the processes in SBGN style make it useful in many studies. The scope of these tools, however, is limited, and the visual styles of KEGG and SBGN are not expressive enough for the variety of meta information and networks available. The generic Mayday Graph Viewer (MGV) offers a superset of their functions, in a less specialized, more general setup.

MGV integrates all described tools for data integration and visualization. It is designed to visually explore and analyze data from multiple sources in the context of biological models. This is extended by some specialized tools for data analysis. Also, MGV can be used as a general purpose visualization tool. The scope of MGV is not to visualize huge biological networks, but instead finding interesting patterns in datasets, on which first analyses have already been made. For this, numerous visualization tools for gene expression data, along with filtering and summary methods are provided. MGV is therefore especially useful for summarizing results and formulating hypotheses. The combination of graphs, which state a model, and expression values allows to produce a joint visualization of model and data in order to succinctly present a hypothesis. Building blocks of for this can be graphs created by the methods described in section 6.6.1.

MGV includes analysis tools into the graph visualization. For this purpose, the node groups, with their statistical and visualization tools have been integrated. They allow to visually group nodes, and use these groups as the basis for new visualizations. These visualizations summarize the data or add new aspects. Node groups also allow to structure the data and help to remove clutter from the graph, without deleting nodes.

The cross dataset tools extend the scope of MGV. Integration of vastly different data is possible. For this kind of data integration, visual exploration is the tool of choice. It is facilitated by the statistical tools described in section 6.7 and the correlation heat map. The correlation heat map, while simple in design, allows for swift visual and quantitative comparisons between two datasets. The presented statistical functions are currently limited to exploratory tools that allow to identify similarities and differences in the data. While this proof of concept allows swift analysis as described, additional tools for three purposes would be useful extensions. The integration of statistical methods for integration like co-inertia analysis [40] or methods based on partial least squares [112] extend the concept beyond visual comparison. Integration of time series alignment tools facilitate the experiment mapping.

The memory consumption of the framework is moderate, and can support large sparse graphs. If a dataset can be opened in Mayday, the additional memory consumption for a graph is independent of the number of probes used. Memory consumption is mostly depending on the number of nodes in a graph. The rendering speed is mostly depending on the number of edges and probes in a graph. The rendering strategy used for the nodes also determines the frame rate. However, rendering leaves room for optimization. First tests indicate that selectively caching nodes that represent many probes can greatly improve performance and is an acceptable trade-off between memory consumption and rendering speed.

The concept and implementation of MGV is open for extensions. The various extension points in MGV allow to extend rendering capabilities, graph layout and analysis tools such as graph measures, without the need to modify core components.

Systems biology often requires data exchange between applications. A framework for this purpose based on Java RMI (remote method invocation) is the Gaggle [162]. Many applications, including MeV and Cytoscape support Gaggle. The integration of Maydayx MGV and Gaggle is currently under way. This will allow to better exchange data, including graphs, between applications supporting Gaggle.

MGV can be extended in many points. Some extensions might challenge the boundaries between the graph viewer and the main application. Many features of Mayday are already invocable from MGV. Future versions of Mayday might use the graph-based presentation of data on all levels as their main way of presenting and organizing data. The node groups that can be used in MGV, intended to extend the probe list concept of Mayday, already are a step into this direction. Node groups can combine nodes from all sources, and as presented in 6.7, basic calculations can be executed on them. How far this concept can be extended, would be subject to user studies. However, similar network based products like Hugin [101], a tool for collaborative data analysis or SpicyNodes [46], which uses a tree structures for data presentation achieved promising results at user studies.

In conclusion, the theoretical and practical framework presented here proved to be more than adequate for integrating visualization of gene expression data and a wide range of meta information. The framework is generic enough not to be limited to a single kind of data. It allows rich visualizations of present and future gene expression, proteomics and metabolomics studies in context.

7. Discussion and Outlook

This thesis presents new approaches for analysis and visualization of complex, multivariate high-dimensional data from high-throughput life science experiments. The analysis of gene expression – an important part of biomedical research – is a challenging endeavor on multiple levels. While several technologies allow to measure the activity of genes, or the concentrations of proteins and chemical compounds, the analysis of the resulting data is a problem at least as complicated as the laboratory work necessary for generating it.

In this work, an exploratory data analysis approach to this problem has been chosen. A major focus of this work is the integration of meta information using visualization and efficient analysis tools. The relevance of meta information is indisputable, as it adds the context in which gene expression happens and in which it is to be interpreted. The heterogenity and complexity of meta information requires several strategies to make use of it in different settings. Context-based plots presented in chapter 5 allow to visually explore and analyze meta information, based on specific visual representations. Interactivity and connected plots allow to combine meta information and primary data. Nominal and ordinal annotations are addressed here.

The graph-based visualization of gene expression data as described in chapter 6 is a different approach. Here, a generic data structure is chosen as a representation, combined with general purpose visualization strategies for gene expression and meta data. The context of gene expression is here displayed in two ways. The functional context can be shown as graphs, on which expression is visualized within nodes. Alternatively, meta information can be displayed via node visualization. This strategy has been found to be very generic and supports a wide range of applications, as illustrated.

Gene activity can not solely be determined by measuring gene expression. Mutations can alter and reduce or prevent the function of a gene product. This final step of gene expression analysis must be performed with genomics technologies: here, resequencing microarrays are used. The analysis of resequencing microarrays with ResqMi makes use of the same general concept: visualization of intensity and analyzed data and presentation of the context. The context here is defined as the genomic location of and the effects of possible mutation.

This chapter concludes this thesis, presenting a discussion of the major aspects of this work and providing an outlook which describes possible further research directions.

7.1. Discussion

The general concept of this work is the application of visual analysis and exploration for the analysis of gene expression data. The applicability of this concept to gene expression has been proven in extensive previous work (e.g. [52]) and helped to produce important results. Major parts of this thesis are concerned with the integration of meta information into this concept. For this purpose, two approaches were taken. Specific applications and visual representations in the context of gene expression were investigated for the context-based plots. This also included ways for including primary expression data within these representations as an application of the "Details on Demand" concept from the visual analytics mantra. A more generic approach is using a representation that can model a wide range of data: graphs. The graph data structure has also been proven to be capable for representing expression data and meta information. This is applied here to produce very dense multivariate visualizations high-dimensional visualizations. The tools presented here provide a modular visualization of data on multiple levels, which is a step towards graph-based visual analysis and exploration of data.

The plots presented in chapter 5 are based on the first approach. New visual representations were applied to gene expression data and meta information. This was done in order to address the problem of visualization of meta information and gene expression data. Tag clouds and chromograms are useful extensions for primary-data centric plots, like heat maps and profile plots. Term pyramids and probe rank plots allow to identify patterns in nominal and ordinal meta information. Profile logos are designed for summarizing the shape of clusters. The examples provided showed that these concepts are feasible for gene expression data and a wide range of meta information, especially when used in parallel. Still, a more integrated and generic approach to meta information visualization would be helpful, also that includes more extension points for refinements and overviews.

The Mayday Graph Viewer implements the second, more generic approach based on graphs. It integrates many visualization tools and presents a range of options for integrative bioinformatics, visual analytics and exploratory analysis in the context of biological networks in the broadest sense. This includes the visualization of gene expression and the addition of numerical and ordinal meta information in various styles via a modular framework. A large variety of analysis and graph layout tools and many extension points allow to further increase the use of MGV. The combination of pathway viewer, general visualization tool and single aspects of a diagramming tool make MGV a useful extension of the gene expression data analysis toolkit.

The requirements formulated by Saraiya *et al.* [156] mentions features for easy integration of pathway data, overlay of biological information and expression data visualization, all of which MGV provides. Additionally, some overview functions exist. However, MGV can be improved concerning semantical scaling and aggregation of pathways beyond filtering tools and manual aggregation of nodes. In the current implementation, the focus was on the visualization features. The addition of more analytical methods (especially centrality, analysis of flows, etc.) could greatly enhance MGV.

MGV also supports the visualization of gene models. Drawing gene models as enriched graphs highlights the order as well as the expression data associated with it. In order to address challenges coming from further research concerning genome-wide transcription, transcripts that contain several instances of an exon, distant exons, or even exons from other chromosomes should be regarded. This would require adaptation for splice graphs with cycles, as well as new ways of dealing with the distances of exons. The current approach does not fail on this as the introns are not drawn to scale, but also does not exploit this to provide an enhanced view.

MGV supports handling of multiple datasets. This is an important feature, because gene expression data is complex and multiple levels of data exist. Integrating multiomics data for visual inspection and basic statistical analysis is possible. For this purpose, the graph framework is used to structure the probes from different datasets. For analyses, the node groups extend the concept of the probe list. They allow generic collections of nodes, which can represent virtually everything. The scalability of this approach to whole genome settings remains to be tested. The multi dataset framework leaves room for extensions, including new methods for data analysis and integration, especially the usage of time series alignment tools.

The pathway viewers for KEGG and BioPax/SBGN are useful for pathway centric visualization, as they support browsing and orientation in the pathway landscape. They feature a subset of the tools provided by the Mayday graph framework and present the data in two well-established visual styles. However, the tools have a limited scope, for many data analysis purposes, MGV is better equipped.

With the tools presented, gene expression data on all levels can be visualized and analyzed. For the most parts of this work, Mayday, a technology-agnostic environment, was used for the implementation of the methods. While this precludes the usage of some technology-specific properties of the data, e.g. the visualization of mass spectra from proteomics analysis, the data model of Mayday is flexible enough to include, at least potentially, a wide range of meta data, either in atomic or structured form. Furthermore, the Mayday platform is capable, by making few assumptions on the data, of working with all data in the same way. In review, MGV and the additional tools presented in chapter 5 form a powerful combination together with Maydays additional features.

ResqMi however is specifically designed for resequencing microarrays from Affymetrix. This is a limitation, even though only few alternative designs are used. ResqMi provides new useful visualization and analysis tools, which enhance the use of resequencing microarrays. Base calling algorithms remain a problem due to high no-call ratios and quality issues in the data. Rigid testing of the presented concepts using sizable external test data would be interesting. Unfortunately, the technology of array-based resequencing is obsolete. Still, the work conducted was not at all in vain. Resequencing of specific genomic regions is an important application of second-generation sequencing technologies and useful especially in a clinical setting. While base calling from array intensities is no longer necessary, the visualization and data handling features of ResqMi can serve as a template for a next-generation resequencing analysis suite.

The methods presented in this work are not restricted to bioinformatics applications. The presence of graphs and networks, that need to be correlated with numerical data is by no means restricted to bioinformatics. In the design and the implementation of this work, bioinformatics applications were the focus. However, especially for the presented graph based visualizations, this does not hinder users to apply MGV to any kind of data. The data structures of Mayday are data-agnostic not only with respect of biological data. Data and graph formats that can be imported by Mayday and MGV are used in virtually all fields. The addition of import features for new formats is feasible via the plugin interface. The other extension points allow the integration of domain-specific processing tools and rendering options.

In summary, this work presents concepts for the visualization and analysis of gene expression data on all levels, focusing on including the context of gene expression into visualizations. This is supported by the presentation of feature-rich applications that implement these concepts. Furthermore, tools and concepts are demonstrated by examples as a proof that the concepts shown here are feasible.

7.2. Outlook

Further directions of the work presented in this thesis include short term perspectives and long term goals for the relevant tools and technologies. Short term improvements on MGV are the inclusion of new data sources and visualization plots. In general, everything can be modeled as a graph, and if it is relevant to biological applications and can be enriched with expression data and meta information, MGV should be able to work on it. For this purpose, more automated methods for extensions should be used. Association mining, motif search, text mining and integration of databases and web services for all relevant purposes should be integrated. Generic and user-friendly interfaces for these purposes are to be designed and developed.

Interactivity is a key feature of MGV which is not captured in exported images. Interactive displays of graphs generated in MGV are more useful that static images, especially when placed in a rich web interface. The concepts have already been tested for general visualization plots [137]. A possible application would be the export of graphs for showing hypotheses enriched with data that allow limited interactivity, e.g. changing node renderers.

Scripting functions and a SQL query interface allow automatization and data mining features to be implemented on graphs. For automatization, Java Script, which currently receives much attention for usage in web browsers can be used. Possible applications are additional animation features, methods to generate graphs and custom analyses. Smart graphs with active components (nodes, edges, groups) can introduce new usage and analysis concepts. However, questions arise on how much activity within graphs is useful.

The SBGN entity relationship language has a high potential. Implementations, focusing on high quality layouts would greatly increase the acceptance of this standard. Furthermore, combining it with expression data and meta information might still increase the use of it, as it happened with the process description language. Extensions for specific data visualization using edges and logical nodes should be investigated. The same is basically true for the activity flow language, which can help the visualization of signaling pathways. In general, the layout requirements of SBGN merit further research on layout algorithms for biological pathways.

The addition of new analysis and visualization tools is limited by the user requirements. Users must be able to efficiently find, understand and apply new and existing methods. Therefore, a user study should be conducted, that evaluates MGV and the overall concept.

The context-based visualizations of meta information should be extended and improved as new types of meta information and applications arise. New data processing and transformations are of interest, as are new visual representations, like timeline trees [25] or stacked graphs [26]. In general, a user study could be helpful to evaluate existing and potential principles for the joint visualization of meta data and primary data.

The development of ResqMi highlighted some important design guidelines for the analysis of future resequencing technologies. Resequencing makes the analysis of single genes more feasible then whole genome sequencing. The challenges and costs of wet lab, data processing and storage and ethical considerations forbid the sequencing of whole genomes on a daily basis for now. Therefore, focused technologies will stand their ground in the repertoire of technologies, especially in medicine. In resequencing data, every position of the sequence is associated with a vector of numbers: meta information and uncertainty estimations. The tools developed in ResqMi for the Affymetrix technology may be obsolete, the rationale behind their use is not. New applications in this field can use the concepts tested in ResqMi. Extensions on this point towards the integration of data analysis tools and genome browsers. An additional chance is going beyond limitations of the current approach, as for example multiple reference sequences can be used.

In the long run, the integration of many of the described methods and concepts in order to form more generic and more efficient tools is conceivable. Enhancements in user interfaces, methods and data basis will allow and require modular, but powerful tools that can work on huge, heterogeneous datasets. This will also require advances in data management and storage. A further, very important point is collaborative analysis of datasets. Even enhanced by powerful visual analytics tools, the cognitive capabilities of human are limited. Therefore, collaboration, especially in parallel, of local and non-local scientists is necessary to explore and analyze huge datasets

Research and development of new user interfaces already made new usage concepts feasible. As slim but powerful tablet computing devices are becoming available, accompanied by ubiquitous network access, visual analytics systems are being developed for these platforms. The technology curve predicts that most the ideas that are now developed with much enthusiasm will not stand the test of everyday use. However there might be niches in which graph based visual analytics can profit from mobile devices with touch screens. The ubiquitous network access is useful for collaborative approaches. Multi-touch displays allow interesting usage concepts, which can facilitate handling of graphs, in collaborate applications, or as a remote control for large visualizations, e.g. on power walls.

The usefulness of visualization efforts in the field of systems biology heavily depends on data availability. Research in this field could therefore be enhanced by establishing public data repositories for biological graphs and networks. While there are many sources of biological pathways, and some authoritative sources of biological models that are actively curated, a general collection of biological graphs would be helpful. However today, this data is separated from each other and other sources, like interaction data, gene and protein annotations and expression data. Careful modeling, reporting standards as used in the microarray community (and many others) strong data mining features would be required. The reward would be comprehensive knowledge base of biological data, which is also rich in cross links – a further step taken towards the goals of both systems biology and integrative bioinformatics.

Similar data consolidation and publication initiatives are necessary for metabolomics data. The wide spread publication of metabolomics data in public repositories could help the development of tools and enhance research. The advances in this field will eventually allow to measure compounds on a whole-metabolome level with high confidence and reproducibility. This data needs to be integrated with expression data and biological networks and must therefore be publicly available. The MIAMET [16] standard is a first step towards this direction. First databases exist, but much work needs to be done to collect and integrate data.

All these approaches can be used to further enhance data visualization and analysis of gene expression and similar data. Constant improvements in current and newly emerging more efficient technologies emphasize the importance of scalable and extensible techniques for visual analysis and exploration. This thesis has presented several starting points for dealing with this.

A. MGV Key Shortcuts

Basic Keystrokes

Keystroke	Action	Operation
F1	Help	Open help window
F2	Lock/unlock	Make nodes fixed or movable and resizable
F3	Node inspector	Show node inspector, edit and search for nodes and edges
F4	Show/hide title	Show or hide the title of the graph
F5	Reset view	Set zoom level to 100 % and reset the size of all nodes to default
F6	Fit to width	Set zoom level in order to fit make the graph as wide as the window
F7	Fit to frame	Fit the graph into the width and height of the window
F8	Reset nodes	Scale all nodes to default size
F9	Show/hide labels	Hide or show the node labels
0	Reset zoom	Set zoom level to 100%
Home	Home	Go to the top of the graph
End	End	Go to the bottom of the graph
PgUp	Page up	Move up one page
PgDown	Page down	Move down one page
Spacebar	Next renderer	Switch to next renderer set
Ctrl + N	New graph	Open new graph viewer window
Ctrl + S	Save graph	Save the graph to MGV file
Ctrl + L	Load graph	Load the graph from MGV file

Selection Keystrokes

Keystroke	Action	Operation
Ctrl + A	Select all	Select all nodes
Esc	Clear selection	Clear selection
Del	Hide	Hide selected nodes
Ctrl + Del	Delete	Permanently delete selected nodes
Ctrl + G	Group	Create node group from selected nodes
Ctrl + M	Merge	Merge selected nodes to single node, remove selected nodes
Shift + Ctrl + M	Join	Join selected nodes to single node, keep selected nodes
Ctrl + →	Move	Move node 1 pixel to the right. Same for ←,↑,↓
Shift + Ctrl + →	Move	Move node 10 pixel to the right. Same for ←,↑,↓
H/J	Arrange	Horizontally/vertically arrange nodes
Ctrl + H/Ctrl + J	Arrange	Horizontally/vertically arrange nodes, align with left/topmost node

Adding Nodes

The following keystrokes add a single new node (or multiple nodes as described) of the described role to the graph at the position of the mouse pointer.

Keystroke	Role	Add new Node
N	Node	Generic Node
T	Note	Empty Note
P	Probe	Probe (selectable)
Ctrl + R	any	Open create new node dialog
Shift + M	Macromolecule	SBGN glyph
Shift + S	Simple Chemical	SBGN glyph
Shift + N	Nucleid acid feature	SBGN glyph
A	Association	SBGN glyph
P	Process	SBGN glyph
O	Omitted Process	SBGN glyph
D	Dissociation	SBGN glyph
Shift+Ctrl+ P	several	Add a new reaction (process, substrate, product, enzyme)

B. Publications

B.1. Articles

2008

- *Stephan Symons*, Kirsten Weber, Michael Bonin, and Kay Nieselt. *ResqMi - a versatile algorithm and software for resequencing microarrays.* In A. Beyer and M. Schroeder, editors: Proceedings of the German Conference on Bioinformatics 2008, pp 10-20.

- Hinnak Northoff, *Stephan Symons*, Derek Zieker, Eva V. Schaible, Katharina Schäfer, Stefanie Thoma, Markus Löffler, Asghar Abbasi, Perikles Simon, Andreas M. Niess, Elvira Fehrenbach. *Gender- and menstrual phase dependent regulation of inflammatory gene expression in response to aerobic exercise.* Exerc Immunol Rev. 2008; 14:86-103

2009

- Christina Pfannenberg, Ingmar Königsrainer, Philip Aschoff, Mehmet Ö. Öksüz, Derek Zieker, Stefan Beckert, *Stephan Symons*, Kay Nieselt, Jörg Glatzle, Claus v. Weyhern, Björn L. Brücher, Claus D. Claussen, Alfred Königsrainer. *F-FDG-PET/CT to Select Patients with Peritoneal Carcinomatosis for Cytoreductive Surgery and Hyperthermic Intraperitoneal Chemotherapy.* Ann Surg Oncol. 2009 16(5): 1295-1303

2010

- *Stephan Symons*, Christian Zipplies, Florian Battke, and Kay Nieselt. *Integrative Systems Biology Visualization with Mayday.* Journal of Integrative Bioinformatics 2010, 7:3

- Florian Battke, *Stephan Symons*, and Kay Nieselt. *Mayday – Integrative analytics for expression data.* BMC Bioinformatics 2010, 11:1

- Markus Löffler, Derek Zieker, Jürgen Weinreich, Stefan Löb, Ingmar Königsrainer, *Stephan Symons*, Sarah Bühler, Alfred Königsrainer, Hinnak Northoff, and Stefan Beckert. *Wound fluid lactate concentration: a helpful marker for diagnosing soft-tissue infection in diabetic foot ulcers? Preliminary findings.* Diabetic Medicine 2011, 28(2):175-8

- Ulrich Sommer, Andrea Schmitt, Markus Heck, Evelin Schaeffer, Markus Fendt, Mathias Zink, Kay Nieselt, *Stephan Symons*, Georg Petroianu, Anja Lex, Mario Herrera-Marschitz, Rainer Spanagel, Peter Falkai, Peter Gebicke-Haerter. *Differential expression of presynaptic genes in a rat model of postnatal hypoxia: relevance to schizophrenia.* . European archives of psychiatry and clinical neuroscience 2010 260: 81-89

2011

- *Stephan Symons*, and Kay Nieselt. *MGV: A Generic Graph Viewer for Comparative Omics Data.* Bioinformatics 2011, to appear

- Florian Battke, *Stephan Symons*, Alexander Herbig and Kay Nieselt. *GaggleBridge: Collaborative data analysis.* Submitted to Bioinformatics

B.2. Posters & Talks

2006

- *Stephan Symons*, Florian Battke, Janko Dietzsch, Matthias Zschunke, and Kay Nieselt. *Automated Processing and Machine Learning Tools for Mayday.* German Conference on Bioinformatics 2006

- Matthias Zschunke, Katrin Deubel, *Stephan Symons*, Janko Dietzsch, and Kay Nieselt. *FAGE and VEGA: two new tools for functional and epigenomic expression analysis in Mayday* . German Conference on Bioinformatics 2006

2007

- *Stephan Symons*, Christian Schillinger, Janko Dietzsch, Florian Battke, and Kay Nieselt. *GeneMining in Mayday, a feature selection framework for binary classification.* German Conference on Bioinformatics 2007

2008

- Florian Battke, *Stephan Symons*, Michael Piechotta, Philipp Bruns, Karin Zimmermann, and Kay Nieselt. *Mayday – Microarray data analysis.* German Conference on Bioinformatics 2008

2009

- Florian Battke, *Stephan Symons*, and Kay Nieselt. *Mayday and RLink – Integrated Expression Analysis.* German Conference on Bioinformatics 2009

- Florian Battke, *Stephan Symons*, and Kay Nieselt. *Mayday RLink – The best of two worlds.* Presentation at useR 2009

2010

- *Stephan Symons*, Florian Battke, Christian Zipplies, and Kay Nieselt. *Mayday - Integrative Visual Analytics for Transcriptomics*. VizBi 2010

Bibliography

[1] Affymetrix, Inc. CustomSeq Resequencing Array Base Calling Algorithm version 2.0: Performance in homozygous and heterozygous SNP detection. Technical report, Affymetrix, Inc, 2006.

[2] T. J. Albert et al. Mutation discovery in bacterial genomes: metronidazole resistance in helicobacter pylori. *Nature Methods*, 2(12):951–953, 2005.

[3] C. Alvarez-Ortega and C. Harwood. Responses of Pseudomonas aeruginosa to low oxygen indicate that growth in the cystic fibrosis lung is by aerobic respiration. *Molecular Microbiology*, 65(1):153, 2007.

[4] J. C. Alwine, D. J. Kemp, and G. R. Stark. Method for detection of specific RNAs in agarose gels by transfer to diazobenzyloxymethyl-paper and hybridization with DNA probes. *PNAS*, 74(12):5350–4, Dec. 1977.

[5] F. Anscombe. Graphs in statistical analysis. *The American Statistician*, 27(1):17–21, 1973.

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

[7] S. Bachthaler and D. Weiskopf. Continuous scatterplots. *IEEE Transactions on Visualization and Computer Graphics*, pages 1428–1435, 2008.

[8] G. Bader, E. Brauner, et al. BioPAX–Biological Pathways Exchange language. *BioPAX Workgroup*, 2005.

[9] W. Bains and G. C. Smith. A Novel Method for Nucleic Acid Sequence Determination. *J Theor Biol*, 135:303–307, 1988.

[10] A. Barabási and Z. Oltvai. Network biology: understanding the cell's functional organization. *Nature Reviews Genetics*, 5(2):101–113, 2004.

[11] F. Battke and K. Nieselt. Mayday SeaSight: Combined Analysis of Deep Sequencing and Microarray Data. *PLoS ONE*, 6(1):e16345, Jan. 2011.

[12] F. Battke, S. Symons, and K. Nieselt. Mayday–integrative analytics for expression data. *BMC Bioinformatics*, 11(1):121, Jan. 2010.

[13] D. Beck, M. Pinsk, and S. Kastner. Symmetry perception in humans and macaques. *Trends in cognitive sciences*, 9(9):405–406, 2005.

[14] M. Y. Becker and I. Rojas. A graph layout algorithm for drawing metabolic pathways. *Bioinformatics*, 17(5):461–467, 2001.

[15] D. Bentley. Whole-genome re-sequencing. *Current opinion in genetics & development*, 16(6):545–552, 2006.

[16] R. Bino, R. Hall, O. Fiehn, J. Kopka, K. Saito, J. Draper, B. Nikolau, P. Mendes, U. Roessner-Tunali, M. Beale, et al. Potential of metabolomics as a functional genomics tool. *Trends in Plant Science*, 9(9):418–425, 2004.

[17] B. Bolstad, R. Irizarry, M. Åstrand, and T. Speed. A comparison of normalization methods for high density oligonucleotide array data based on variance and bias. *Bioinformatics*, 19(2):185, 2003.

[18] D. Borland and R. Taylor II. Rainbow color map (still) considered harmful. *IEEE Computer Graphics and Applications*, pages 14–17, 2007.

[19] C. M. L. S. Bouton. DRAGON View: information visualization for annotated microarray data. *Bioinformatics*, 18(2):323–324, Feb. 2002.

[20] U. Brandes, M. Eiglsperger, I. Herman, M. Himsolt, and M. Marshall. Graphml progress report structural layer proposal. In P. Mutzel, M. Jünger, and S. Leipert, editors, *Graph Drawing*, volume 2265 of *Lecture Notes in Computer Science*, pages 109–112. Springer Berlin / Heidelberg, 2002.

[21] A. Brazma, P. Hingam, J. Quackenbush, G. Sherlock, and P. S. et al. Minimum Information about a microarray experiment (MIAME) -toward standards for microarray data. *Nature Genetics*, 29:365–371, 12 2001.

[22] A. Brazma, H. Parkinson, U. Sarkans, M. Shojatalab, J. Vilo, N. Abeygunawardena, E. Holloway, M. Kapushesky, P. Kemmeren, G. Lara, et al. ArrayExpress—a public repository for microarray gene expression data at the EBI. *Nucleic Acids Research*, 31(1):68, 2003.

[23] R. Breitling, P. Armengaud, A. Amtmann, and P. Herzyk. Rank products: a simple, yet powerful, new method to detect differentially regulated genes in replicated microarray experiments. *FEBS letters*, 573(1-3):83–92, Aug. 2004.

[24] J. Bullard, E. Purdom, K. Hansen, and S. Dudoit. Evaluation of statistical methods for normalization and differential expression in mrna-seq experiments. *BMC Bioinformatics*, 11(1):94, 2010.

[25] M. Burch, F. Beck, and S. Diehl. Timeline trees: visualizing sequences of transactions in information hierarchies. In *Proceedings of the Working Conference on Advanced Visual Interfaces*, pages 75–82. ACM, New York, NY, USA, 2008.

[26] L. Byron and M. Wattenberg. Stacked graphs–geometry & aesthetics. *IEEE Transactions on Visualization and Computer Graphics*, pages 1245–1252, 2008.

[27] S. Card, J. Mackinlay, and B. Shneiderman, editors. *Readings in Information Visualization: Using Vision to Think*. Morgan Kaufmann, San Francisco, CA, USA, 1999.

[28] L. Carroll. *Alice's Adventures in Wonderland*. Project Gutenberg, 1865.

[29] R. Caspi, H. Foerster, C. A. Fulcher, P. Kaipa, M. Krummenacker, M. Latendresse, S. Paley, S. Y. Rhee, A. G. Shearer, C. Tissier, T. C. Walk, P. Zhang, and P. D. Karp. The MetaCyc Database of metabolic pathways and enzymes and the BioCyc collection of Pathway/Genome Databases. *Nucleic Acids Research*, 36:D623–31, Jan 2008.

[30] E. G. Cerami, B. E. Gross, et al. Pathway Commons, a web resource for biological pathway data. *Nucleic Acids Research*, 39:D685–D690, 2010.

[31] D. Chandran, F. Bergmann, and H. Sauro. Athena: Modular CAM/CAD Software for Synthetic Biology. *Arxiv preprint arXiv:0902.2598*, 2009.

[32] H. Chipman, T. J. Hastie, and R. Tibshirani. *Clustering microarray data*, chapter 4, pages 159–200. Chapman & Hall/CRC, 2003.

[33] C. Choi, R. Munch, et al. SYSTOMONAS–an integrated database for systems biology analysis of Pseudomonas. *Nucleic Acids Research*, 35(Database issue):D533, 2007.

[34] R. M. Clark et al. Common Sequence Polymorphisms Shaping Genetic Diversity in Arabidopsis thaliana. *Science*, 317:338–342, 2007.

[35] W. Claudino, A. Quattrone, L. Biganzoli, M. Pestrin, I. Bertini, and A. Di Leo. Metabolomics: available results, current research projects in breast cancer, and future applications. *Journal of Clinical Oncology*, 25(19):2840, 2007.

[36] W. Cleveland and R. McGill. Graphical perception: Theory, experimentation, and application to the development of graphical methods. *Journal of the American Statistical Association*, 79(387):531–554, 1984.

[37] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms.* MIT Press, 20.

[38] G. E. Crooks, G. Hon, J.-M. Chandonia, and S. E. Brenner. WebLogo: a sequence logo generator. *Genome Research*, 14(6):1188–90, June 2004.

[39] A. A. Cuellar, C. M. Lloyd, P. F. Nielsen, D. P. Bullivant, D. P. Nickerson, and P. J. Hunter. An Overview of CellML 1.1, a Biological Model Description Language. *Simulation*, 79(12):740–747, Dec. 2003.

[40] A. Culhane, G. Perrière, and D. Higgins. Cross-platform comparison and visualisation of gene expression data using co-inertia analysis. *BMC Bioinformatics*, 4(1):59, 2003.

[41] D. J. Cutler et al. High-Throughput Variation Detection and Genotyping Using Microarrays. *Genome Research*, 11:1913–1925, 2001.

[42] L. Davignon et al. Use of Resequencing Oligonucleotide Microarrays for Identification of Streptococcus pyrogens and Associated Antibiotic Resistance Determinants. *Journal of Clinical Microbiology*, 43(11):5690–5695, 2005.

[43] E. Demir, M. P. Cary, et al. The BioPAX community standard for pathway data sharing. *Nature Biotechnology*, 28(9):935–942, Sept. 2010.

[44] K. Deubel. Funktionelle Analyse von Genexpressionsdaten und Visualisierung beteiligter metabolischer Netzwerke. Diplomarbeit Bioinformatik, University of Tübingen, 2006.

[45] J. Dougherty, R. Kohavi, and M. Sahami. Supervised and unsupervised discretization of continuous features. In *Proceedings of the 12th International Conference on Machine Learning*, pages 194–202. Morgan Kaufmann, San Francisco, CA, USA, 1995.

[46] M. Douma, G. Ligierko, O. Ancuta, P. Gritsai, and S. Liu. SpicyNodes: Radial Layout Authoring for the General Public. *Visualization and Computer Graphics, IEEE Transactions on*, 15(6):1089–1096, 2009.

[47] S. Draghici. *Data Analysis Tools for DNA Microarrays.* Chapman & Hall/CRC, 2003.

[48] P. Eades and K. Sugiyama. How to draw a directed graph. *Journal of Information Processing*, 13(4):424–437, 1991.

[49] P. Eades and N. C. Wormald. Edge crossings in drawings of bipartite graphs. *Algorithmica*, 11:379–403, 1994. 10.1007/BF01187020.

[50] R. Edgar, M. Domrachev, and A. Lash. Gene Expression Omnibus: NCBI gene expression and hybridization array data repository. *Nucleic Acids Research*, 30(1):207, 2002.

[51] K. Eilbeck, S. E. Lewis, C. J. Mungall, M. Yandell, L. Stein, R. Durbin, and M. Ashburner. The Sequence Ontology: a tool for the unification of genome annotations. *Genome Biology*, 6(5):R44, Jan. 2005.

[52] M. Eisen, P. Spellman, P. Brown, and D. Botstein. Cluster analysis and display of genome-wide expression patterns. *PNAS*, 95(25):14863, 1998.

[53] Ensembl.org. Assembly and Genebuild Statistics for Homo sapiens. http://www.ensembl.org/Homo_sapiens/Info/StatsTable?db=core, 04 2011.

[54] R. D. Finn, J. Mistry, J. Tate, P. Coggill, A. Heger, J. E. Pollington, O. L. Gavin, P. Gunasekaran, G. Ceric, K. Forslund, L. Holm, E. L. L. Sonnhammer, S. R. Eddy, and A. Bateman. The Pfam protein families database. *Nucleic Acids Research*, 38:D211–22, Jan. 2010.

[55] P. Flicek, M. R. Amode, D. Barrell, K. Beal, S. Brent, Y. Chen, P. Clapham, G. Coates, S. Fairley, S. Fitzgerald, L. Gordon, M. Hendrix, T. Hourlier, N. Johnson, A. Kähäri, D. Keefe, S. Keenan, R. Kinsella, F. Kokocinski, E. Kulesha, P. Larsson, I. Longden, W. McLaren, B. Overduin, B. Pritchard, H. S. Riat, D. Rios, G. R. S. Ritchie, M. Ruffier, M. Schuster, D. Sobral, G. Spudich, Y. A. Tang, S. Trevanion, J. Vandrovcova, A. J. Vilella, S. White, S. P. Wilder, A. Zadissa, J. Zamora, B. L. Aken, E. Birney, F. Cunningham, I. Dunham, R. Durbin, X. M. Fernández-Suarez, J. Herrero, T. J. P. Hubbard, A. Parker, G. Proctor, J. Vogel, and S. M. J. Searle. Ensembl 2011. *Nucleic Acids Research*, 39:D800–806, 2010.

[56] R. W. Floyd. Algorithm 97: Shortest path. *Commununicatinos of the ACM*, 5:345, June 1962.

[57] T. M. J. Fruchterman and E. M. Reingold. Graph drawing by force-directed placement. *Software: Practice and Experience*, 21(11):1129–1164, 1991.

[58] B. Fry. *Visualizing data: exploring and explaining data with the processing environment.* O'Reilly Media, Sebastopol, CA, USA, 2007.

[59] A. Funahashi, Y. Matsuoka, A. Jouraku, M. Morohashi, N. Kikuchi, and H. Kitano. CellDesigner 3.5: a versatile modeling tool for biochemical networks. *Proceedings of the IEEE*, 96(8):1254–1265, 2008.

[60] W. Fury, F. Batliwalla, P. Gregersen, and W. Li. Overlapping probabilities of top ranking gene lists, hypergeometric distribution, and stringency of gene selection criterion. In *Engineering in Medicine and Biology Society, 2006. EMBS'06. 28th Annual International Conference of the IEEE*, volume 1, pages 5531–5534. IEEE, New York, NY, USA, Jan. 2006.

[61] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design patterns: elements of reusable object-oriented software.* Addison-Wesley Reading, MA, USA, 1995.

[62] F. García-Alcalde, F. García-Lopez, et al. Paintomics: a web based tool for the joint visualization of transcriptomics and metabolomics data. *Bioinformatics*, 27(1):137–139, Nov. 2010.

[63] N. Gehlenborg, J. Dietzsch, and K. Nieselt. A Framework for Visualization of Microarray Data and Integrated Meta Information. *Information Visualization*, 4(3):164–175, June 2005.

[64] R. Gentleman, V. Carey, et al. Bioconductor: open software development for computational biology and bioinformatics. *Genome Biology*, 5(10):R80, 2004.

[65] A. Gnirke, A. Melnikov, J. Maguire, P. Rogov, E. LeProust, W. Brockman, T. Fennell, G. Giannoukos, S. Fisher, C. Russ, et al. Solution hybrid selection with ultra-long oligonucleotides for massively parallel targeted sequencing. *Nature biotechnology*, 27(2):182, 2009.

[66] T. Golub, D. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. Mesirov, H. Coller, M. Loh, J. Downing, M. Caliguri, C. Bloomfield, and E. Lander. Molecular Classification of Cancer: Class Discovery and Class Prediction by Gene Expression Monitoring. *Science*, 286:531–537, 1999.

[67] R. Goodacre, S. Vaidyanathan, W. Dunn, G. Harrigan, and D. Kell. Metabolomics by numbers: acquiring and understanding global metabolite data. *TRENDS in Biotechnology*, 22(5):245–252, 2004.

[68] J. Gross and J. Yellen. *Graph Theory and Its Applications.* CRC Press, 1998.

[69] J. G. Hacia. Resequencing and mutational analysis using oligonucleotide microarrays. *Nature Genetics*, 21:42–47, 1999.

[70] O. Harismendy, P. Ng, R. Strausberg, X. Wang, T. Stockwell, K. Beeson, N. Schork, S. Murray, E. Topol, S. Levy, et al. Evaluation of next generation sequencing platforms for population targeted sequencing studies. *Genome Biol*, 10(3):R32, 2009.

[71] M. Harrower and C. Brewer. Colorbrewer. org: an online tool for selecting colour schemes for maps. *Cartographic Journal*, 40(1):27–37, 2003.

[72] Y. Hassan-Montero and V. Herrero-Solana. Improving tag-clouds as visual information retrieval interfaces. In *International Conference on Multidisciplinary Information Sciences and Technologies*, pages 25–28. Citeseer, 2006.

[73] S. Heber, M. Alekseyev, S.-H. Sze, H. Tang, and P. a. Pevzner. Splicing graphs and EST assembly problem. *Bioinformatics*, 18 Suppl 1:S181–8, Jan. 2002.

[74] J. Heer and M. Agrawala. Multi-scale banking to 45 degrees. *IEEE Transactions on Visualization and Computer Graphics*, pages 701–708, 2006.

[75] L. J. Heyer, S. Kruglyak, and S. Yooseph. Exploring expression data: identification and analysis of coexpressed genes. *Genome Research*, 9(11):1106–15, Nov 1999.

[76] J. Hintze and R. Nelson. Violin plots: a box plot-density trace synergism. *American Statistician*, 52(2):181–184, 1998.

[77] D. Holten and J. van Wijk. A user study on visualizing directed edges in graphs. In *Proceedings of the 27th International Conference on Human Factors in computing systems*, pages 2299–2308. ACM, New York, NY, USA, 2009.

[78] C. E. Horak and M. Snyder. ChIP-chip: a genomic approach for identifying transcription factor binding sites. *Methods in Enzymology*, 350:469–83, Jan. 2002.

[79] Z. Hu, D. M. Ng, T. Yamada, C. Chen, S. Kawashima, J. Mellor, B. Linghu, M. Kanehisa, J. M. Stuart, and C. DeLisi. VisANT 3.0: new modules for pathway visualization, editing, prediction and construction. *Nucleic Acids Research*, 35:W625–32, July 2007.

[80] W. Huber, A. Von Heydebreck, H. Sültmann, A. Poustka, and M. Vingron. Variance stabilization applied to microarray data calibration and to the quantification of differential expression. *Bioinformatics*, 18(suppl 1):S96, 2002.

[81] M. Hucka, F. Bergmann, S. Hoops, S. Keating, S. Sahle, J. Schaff, L. Smith, and D. Wilkinson. The Systems Biology Markup Language (SBML): Language Specification for Level 3 Version 1 Core. *Nature Precedings*, Oct. 2010.

[82] Illumina, Inc. Genome Analyzer IIx. Web Page, May 2011.

[83] A. Inselberg. *Parallel Coordinates - Visual Multidimensional Goemetry and Its Applications*. Springer, 2009.

[84] F. Jacob and J. Monod. Genetic regulatory mechanisms in the synthesis of proteins. *Journal of Molecular Biology*, 3:318–56, June 1961.

[85] B. H. Junker, C. Klukas, et al. VANTED: a system for advanced data analysis and visualization in the context of biological networks. *BMC Bioinformatics*, 7(1):109, Jan. 2006.

[86] G. Jäger. A 3D-Visual Analytics Framework forExpression Data. Master's thesis, Universtiy of Tübingen, 2010.

[87] K. Kadota, Y. Nakai, and K. Shimizu. A weighted average difference method for detecting differentially expressed genes from microarray data. *Algorithms for Molecular Biology*, 3:8, Jan. 2008.

[88] T. Kamada and S. Kawai. An algorithm for drawing general undirected graphs. *Information processing letters*, 31(1):7–15, 1989.

[89] N. Kambhatla and T. Leen. Dimension reduction by local principal component analysis. *Neural Computation*, 9(7):1493–1516, 1997.

[90] M. Kanehisa. KEGG: Kyoto Encyclopedia of Genes and Genomes. *Nucleic Acids Research*, 28(1):27–30, Jan. 2000.

[91] M. Kanehisa, M. Araki, S. Goto, M. Hattori, M. Hirakawa, M. Itoh, T. Katayama, S. Kawashima, S. Okuda, T. Tokimatsu, and Y. Yamanishi. Kegg for linking genomes to life and the environment. *Nucleic Acids Research*, 36:D480–4, Jan 2008.

[92] P. Kapranov, A. T. Willingham, and T. R. Gingeras. Genome-wide transcription and the implications for genomic organization. *Nature Reviews Genetics*, 8(6):413–23, June 2007.

[93] P. Karp and S. Paley. Automated Drawing of Metabolic Pathways. In *Proceedings of the 3rd International Conference on Bioinformatics and Genome Research*, pages 225–238, 1994.

[94] O. Kaser and D. Lemire. Tag-cloud drawing: Algorithms for cloud visualization. *Arxiv preprint cs/0703109*, 2007.

[95] D. Keim, F. Mansmann, J. Schneidewind, J. Thomas, and H. Ziegler. Visual analytics: Scope and challenges. In S. Simoff, M. Böhlen, and A. Mazeika, editors, *Visual Data Mining*, volume 4404 of *Lecture Notes in Computer Science*, pages 76–90. Springer Berlin / Heidelberg, 2008.

[96] D. A. Keim, F. Mansmann, J. Schneidewind, and H. Ziegler. Challenges in Visual Data Analysis. In *International Conference on Information Visualisation*, pages 9–16, Los Alamitos, CA, USA, 2006. IEEE, New York, NY, USA.

[97] T. Kelder, B. Conklin, C. Evelo, and A. Pico. Finding the right questions: exploratory pathway analysis to enhance biological discovery in large datasets. *PLoS Biology*, 8(8):e1000472, 2010.

[98] A. Keller, C. Backes, M. Al-Awadhi, A. Gerasch, J. Küntzer, O. Kohlbacher, M. Kaufmann, and H. Lenhof. GeneTrailExpress: a web-based pipeline for the statistical evaluation of microarray experiments. *BMC Bioinformatics*, 9(1):552, 2008.

[99] R. Keller. Statistische Anreicherungsmethoden und -analysen auf azyklischen Graphen. Diplomarbeit Bioinformatik, Universtiy of Tübingen, 2010.

[100] S. Kerrien, S. Orchard, L. Montecchi-Palazzi, B. Aranda, A. Quinn, N. Vinod, G. Bader, I. Xenarios, J. Wojcik, D. Sherman, M. Tyers, J. Salama, S. Moore, A. Ceol, A. Chatr-aryamontri, M. Oesterheld, V. Stumpflen, L. Salwinski, J. Nerothin, E. Cerami, M. Cusick, M. Vidal, M. Gilson, J. Armstrong, P. Woollard, C. Hogue, D. Eisenberg, G. Cesareni, R. Apweiler, and H. Hermjakob. Broadening the horizon - level 2.5 of the HUPO-PSI format for molecular interactions. *BMC Biology*, 5(1):44, 2007.

[101] K. Kim, W. Javed, C. Williams, N. Elmqvist, and P. Irani. Hugin: a framework for awareness and coordination in mixed-presence collaborative information visualization. In *ACM International Conference on Interactive Tabletops and Surfaces*, pages 231–240. ACM, New York, NY, USA, 2010.

[102] C. Klukas and F. Schreiber. Dynamic exploration and editing of KEGG pathway diagrams. *Bioinformatics*, 23(3):344, 2007.

[103] T. A. Knijnenburg, J. H. de Winde, et al. Exploiting combinatorial cultivation conditions to infer transcriptional regulation. *BMC Genomics*, 8:25, 2007.

[104] J. Köhler, J. Baumbach, et al. Graph-based analysis and visualization of experimental results with ONDEX. *Bioinformatics*, 22(11):1383–90, June 2006.

[105] K. Kohn. Molecular interaction map of the mammalian cell cycle control and DNA repair systems. *Molecular Biology of the Cell*, 10(8):2703, 1999.

[106] K. Kojima, M. Nagasaki, E. Jeong, M. Kato, and S. Miyano. An efficient grid layout algorithm for biological networks utilizing various biological attributes. *BMC Bioinformatics*, 8:76, 2007.

[107] N. Kono, K. Arakawa, et al. Pathway Projector: Web-Based Zoomable Pathway Browser Using KEGG Atlas and Google Maps API. *PloS One*, 4(11):e7710, 2009.

[108] Y. Koren. On spectral graph drawing. In T. Warnow and B. Zhu, editors, *Computing and Combinatorics*, volume 2697 of *Lecture Notes in Computer Science*, pages 496–508. Springer Berlin / Heidelberg, 2003.

[109] A. Koschmieder, K. Zimmermann, S. Triß l, T. Stoltmann, and U. Leser. Tools for managing and analyzing microarray data. *Briefings in Bioinformatics*, page to appear, 2011.

[110] N. Kumar, N. Lolla, E. Keogh, S. Lonardi, and C. A. Ratanamahatana. Time-series bitmaps: a practical visualization tool for working with large time series databases. In *SIAM 2005 Data Mining Conference*, pages 531–535. SIAM, 2005.

[111] D. S. Latchman. *Gene Regulaton.* Taylor & Francis, 2005.

[112] K. Lê Cao, P. Martin, C. Robert-Granié, and P. Besse. Sparse canonical methods for biological data integration: application to a cross-platform study. *BMC Bioinformatics*, 10(1):34, 2009.

[113] N. Le Novère. Model storage, exchange and integration. *BMC Neuroscience*, 7 Suppl 1:S11, Jan. 2006.

[114] N. Le Novere, B. Bornstein, A. Broicher, M. Courtot, M. Donizelli, H. Dharuri, L. Li, H. Sauro, M. Schilstra, B. Shapiro, et al. BioModels Database: a free, centralized database of curated, published, quantitative kinetic models of biochemical and cellular systems. *Nucleic Acids Research*, 34(suppl 1):D689, 2006.

[115] N. Le Novère, M. Hucka, H. Mi, S. Moodie, F. Schreiber, A. Sorokin, E. Demir, K. Wegner, M. I. Aladjem, S. M. Wimalaratne, F. T. Bergman, R. Gauges, P. Ghazal, H. Kawaji, L. Li, Y. Matsuoka, A. Villéger, S. E. Boyd, L. Calzone, M. Courtot, U. Dogrusoz, T. C. Freeman, A. Funahashi, S. Ghosh, A. Jouraku, S. Kim, F. Kolpakov, A. Luna, S. Sahle, E. Schmidt, S. Watterson, G. Wu, I. Goryanin, D. B. Kell, C. Sander, H. Sauro, J. L. Snoep, K. Kohn, and H. Kitano. The Systems Biology Graphical Notation. *Nature Biotechnology*, 27(8):735–41, Aug. 2009.

[116] C. Li, G. C. Tseng, and W. H. Wong. *Model-based analysis of oligonucleotide arrays and issues in cDNA microarray analyis*, chapter 1, pages 1–34. Chapman & Hall/CRC, 2003.

[117] H. Li and N. Homer. A survey of sequence alignment algorithms for next-generation sequencing. *Briefings in Bioinformatics*, 11(5):473, 2010.

[118] W. Li and H. Kurata. A grid layout algorithm for automatic drawing of biochemical networks. *Bioinformatics*, 21(9):2036, 2005.

[119] B. Lin et al. Broad-spectrum respiratory tract pathogen identification using resequencing DNA microarrays. *Genome Research*, 16:527–535, 2006.

[120] J. Lin, E. Keogh, S. Lonardi, and B. Chiu. A symbolic representation of time series, with implications for streaming algorithms. In *Proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, page 11. ACM, New York, NY, USA, 2003.

[121] S. P. Lloyd. Least Squares Quantization in PCM. *IEEE Transactions on Information Theory*, 28:129–137, 1982.

[122] S. Lohmann, J. Ziegler, and L. Tetzlaff. Comparison of tag cloud layouts: Task-related performance and visual exploration. *Human-Computer Interaction–INTERACT 2009*, pages 392–404, 2009.

[123] S. Madeira and A. Oliveira. An evaluation of discretization methods for non-supervised analysis of time-series gene expression data. In *Instituto de Engenharia de Sistemas e Computadores Investigacao e Desenvolvimento, Technical Report*, volume 42. Citeseer, 2005.

[124] A. Maitra et al. The Human MitoChip: A high-throughput sequencing microarray for mitochondrial mutation detection. *Genome Research*, 14:812–819, 2004.

[125] A. P. Malanoski et al. Automated identification of multiple micro-organisms from resequencing DNA microarrays. *Nucleic Acids Research*, 34(18):5300–5311, 2006.

[126] M. Mann and O. N. Jensen. Proteomic analysis of post-translational modifications. *Nature Biotechnology*, 21(3):255–61, Mar. 2003.

[127] M. Margulies, M. Egholm, W. Altman, S. Attiya, J. Bader, L. Bemben, J. Berka, M. Braverman, Y. Chen, Z. Chen, et al. Genome sequencing in open microfabricated high density picoliter reactors. *Nature*, 437(7057):376, 2005.

[128] J. Marioni, C. Mason, S. Mane, M. Stephens, and Y. Gilad. RNA-seq: an assessment of technical reproducibility and comparison with gene expression arrays. *Genome Research*, 18(9):1509, 2008.

[129] V. D. Marks, S. J. Ho Sui, D. Erasmus, G. K. van der Merwe, J. Brumm, W. W. Wasserman, J. Bryan, and H. J. J. van Vuuren. Dynamics of the yeast transcriptome during wine fermentation reveals a novel fermentation stress response. *FEMS Yeast Research*, 8(1):35–52, Feb. 2008.

[130] L. Matthews, G. Gopinath, M. Gillespie, M. Caudy, D. Croft, B. de Bono, P. Garapati, J. Hemish, H. Hermjakob, B. Jassal, A. Kanapin, S. Lewis, S. Mahajan, B. May, E. Schmidt, I. Vastrik, G. Wu, E. Birney, L. Stein, and P. D'Eustachio. Reactome knowledgebase of human biological pathways and processes. *Nucleic Acids Research*, 37(suppl 1):D619–D622, 2009.

[131] P. Mendes, D. Camacho, and A. de la Fuente. Modelling and simulation for metabolomics data analysis. *Biochemical Society Transactions*, 33(6):1427–1429, 2005.

[132] G. Michal et al. *Biochemical pathways: an atlas of biochemistry and molecular biology.* Wiley, New York, NY, USA, 1999.

[133] S. K. Mithani et al. Mitochondrial Resequencing Arrays Detect Tumor-Specific Mutations in Salivary Rinses of Patients with Head and Neck Cancer. *Cini Cancer Res*, 13:7335–7340, 2007.

[134] C. S. Möller-Levet, K.-H. Cho, and O. Wolkenhauer. Microarray data clustering based on temporal variation: FCV with TSD preclustering. *Applied Bioinformatics*, 2(1):35–45, Jan. 2003.

[135] S. Moodie, N. Le Novere, E. Demir, H. Mi, and F. Schreiber. Systems biology graphical notation: process description language level 1. *Nature Precedings*, 2010.

[136] A. Mortazavi, B. Williams, K. McCue, L. Schaeffer, and B. Wold. Mapping and quantifying mammalian transcriptomes by rna-seq. *Nature methods*, 5(7):621–628, 2008.

[137] M. Munz. Interaktive Visualisierung von systembiologischen Daten. Bachelor's thesis, University of Tübingen, 2011.

[138] H. Neuweger, M. Persicke, et al. Visualizing Post Genomics Data-sets on customized Pathway Maps by ProMeTra- aeration-dependent gene expression and metabolism of Corynebacterium glutamicum as an example. *BMC Systems Biology*, 3(1):82, 2009.

[139] M. Newman. Modularity and community structure in networks. *PNAS*, 103(23):8577, 2006.

[140] K. Nieselt, F. Battke, A. Herbig, P. Bruheim, A. Wentzel, O. y. M. Jakobsen, H. v. Sletta, M. T. Alam, M. E. Merlo, J. Moore, W. A. M. Omara, E. R. Morrissey, M. A. Juarez-Hermosillo, A. Rodríguez-García, M. Nentwich, L. Thomas, M. Iqbal, R. Legaie, W. H. Gaze, G. L. Challis, R. C. Jansen, L. Dijkhuizen, D. A. Rand, D. L. Wild, M. Bonin, J. Reuther, W. Wohlleben, M. C. M. Smith, N. J. Burroughs, J. F. Martín, D. A. Hodgson, E. Takano, R. Breitling, T. E. Ellingsen, and E. M. H. Wellington. The dynamic architecture of the metabolic switch in Streptomyces coelicolor. *BMC Genomics*, 11(1):10, Jan. 2010.

[141] A. Noack. Energy models for graph clustering. *Journal of Graph Algorithms and Applications*, 11(2):453–480, 2007.

[142] A. Noack. Modularity clustering is force-directed layout. *Physical Review E*, 79(2):026102, 2009.

[143] T. Ottmann and P. Widmayer. *Algorithmen und Datenstrukturen.* Spektrum Akademischer Verlag, 2002.

[144] A. R. Pico, T. Kelder, et al. WikiPathways: Pathway Editing for the People. *PLoS Biology*, 6(7):e184, 07 2008.

[145] J. Pinney, D. Westhead, and G. McConkey. Petri net representations in systems biology. *Biochemical Society Transactions*, 31(6):1513–1515, 2003.

[146] H. Purchase. Metrics for graph drawing aesthetics. *Journal of Visual Languages & Computing*, 13(5):501–516, 2002.

[147] R Development Core Team. *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria, 2010. ISBN 3-900051-07-0.

[148] M. E. Ritchie, J. Silver, A. Oshlack, M. Holmes, D. Diyagama, A. Holloway, and G. K. Smyth. A comparison of background correction methods for two-colour microarrays. *Bioinformatics*, 23(20):2700–2707, 2007.

[149] P. Robert and Y. Escoufier. A unifying tool for linear multivariate statistical methods: the rv-coefficient. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 25(3):257–265, 1976.

[150] J. Roberts. State of the art: Coordinated & multiple views in exploratory visualization. In *Coordinated and Multiple Views in Exploratory Visualization, 2007. CMV'07. Fifth International Conference on*, pages 61–71. IEEE, New York, NY, USA, 2007.

[151] M. Robinson and A. Oshlack. A scaling normalization method for differential expression analysis of RNA-seq data. *Genome Biol*, 11(3):R25, 2010.

[152] M. Ronaghi. Pyrosequencing sheds light on DNA sequencing. *Genome Research*, 11(1):3, 2001.

[153] A. Saeed, V. Sharov, J. White, J. Li, W. Liang, N. Bhagabati, J. Braisted, M. Klapa, T. Currier, M. Thiagarajan, et al. Tm4: a free, open-source system for microarray data management and analysis. *Biotechniques*, 34(2):374, 2003.

[154] N. Salomonis, K. Hanspers, et al. GenMAPP 2: new features and resources for pathway analysis. *BMC Bioinformatics*, 8(1):217, 2007.

[155] J. Salonen. Self-organising map based tag clouds - Creating Spatially Meaninful Representations of Tagging Data. In *Proceedings of 1st OPAALS workshop*, volume 2007, 2007.

[156] P. Saraiya, C. North, and K. Duca. Visualizing biochemical pathways: requirements analysis, systems evaluation and research agenda. *Information Visualization*, 4(3):191–205, 2005.

[157] E. Schadt, C. Li, B. Ellis, and W. Wong. Feature extraction and normalization algorithms for high-density oligonucleotide gene expression array data. *Journal of Cellular Biochemistry*, 84(s 37):120–125, 2001.

[158] C. Schillinger. Robuste Merkmalsselektion aus Bipartitionen von Microarray-Expressionsdaten mit Anwendungen auf aggregierte Tumordaten. Diplomarbeit Bioinformatik, University of Tübingen, 2007.

[159] T. D. Schneider and R. M. Stephens. Sequence Logos: a new way to display consensus sequences. *Nucleic Acids Research*, 18(20):6097–6100, 1990.

[160] T. D. Schneider, G. D. Stormo, L. Gold, and A. Ehrenfeucht. Information content of binding sites on nucleotide sequences. *Journal of Molecular Biology*, 188(3):415–31, Apr. 1986.

[161] E. Segel and J. Heer. Narrative Visualization: Telling Stories with Data. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1139–1148, 2010.

[162] P. Shannon, D. Reiss, R. Bonneau, and N. Baliga. The Gaggle: an open-source software system for integrating bioinformatics software and data sources. *BMC Bioinformatics*, 7(1):176, 2006.

[163] J. Shendure. The beginning of the end for microarrays? *Nature Methods*, 5(7):585–7, July 2008.

[164] L. Shi, L. Reid, W. Jones, R. Shippy, J. Warrington, S. Baker, P. Collins, F. De Longueville, E. Kawasaki, K. Lee, et al. The MicroArray Quality Control (MAQC) project shows inter- and intraplatform reproducibility of gene expression measurements. *Nature biotechnology*, 24(9):1151–1161, 2006.

[165] C. Sieber. Regulatorische Netzwerke mit Association Mining. Bachelor's thesis, University of Tübingen, 2008.

[166] A. K. Smilde, H. A. L. Kiers, S. Bijlsma, C. M. Rubingh, and M. J. van Erk. Matrix correlations for high-dimensional data: the modified RV-coefficient. *Bioinformatics*, 25(3):401–5, Feb. 2009.

[167] C. Smith, J. Elizabeth, G. O'Maille, R. Abagyan, and G. Siuzdak. XCMS: processing mass spectrometry data for metabolite profiling using nonlinear peak alignment, matching, and identification. *Analytical Chemistry*, 78(3):779–787, 2006.

[168] M. E. Smoot, K. Ono, et al. Cytoscape 2.8: new features for data integration and network visualization. *Bioinformatics*, 27(3):431–432, 2011.

[169] P. T. Spellman, G. Sherlock, M. Q. 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. *Molecular Biology of the Cell*, 9(12):3273–97, Dec. 1998.

[170] D. Stekel. *Microarray Bioinformatics.* Cambridge University Press, 2003.

[171] S. Stevens. On the theory of scales of measurement. *Science*, 103(2684):677–680, 1946.

[172] B. Stewart and L. Best. An Examination of Cleveland and McGill's Hierarchy of Graphical Elements. *Diagrammatic Representation and Inference*, pages 334–337, 2010.

[173] E. A. Stone and J. F. Ayroles. Modulated Modularity Clustering as an Exploratory Tool for Functional Genomic Inference. *PLoS Biology*, 5(5):e1000479, 2009.

[174] M. Streit, A. Lex, M. Kalkusch, K. Zatloukal, and D. Schmalstieg. Caleydo: connecting pathways and gene expression. *Bioinformatics*, 25(20):2760, 2009.

[175] A. Subramanian, P. Tamayo, V. Mootha, S. Mukherjee, B. Ebert, M. Gillette, A. Paulovich, S. Pomeroy, T. Golub, E. Lander, et al. Gene set enrichment analysis: a knowledge-based approach for interpreting genome-wide expression profiles. *PNAS*, 102(43):15545, 2005.

[176] I. M. Sulaiman et al. Evaluation of Affymetrix severe acute respiratory syndrome resequencing GeneChips in characterization of the genomes of two strains of Coronavirus infecting humans. *Appl. Env. Microbiol.*, 72:207–211, 2006.

[177] D. Swayne, D. Lang, A. Buja, and D. Cook. GGobi: Evolving from XGobi into an extensible framework for interactive data visualization. *Computational Statistics & Data Analysis*, 43(4):423–444, 2003.

[178] S. Symons et al. ResqMi - a versatile algorithm and software for Resequencing Microarrays. In A. Beyer and M. Schroeder, editors, *Proceedings of the German Conference on Bioinformatics 2008*, pages 10–20, 2008.

[179] S. Symons and K. Nieselt. MGV: A Generic Graph Viewer for Comparative Omics Data. *Bioinformatics*, to appear, 2011.

[180] S. Symons, C. Zipplies, F. Battke, and K. Nieselt. Integrative systems biology visualization with MAYDAY. *Journal of Integrative Bioinformatics*, 7(3), Jan. 2010.

[181] D. Szklarczyk, A. Franceschini, M. Kuhn, M. Simonovic, A. Roth, P. Minguez, T. Doerks, M. Stark, J. Muller, P. Bork, L. J. Jensen, and C. v. Mering. The STRING database in 2011: functional interaction networks of proteins, globally integrated and scored. *Nucleic Acids Research*, 39(suppl_1):D561–568, 2010.

[182] R. Tarjan. Depth-first search and linear grajh algorithms. In *Conference Record 1971 Twelfth Annual Symposium on Switching and Automata Theory*, pages 114–121. IEEE, New York, NY, USA, 1971.

[183] A. Thalamuthu, I. Mukhopadhyay, X. Zheng, and G. Tseng. Evaluation and comparison of gene clustering methods in microarray analysis. *Bioinformatics*, 22(19):2405, 2006.

[184] The UniProt Consortium. Ongoing and future developments at the Universal Protein Resource. *Nucleic Acids Research*, 39(suppl_1):D214–219, 2010.

[185] M. Theus. Interactive data visualization using mondrian. *Journal of Statistical Software*, 7(11):1–9, 2003.

[186] O. Thimm, O. Bläsing, Y. Gibon, A. Nagel, S. Meyer, P. Krüger, J. Selbig, L. Müller, S. Rhee, and M. Stitt. Mapman: a user-driven tool to display genomics data sets onto diagrams of metabolic pathways and other biological processes. *Plant Journal*, 37(6):914–939, 2004.

[187] T. Tokimatsu, N. Sakurai, et al. KaPPA-View. A web-based analysis tool for integration of transcript and metabolite data on plant metabolic pathway maps. *Plant Physiology*, 138(3):1289, 2005.

[188] E. R. Tufte. *The Visual Display of Quantitative Information.* Graphics Press, Cheshire, CT, 1983.

[189] E. R. Tufte. *Visual and Statistical Thinking: Displays of Evidence for Decision Making.* Graphics Press, Cheshire, CT, 1997.

[190] E. R. Tufte. *Beautiful Evidence*, chapter Sparklines: Intense, Simple, Word-sized Graphs, pages 47–63. Graphics Press, Cheshire, CT, 2006.

[191] J. Tukey. The future of data analysis. *The Annals of Mathematical Statistics*, 33(1):1–67, 1962.

[192] J. W. Tukey. *Exploratory Data Analysis.* Addison Wesley, 1977.

[193] V. Tusher, R. Tibshirani, and G. Chu. Significance analysis of microarrays applied to the ionizing radiation response. *PNAS*, 98:5116–5121, 2001.

[194] UniProt.org. UniProt Website. http://www.uniprot.org/, 04 2011.

[195] R. G. E. van Eijsden et al. Chip-based mtDNA mutation screening enables fast and reliable diagnosos of OXPHOS patients. *Genetics in Medicine*, 8(10):620–627, 2006.

[196] M. Van Iersel, T. Kelder, et al. Presenting and exploring biological pathways with PathVisio. *BMC Bioinformatics*, 9(1):399, 2008.

[197] V. E. Velculescu, L. Zhang, B. Vogelstein, and K. W. Kinzler. Serial analysis of gene expression. *Science*, 270(5235):484–7, Oct. 1995.

[198] F. Viégas and M. Wattenberg. Tag clouds and the case for vernacular visualization. *Interactions*, 15(4):49–52, 2008.

[199] Z. Wang et al. Identifying Influenza Viruses with Resequencing Microarrays. *Emerging Infectios Diseases*, 12:638–646, 2006.

[200] Z. Wang, M. Gerstein, and M. Snyder. RNA-Seq: a revolutionary tool for transcriptomics. *Nature Reviews Genetics*, 10(1):57–63, 2009.

[201] M. Wattenberg, F. Viégas, and K. Hollenbach. Visualizing activity on wikipedia with chromograms. In *Human-Computer Interaction – INTERACT 2007*, pages 272–287. Springer, 2007.

[202] D. Watts and S. Strogatz. Collective dynamics of 'small-world'networks. *Nature*, 393(6684):440–442, 1998.

[203] K. Weber. Verbesserte Graphische Oberfläche und Algorithmen für Affymetrix Resequenzierungs-Microarrays. Diplomarbeit Bioinformatik, University of Tübingen, 2005.

[204] E. Wegman. Hyperdimensional data analysis using parallel coordinates. *Journal of the American Statistical Association*, 85(411):664–675, 1990.

[205] L. Wilkinson. *The Grammar of Graphics.* Wiley, New York, NY, USA, 2005.

[206] L. Wilkinson and M. Friendly. The history of the cluster heat map. *The American Statistician*, 63(2):179–184, 2009.

[207] D. Wishart, C. Knox, A. Guo, R. Eisner, N. Young, B. Gautam, D. Hau, N. Psychogios, E. Dong, S. Bouatra, et al. HMDB: a knowledgebase for the human metabolome. *Nucleic Acids Research*, 37:D603, 2009.

[208] E. Wit and J. McClure. *Statistics for Microarrays.* Wiley, New York, NY, USA, 2004.

[209] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques, Second Edition (Morgan Kaufmann Series in Data Management Systems).* Morgan Kaufmann, San Francisco, CA, USA, 2005.

[210] K.-K. Yan, G. Fang, N. Bhardwaj, R. P. Alexander, and M. Gerstein. Comparing genomes to computer operating systems in terms of the topology and evolution of their regulatory control networks. *PNAS*, 107(20):9186–9191, 2010.

[211] K. Yee, D. Fisher, R. Dhamija, and M. Hearst. Animated exploration of dynamic graphs with radial layout. In *Infovis*, pages 43–51. IEEE, New York, NY, USA, 2001.

[212] Y. Zhan and D. Kulp. Model-P: a basecalling method for resequencing microarrays of diploid samples. *Bioinformatics*, 21(Suppl. 2):ii182–ii189, 2005.

[213] Q. Zhu, J. Miecznikowski, and M. Halfon. Preferred analysis methods for Affymetrix GeneChips. II. An expanded, balanced, wholly-defined spike-in dataset. *BMC Bioinformatics*, 11(1):285, 2010.

[214] C. Zipplies. Skalierbare Methoden zur interaktiven Visualisierung genomweiter Genexpressionsdaten. Diplomarbeit Bioinformatik, University of Tübingen, 2009.

Lebenslauf

Name	Stephan Hermann Georg Symons
Geburtsdatum und -ort	21. 03. 1980 in Duisburg
1986-1990	Grundschule Krefelder Straße, Duisburg
1990-1996	Realschule Rheinhausen, Duisburg
1996-1999	Krupp-Gymnasium, Duisburg
06/1999	Abitur (Note 2.1) Leistungskurse Biologie und Deutsch
07/1999 - 06/2000	Zivildienst: von-Bodelschwingh-Haus, Duisburg
10/2000 - 10/2006	Studium der Bioinformatik, Eberhard-Karls-Universität Tübingen
11/2005 - 04/2006	Diplomarbeit, Titel “*Machine Learning Algorithms for Microarray Data*”, Betreuer: Dr. Nieselt
10/2006	Diplom Bioinformatik (Note: sehr gut)
seit 10/2006	Promotion an der Mathematisch-Naturwissenschaftlichen Fakultät der Eberhard-Karls-Universität Tübingen, Fachbereich Informatik, Arbeitsgruppe Integrative Transkriptomik