Analysis of integrated transcriptomics and metabolomics data — a systems biology approach

Dissertation

zur Erlangung des Grades
Doktor der Naturwissenschaften (Dr. rer. nat.)
der Mathematisch-Naturwissenschaftlichen Fakultät
der Universität Potsdam
von

Carsten O. Daub

Potsdam
2004

Dekan:	Prof. Dr. Robert Seckler
Gutachter:	Prof. Dr. Hanspeter Herzel
	Prof. Dr. Martin Vingron
	Dr. Joachim Selbig

Abstract

Recent high-throughput technologies enable the acquisition of a variety of complementary data and imply regulatory networks on the *systems biology* level. A common approach to the reconstruction of such networks is the cluster analysis which is based on a similarity measure.
We use the information theoretic concept of the mutual information, that has been originally defined for discrete data, as a measure of similarity and propose an extension to a commonly applied algorithm for its calculation from continuous biological data. We compare our approach to previously existing algorithms. We develop a performance optimised software package for the application of the mutual information to large-scale datasets. Furthermore, we design and implement a web-based service for the analysis of integrated data measured with different technologies. Application to biological data reveals biologically relevant groupings and reconstructed signalling networks show agreements with physiological findings.

Kurzfassung

Moderne Hochdurchsatzmethoden erlauben die Messung einer Vielzahl von komplementären Daten und implizieren die Existenz von regulativen Netzwerken auf einem systembiologischen Niveau. Ein üblicher Ansatz zur Rekonstruktion solcher Netzwerke stellt die Clusteranalyse dar, die auf einem Ähnlichkeitsmaß beruht.
Wir verwenden das informationstheoretische Konzept der wechselseitigen Information, das ursprünglich für diskrete Daten definiert ist, als Ähnlichkeitsmaß und schlagen eine Erweiterung eines für gewöhnlich für die Anwendung auf kontinuierliche biologische Daten verwendeten Algorithmus vor. Wir vergleichen unseren Ansatz mit bereits existierenden Algorithmen. Wir entwickeln ein geschwindigkeitsoptimiertes Computerprogramm für die Anwendung der wechselseitigen Information auf große Datensätze. Weiterhin konstruieren und implementieren wir einen web-basierten Dienst für die Analyse von integrierten Daten, die durch unterschiedliche Meßmethoden gemessen wurden. Die Anwendung auf biologische Daten zeigt biologisch relevante Gruppierungen, und rekonstruierte Signalnetzwerke zeigen Übereinstimmungen mit physiologischen Erkenntnissen.

Acknowledgements

The present work has been carried out in the Bioinformatics group at the Max Planck Institute of Molecular Plant Physiology in Golm.

First and foremost, I would like to thank Sebastian Kloska, who supervised my work from November 2000 to August 2001, and Joachim Selbig, who supervised my work from February 2002 on. They introduced me into the field of bioinformatics and their support build a bases for this work. I am grateful to Sebastian Kloska for continuing his support even after he left the institute and to Joachim Selbig for taking over the supervision. I also would like to thank Bernd Müller-Röber for his advices.

The stimulating discussions with the members of the bioinformatics group, present and past, Jan Hannemann, Stefanie Hartmann, Peter Humburg Jan Hummel, Peter Krüger, Matthias Scholz, Danny Tomuschat, and Janko Weise, gave a friendly and productive working atmosphere.

I also would like to thank all colleagues of the Max Planck Institute of Molecular Plant Physiology and the collaborators from the University of Potsdam. Especially I would like to thank Ralf Steuer for fruitful discussions and for providing me with expert knowledge about mutual information, and Joachim Kopka and Victoria Nikiforova, for being essential partners in learning about the biological background.

I am especially grateful to my parents, Ingrid and Horst Daub, who supported me all my way, and Hanna for her encouragement.

Contents

Chapter 1

Introduction

1.1 Systems biology

Cells, tissues, organs and organisms are examples of biological systems. Traditionally, the understanding of such systems was approached by investigating progressively smaller details of the systems to gain an understanding of the larger concepts. Recently, the understanding of a biological system as a whole comes to the fore with the intention to describe the structure and dynamics of phenomena that are not resolvable into local events. Systems biology aims at answering a set of key questions dealing with

- basic structures and properties of biological networks,
- behaviour of biological systems over time under various conditions,
- robustness and stability of biological networks, and
- modification of biological systems to achieve desired properties.

Even though this trend is relatively new within biology, early attempts originated in other fields of science have been made [1]. In recent years, new attempts have made

use of breakthroughs in measurement techniques, such as large-scale gene expression measurements, metabolite and protein profiling. These attempts can be divided into two groups. The first attempt integrates data from different levels and sources [2] whereas the other attempt shifts the focus from the elementary components of biological processes to systems of such components [3]. Both approaches demand the cooperation of scientists from different disciplines, such as biology, computer science, systems theory, physics, chemistry, and interdisciplinary areas of applied science.

1.2 Microarray technology

Living cells interact with their environment and respond to external signals. They adapt their metabolic processes in a highly regulated fashion. The nucleotide sequence encoding a gene under regulation is thereby transcribed from the DNA, which is located in the cell nucleus, to messenger RNA (mRNA). The transcript leaves the nucleus into the cytoplasm where the mRNA is used by the ribosomes as template for the production of proteins. Each mRNA molecule is a template for a specific protein. The amount of produced proteins is thereby mainly regulated by the amount of the corresponding mRNA.

The aim of microarray technology is the measurement of the levels of thousands of mRNA molecules at once thus providing insight into the transcriptional state of a cell (*transcriptome*) [4, 5, 6]. For a specific cell status, mRNA levels corresponding to all transcribed genes are measured in parallel probes. To this end, mRNA prepared from cell samples is reversively transcribed to cDNA by enzymes with simultaneous incorporation of marker molecules. The marker molecules later allow for a quantitative analysis of the cDNA molecules. Different kind of marker molecules are commonly used, based on radioactive or fluorescent labeling. The labeled cDNA molecules are then applied to the microarray.

The microarray consists of a supporting material, usually chemically treated glass slides or nylon membranes. Spots of DNA fragments are immobilised on these slides

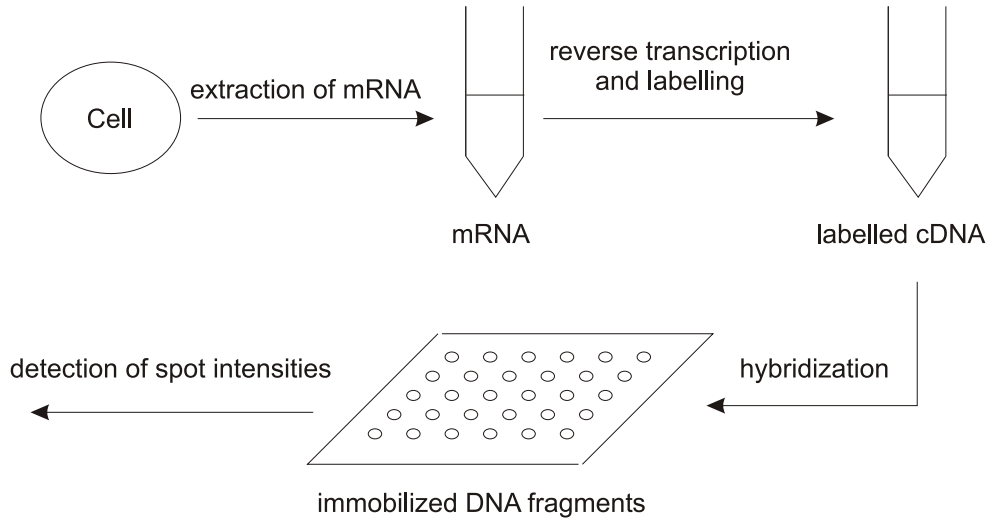

Figure 1.1: **Microarray technology.** Samples from cells grown under specific experimental conditions are prepared and mRNA is extracted. Then mRNA is reversely transcribed into more stable cDNA with simultaneous incorporation of labelling molecules and *hybridised* to the microarray. Spot intensities are detected and post-processed.

or membranes, which then also are called *arrays*. The DNA fragments within each spot are identical and represent characteristic target sequences for the labeled cDNA molecules of the probe. These fragments correspond to genes of interest. When the probe molecules are applied to the microarray, the single stranded cDNA molecules bind only to the spots consisting of DNA fragments of complementary sequences resulting in double stranded DNA. This process is often referred to as *hybridisation*. The level of labeled cDNA on a particular spot can be detected and corresponds to the amount of original mRNA complementary to the DNA fragments in the spot. The schematic work-flow of a microarray analysis is shown in Figure 1.1.

Microarray technology has been used for a variety of applications ranging from studies of knock-out mutant organisms, analysis on time-series from organisms reacting to different environmental conditions [7], to therapeutical [8, 9] and diagnostic [10] purposes.

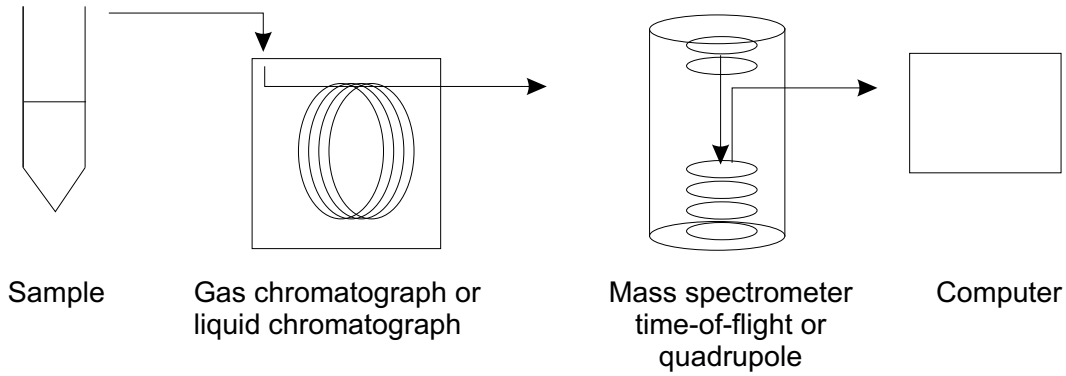

Figure 1.2: **Metabolite profiling technology.** Tissues are homogenised and extracted to gain proper samples. Chromatography splits metabolites up according to their chemical properties, e.g. hydrophobicity or polarity. Mass spectrometry is then applied for identification and quantification of the samples.

1.3 Metabolite profiling technology

Similar to the quantification of predefined targets of mRNA with microarrays, the classical *metabolite profiling* aims at the quantification of a number of predefined target metabolites. A target can be a set of metabolites shared among different pathways or all metabolites of a specific pathway. In extension to the target oriented metabolite profiling, the global detection of all metabolites present in a sample – the *metabolome* – was approached. Metabolomic analyses thereby aim to provide a comprehensive insight into the metabolic state of a system — the full suite of metabolites expressed in a cell.

Approaches for tissue sampling, probe homogenisation, extraction, storage, and sample preparation have to be made adequately to maintain an *unbiased* approach for the estimated large number of metabolites. Many different protocols are available, but no comprehensive comparison of extraction techniques has been published. Common criteria for all these methods are a high reproducibility, robustness, and recovery.

Numerous techniques exist for the detection of metabolites. All commonly used ones characterise the samples according to more than one physical parameter: in a

first step metabolites are separated by chromatography according to their chemical properties (for example hydrophobicity or polarity). Depending on the substance class of metabolites under interest, gas chromatography (GC) or liquid chromatography (LC), also with their derivations like reverse-phase liquid chromatography (RPLC) and normal-phase liquid chromatography (NPLC), are among the most frequently used techniques. After the chromatographical separation of the samples, mass spectrometry (MS) is applied for metabolite detection. MS has shown its suitability for metabolite detection in complex matrices [11, 12]. Figure 1.2 shows a schematic overview of the experimental setup. The parallel use of GC/MS and other combinations like RPLC/MS comprehensively covers the accurate identification and quantification of a large fraction of all available metabolites.

1.4 Protein measurement technology

A large variety of different protein identification and quantification strategies has been described and employed by the proteomic community. Conventional proteomics relies heavily on 2-D gel electrophoresis followed by the identification of proteins by direct excision and processing from the gel (2-D PAGE). However, many proteins such as hydrophobic membrane-associated proteins or proteins of low abundance fail to be detected. For this, techniques for the labeling of amino acids, that are incorporated into the proteins, or the chemical labeling of the proteins are applied [13]. A prominent examples for the chemical labeling is the **i**sotope-**c**oded **a**ffinity **t**ag (ICAT) where the thiol group of cystein is labeled at the alkylation step of sample preparation. Labeled proteins are either enzymatically or chemically digested or separated on a gel and detected based on MS.

1.5 Data integration

For a systems biology approach towards the understanding of the regulation of networks within organisms, data from various experimental sources are integrated. Samples are chemically processed in a way that metabolites, proteins, and mRNA are extracted and abundances of compounds can be measured separately.

In an ideal case, each sample is taken from an organism grown under highly controlled conditions. These conditions might reflect a time series of developmental stages, stress conditions, or genetically modified organisms with altered gene activities like knock-out genes or over expressed genes. A flow chart for the integrated extraction of metabolites, proteins and mRNA from one biological sample is shown in Figure 1.3. mRNA, metabolite, and protein abundance values for one experiment are obtained from the very same sample. For the analysis in a systems biological approach, data from these different experimental sources are integrated into one dataset.

The dataset can be organised as a matrix with the rows containing expression,

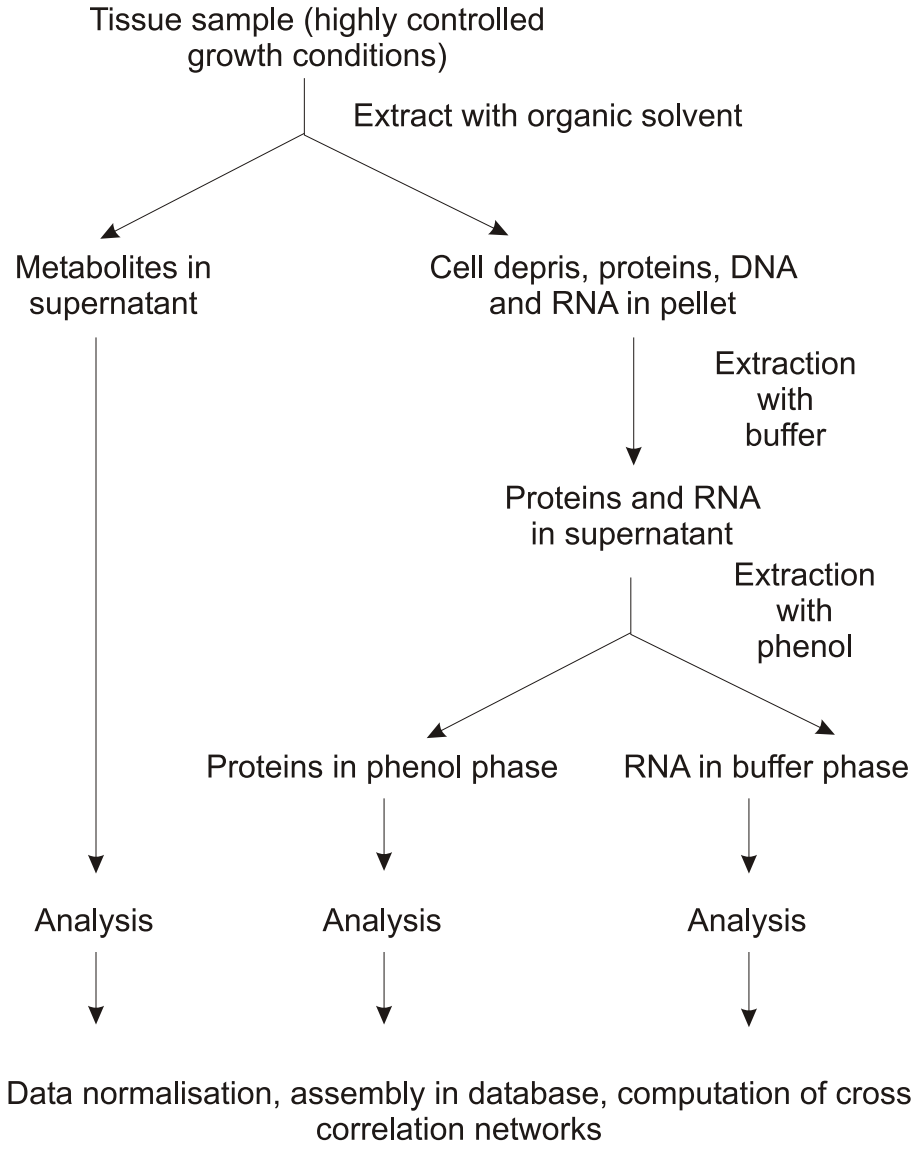

Figure 1.3: Proposed flow chart for the investigation of cross-correlations of metabolites, proteins and mRNA by integrative extraction from one sample (taken from [14]).

	exp. 1	exp. 2	exp. 3	...	exp. n
gene 1	$g_{1,1}$	$g_{1,2}$	$g_{1,3}$	...	$g_{1,n}$
gene 2	$g_{2,1}$	$g_{2,2}$	$g_{2,3}$	...	$g_{2,n}$
gene 3	$g_{3,1}$	$g_{3,2}$	$g_{3,3}$	...	$g_{3,n}$
...	...	...	...	...	...
gene k	$g_{k,1}$	$g_{k,2}$	$g_{k,3}$	...	$g_{k,n}$
metabolite 1	$m_{1,1}$	$m_{1,2}$	$m_{1,3}$	...	$m_{1,n}$
metabolite 2	$m_{2,1}$	$m_{2,2}$	$m_{2,3}$	...	$m_{2,n}$
metabolite 3	$m_{3,1}$	$m_{3,2}$	$m_{3,3}$	...	$m_{3,n}$
...	...	...	...	...	...
metabolite l	$m_{l,1}$	$m_{l,2}$	$m_{l,3}$	...	$m_{l,n}$
protein 1	$p_{1,1}$	$p_{1,2}$	$p_{1,3}$	...	$p_{1,n}$
protein 2	$p_{2,1}$	$p_{2,2}$	$p_{2,3}$	...	$p_{2,n}$
protein 3	$p_{3,1}$	$p_{3,2}$	$p_{3,3}$	...	$p_{3,n}$
...	...	...	...	...	...
protein m	$p_{m,1}$	$p_{m,2}$	$p_{m,3}$	...	$p_{m,n}$

Table 1.1: **Dataset format.** Samples from the very same tissue are analysed with different experimental techniques. Resulting experimental data is integrated into one dataset. All data within one column refer to the same experimental condition (mutant or time point in a time series) and are measured from the very same tissue. Each row corresponds to a gene, metabolite, or protein.

metabolite, and protein abundances and the columns representing the experimental conditions. An example matrix comprising such a structure is shown in Table 1.1.

1.6 Current methods of data analysis

Within the analysis of gene expression data and metabolite data, the term *data analysis* summarises various steps of data processing. The analysis starts from images containing the raw information about gene abundances or chromatograms with peaks representing the metabolite concentrations. The application of high-

level multivariate statistical technique to this raw data assumes a prior low-level normalisation.

1.6.1 Data normalisation

To enable the comparison of expression data obtained from different array experiments or of metabolite concentrations measured from different samples, a correction for technical and biological variation needs to be applied. The detailed procedures vary depending on the array formats and experimental designs. Basically, the normalisation procedures for gene expression data attempt to correct for the

- number of cells in the sample
- efficiency of total RNA isolation
- efficiency of mRNA isolation and labelling
- efficiency of hybridisation
- sensitivity of signal measurement.

For metabolite measurement the corrections attempt for the

- smoothing of all peaks of a chromatogram
- determination and subtraction of the background noise
- division of each peak by the area of a standard substance peak which was added to the sample in a defined amount
- normalisation to the fresh weight (FW) or to the total ion chromatogram (TIC)

These data preprocessing steps are often referred to as *low-level* data analysis.

Normalisation can be applied among attributes (corresponding to the rows in Table 1.1) or among the experiments (corresponding to the columns in Table 1.1) of a dataset depending on the aims of the normalisation. A common strategy for the normalisation of metabolite concentration data, for example, is the assumption that all cells of an organism contain on average a constant concentration of metabolites. This assumption can be realised by the normalisation with the vector norm for each of the experimental conditions in the data matrix. For the $L1$-norm, all data points within an experimental condition are divided by the sum of these data points. Alternatively, normalisations within the attributes can be applied. Such normalisations might correct, for example, for different variances within the attributes. Among commonly applied corrections are the normalisation the the mean, median, or to the variance.

1.6.2 High-level data analysis

The repertoire of high-level data analysis methods applied is manifold. Methods recently reported in literature can be grouped into

- projection
- classification
- clustering

Projection

Projection methods reduce the dimensionality of high dimensional datasets to enable their visualisation in two- or three-dimensional space. Their application helps to circumvent the constrains arising from high feature spaces compared to relatively few experiments which is sometimes referred to as *the curse of dimensionality* and is commonly arising in gene expression and metabolite analysis [15]. Projection

methods aim at representing distances between objects in the original data space as closely as possible in a reduced data space [16].

Classification

Classification aims at finding characteristic discriminative features (so called *classifiers*) to assign objects to groups. The groups have to be known in advance for the training of the classifier. These classifiers are then used to assign new objects to existing groups. In the field of cancer research, the objects may consist of samples of tumor and non-tumor cells [17]. Classifiers achieved from samples of known cancer status are used to assign new samples to the 'tumor' or 'non-tumor' group. For *Arabidopsis thaliana* objects consisting of different ecotypes were distinguished using metabolic properties [18].

Clustering

Clustering aims at the grouping of objects which show similar patterns of features. For the analysis of microarray data or metabolite data, these objects might consist of groups of genes/metabolites or groups of experiments. The similarity between objects are determined differently depending on the algorithm used. For K-means clustering [19, 20], the number of clusters has to be selected in advance. The algorithm searches in an iterative manner for the optimal location of cluster centers in the Euclidean space spanned by the dimensions of the dataset under consideration. Self-organizing maps (SOM) search for optimal positions of cluster centers for a map of nodes in a high dimensional space spanned by the dimensions of the dataset. The initial geometry of the nodes has to be chosen in advance. For biclustering [21, 22], subsets of objects exhibiting consistent patterns over both genes/metabolites and experiments of a dataset are detected.

Hierarchical clustering bases on the prior calculation of a distance matrix containing the pair-wise comparisons of all objects of a dataset. The distance between these objects is a measure of the correlation between them and can be defined in various

ways. Examples of correlation measures range from linear correlation and Bayesian similarity metrics [23] to information theoretic concepts like mutual information [24, 25]. The result of the application of a clustering algorithm to a distance matrix is commonly displayed as dendrogram where the lengths of the branches correspond to the distances between the objects. Figure 1.4 depicts an example dendrogram together with the courses of expression values for the two main clusters of genes.

Compared to the manifold of methods reported for the analysis of microarray data, the analysis of metabolomic studies is so far restricted to relatively few approaches. Metabolic correlation networks have been calculated from the pair-wise comparison of metabolite concentrations [27, 28, 29]. Based on the theory of stochastic systems, the correlations of a metabolomic network have been interpreted as *fingerprint* of the underlying biophysical system and have been compared to known biochemical pathways [30]. Different genotypes have been distinguished using machine learning techniques [18] and principal component analysis (PCA) [12].

The frequently reported methods for the analysis of gene expression and metabolite data are summarised in Table 1.2.

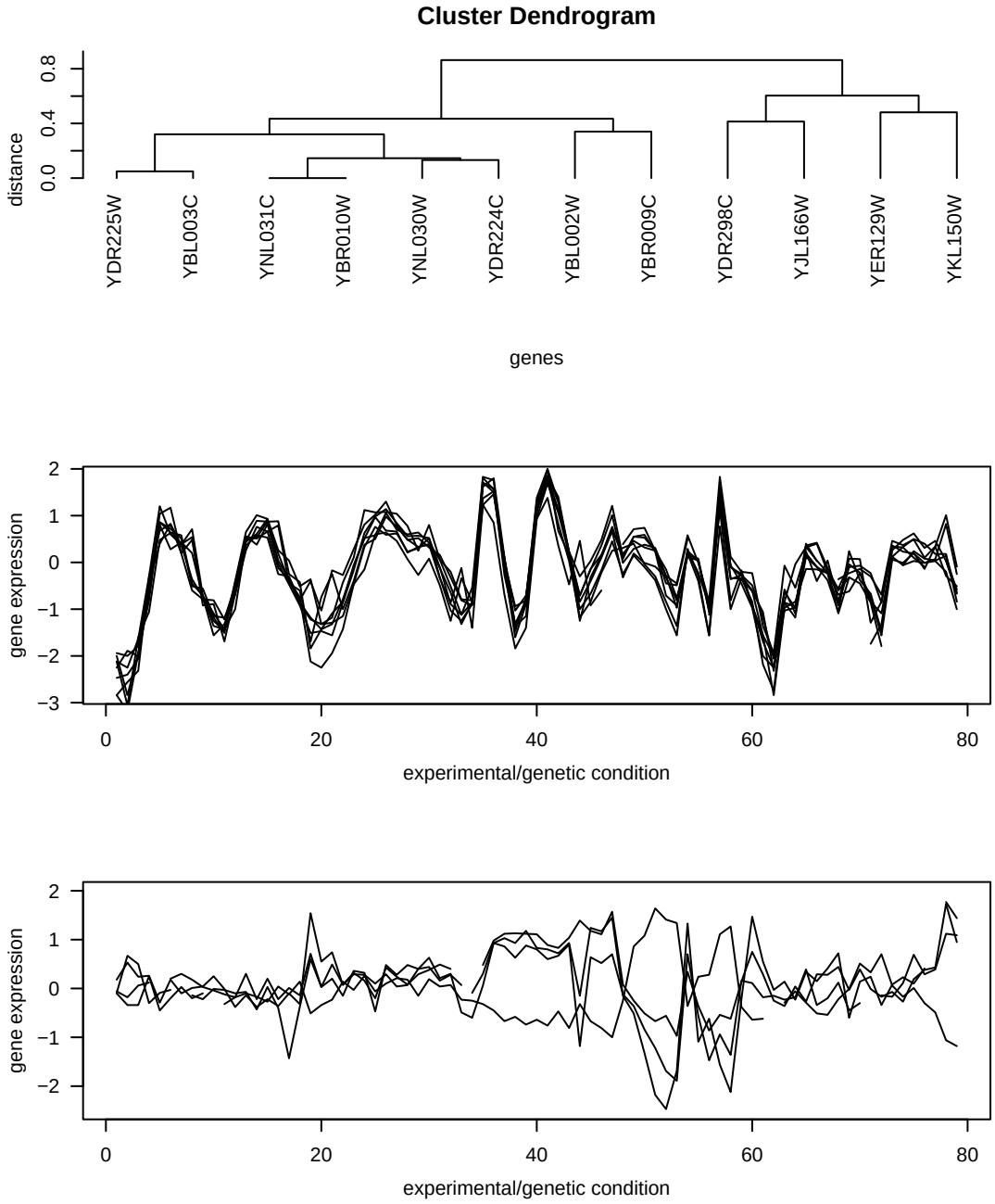

Figure 1.4: **Clustering.** The result of a hierarchical clustering displays all genes of a dataset (example data is a cutout from [26]) in a dendrogram (top). The expression values of genes within the two main clusters (middle, bottom) show a fairly similar course.

Method	Microarray	Metabolomic
Projection		
Principal component analysis (PCA)	[16, 31]	[12]
Singular value decomposition (SVD)	[32, 33]	
Canonical correlation analysis (CCA)	[34]	
Independent component analysis (ICA)	[35]	[36]
Classification		
Support vector machines (SVM)	[37, 38]	
Decision trees	[39]	
Artificial neural networks	[40]	[18]
k-nearest neighbours	[41]	
Clustering		
K-means clustering	[19, 20]	
Self organizing maps (SOM)	[42]	
Expectation maximisation (EM) algorithm	[43, 44, 45]	
Autoclass algorithm	[46]	
FITCH	[47]	
Hierarchical clustering	[26]	[27, 28, 29]
Clustering based on random graph theory	[48]	
Biclustering	[21, 22]	
Gene Shaving	[49]	

Table 1.2: **Data analysis.** An overview of frequently reported methods used for the analysis of data from gene expression and metabolite measurements.

1.7 Contributions

1.7.1 Mutual information

For the elucidation of functional relationships inherent in data, the clustering of data matrices of co-expressed genes [26] or metabolites [12] is a commonly used approach. For the obtained results, the choice of the similarity measure is as crucial as the choice of the clustering method itself [50]. Often, linear similarity measures such as the Euclidean distance [51] or the Pearson correlation coefficient [26] are applied. As extension to these linear methods, information theoretic concepts, such as the mutual information, have been used [24, 25, 52, 53, 54] to enable the detection of non-linear correlations within data. The concept of mutual information was initially developed for discrete data. For its application to continuous data from gene expression or metabolite measurements, the experimental data need to be partitioned into discrete intervals, or bins. In a widely used approach [24], each data point is assigned to one, and only one, bin. Even small fluctuations due to biological or measurement noise can thereby strongly affect the resulting mutual information [55].

We have developed a generalisation to the classical binning in which we aim to overcome some of the drawbacks associated with the simple approach. Within this algorithm, data points are allowed to be assigned to several bins simultaneously. We implemented the algorithm into a software package.

1.7.2 MetaGeneAlyse

Furthermore, we have developed a software package for the analysis of gene expression data, metabolite data, and datasets merged from both types of data: *MetaGeneAlyse* is a compilation of scripts and programs for data normalisation, data analysis, and data visualisation, accessible via a web-frontend. Among several other algorithms for distance matrix calculation, it contains a routine for the calculation

of mutual information based on the generalised binning method.

1.7.3 Application on data

We applied the mutual information as a measure of similarity to evaluate the dependencies between genes and metabolites. In a first application, we addressed the question whether recent large-scale gene expression datasets are well described by commonly used linear similarity measures or if we are able to detect non-linear gene-gene association.

Aiming at the discovery of novel biological insights into large-scale gene expression dataset, we carried out a cluster analysis using mutual information as similarity measure and evaluated the results on the basis of biological expert knowledge.

In an integrated systems biology approach, for samples of gene expression and metabolite measurements, we reconstruct a network of informational fluxes arising from stress response.

Chapter 2

Data analysis

2.1 Mutual information

2.1.1 Introduction to mutual information

The detection of relationships between objects plays a central role for the evaluation of complex networks. To unravel functional connections inherent in such networks, the clustering of objects is a commonly used procedure. The choice of an appropriate distance measure underlying the clustering procedure is thereby highly important. In extension to other distance measures, mutual information provides a general measure for statistical independence between objects. Since this work later on exemplifies the concepts derived in this chapter in terms of analysing data from gene expression and metabolite measurements, one might think of objects as genes or metabolites which are also commonly referred to as the variables of a dataset.

Within the analysis of biological data, the concept of mutual information has been used in various contexts ranging from gene expression [24, 52, 53, 54] and DNA sequence analysis [56, 57], to reverse engineering [58]. Due to its general applicability, mutual information is also widely utilised in diverse disciplines, such as physics [59],

image recognition [60], speech recognition [61], and various others. Extending the abilities of commonly used linear measures, such as the Euclidean distance or the Pearson correlation coefficient, mutual information is able to detect any type of functional relationship as illustrated in Figure 2.1.

2.1.2 Definition of the mutual information

2.1.3 Shannon entropy

The concept of the mutual information was initially developed for discrete random data. Examples for discrete biological data are DNA sequence information or the number of plants showing a specific phenotype. For a random variable, A, with a finite set of M_A possible states, $a_1, \ \dots \ , a_{M_A}$, and each state with its corresponding probability $p(a_i)$, the Shannon entropy $H(A)$ is defined as [63]

$$H(A) = -\sum_{i=1}^{M_A} p(a_i) \log p(a_i) \tag{2.1}$$

In a more descriptive definition, the Shannon entropy can be seen as a measure for how evenly the the states of the system A are distributed. In the case where the outcome of a measurement on A is completely determined to be a_j, thus if $p(a_j) = 1$ and $p(a_i) = 0$ for all $i \neq j$, the entropy becomes zero. Whereas in the other extreme case, where all probabilities are equal, the entropy becomes maximal.

The joint entropy $H(A, B)$ of two random variables A and B is defined analogously

$$H(A, B) = -\sum_{i=1}^{M_A} \sum_{j=1}^{M_B} p(a_i, b_j) \log p(a_i, b_j) \tag{2.2}$$

where $p(a_i, b_j)$ denotes the joint probability that A is in state a_i and B is in state b_j. Alternatively, the joint probability can be expressed in terms of the conditional entropy $H(A|B)$

$$H(A, B) = H(A|B) + H(B) \tag{2.3}$$

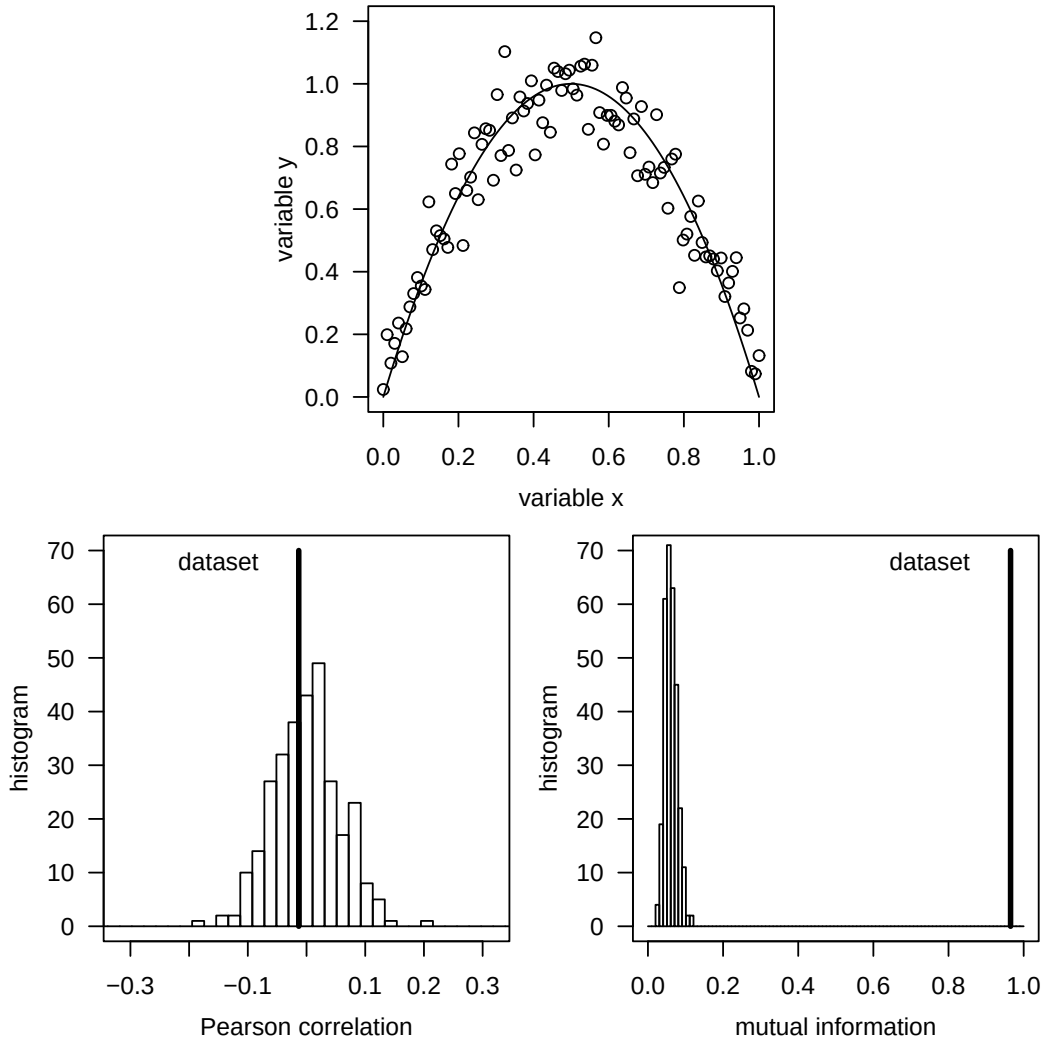

Figure 2.1: **Ability of the mutual information.** Two variables X and Y (100 data points) show a hypothetical dependency according to the function $f(x) = 4x(1-x)$ (top). The Pearson correlation coefficient is not able to detect a significant correlation as shown in the histogram plot of the dataset compared to 300 realisations of shuffled data (left). Mutual information clearly shows that the two datasets are not statistically independent (right). This figure is taken from [62].

with $H(A|B)$ being defined as

$$H(A|B) = -\sum_{i=1}^{M_A}\sum_{j=1}^{M_B} p(a_i, b_j)\log p(a_i|b_j) \tag{2.4}$$

When now considering the general relation valid for arbitrary systems A and B

$$H(A|B) \leq H(A) \tag{2.5}$$

this leads to a relation for the joint entropy

$$H(A, B) \leq H(A) + H(B) \tag{2.6}$$

The equality is fulfilled only in the case of statistical independence of A and B. The definition of the mutual information $MI(A, B)$ is given by [63, 64]

$$MI(A, B) = H(A) + H(B) - H(A, B) \geq 0 \tag{2.7}$$

Only for statistical independence of A and B the mutual information becomes zero. It increases the less statistically independent A and B are.

2.1.4 Kullback entropy

In a different approach, the mutual information was given by Kullback in the form of a conditional entropy

$$K(p|p^0) := \sum p_i \log \frac{p_i}{p_i^0} \geq 0 \tag{2.8}$$

The Kullback entropy can be regarded as a measure of distance between the two probability distributions p and p^0 and interpreted as information gain when replacing an initial probability p^0 by a final one p. The Kullback entropy, therefore, established as distance measure between the two distributions. Assuming the hypothesis of statistical independence for p^0 for two systems A and B, $p^0(a_i, b_j)$ can be written as

$$p^0(a_i, b_j) = p(a_i)\, p(b_j) \tag{2.9}$$

and the Kullback entropy can be rewritten as

$$K(p|p^0) = \sum_{i=1}^{M_A}\sum_{j=1}^{M_B} p(a_i, b_j)\log \frac{p(a_i, b_j)}{p(a_i)\, p(b_j)} \tag{2.10}$$

With the particular choice of p and p^0, the Kullback entropy corresponds the the mutual information of Eq. (2.7).

2.1.5 Entropy estimation from continuous data

The estimation of the mutual information according to Eq. (2.2) and Eq. (2.7) assumes the knowledge of the probability distributions of the variables A and B. In the case of discrete data, the estimation of the probabilities $p(a_i)$ is straightforward. Examples for discrete data are DNA sequence information and the number of individuals showing a specific characteristic. In most practical applications, these distributions are not known, but have to be estimated from continuous data.

Simple binning approach

The most straightforward and widely used approach [24, 53] for the estimation of mutual information from continuous data bases on a histogram technique in which data is binned into M discrete intervals a_i, $i = 1 \dots M_A$. For experimental data consisting of N measurements of a variable x_u, $u = 1 \dots N$, the number of data points within each of the bins is counted by an indicator function Θ_i

$$\Theta_i(x_u) = \begin{cases} 1 & \text{if} \quad x_u \in a_i \\ 0 & \quad \text{otherwise} \end{cases} \tag{2.11}$$

The probabilities for each interval are then estimated based on the relative frequency of data points in that interval

$$\hat{p}(a_i) = \frac{1}{N} \sum_k \Theta_i(x_u) \tag{2.12}$$

The joint probabilities $\hat{p}(a_i, b_j)$ for two variables are calculated analogously from a two dimensional histogram.

When estimating mutual information from continuous data using the simple binning, as described above, each data point is assigned to one, and only one, bin. Due to even small fluctuations arising from biological or measurement noise, data points near to the border of a bin might be shifted to neighbouring bins. The arbitrarily chosen borders of the bins can thereby strongly affect the resulting mutual information [55], especially for datasets of moderate size.

Size of data

When calculating mutual information according to the simple binning approach from a finite dataset with N data points, the estimated value of the mutual information is always larger than the 'true' one [25, 65, 66, 67, 68]. For illustration, the mutual information is estimated from a dataset of known distribution for which the 'true' mutual information is known. In literature, this effect is referred to as *finite size effect*.

Figure 2.2 (top) shows an example of 'measurements' of the two variable x and y. Both variables are artificially generated independent and equidistributed random numbers and the 'true' mutual information between them is zero. For numerical estimation, data is divided into $M_x = M_x = 10$ bins and the mutual information is calculated according to the simple binning approach using the indicator function of Eq. (2.11). This 'experiment' is repeated for 300 independent realisations of the dataset and the distribution of the 300 outcomes are plotted as histogram. From this it can be clearly observed that the estimated mutual information fluctuates around a value larger than the 'true' value of zero.

Kernel density estimation

Besides the commonly applied partitioning of data into discrete bins, more sophisticated methods are available for estimating the mutual information in a continuous form

$$\hat{M}I(X,Y) = \int_x \int_y \hat{f}(x,y) \log \frac{\hat{f}(x,y)}{\hat{f}(x)\,\hat{f}(y)} \tag{2.13}$$

where X and Y are variables that have been jointly measured under $u = 1 \ldots N$ conditions yielding the samples $(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)$. Within the kernel density estimation (KDE) approach [55], the histogram of the simple binning approach is freed from a particular choice of the origin of the bins and their position. So called *kernel functions* $K(x)$ are placed on each data point and the kernel density estimator

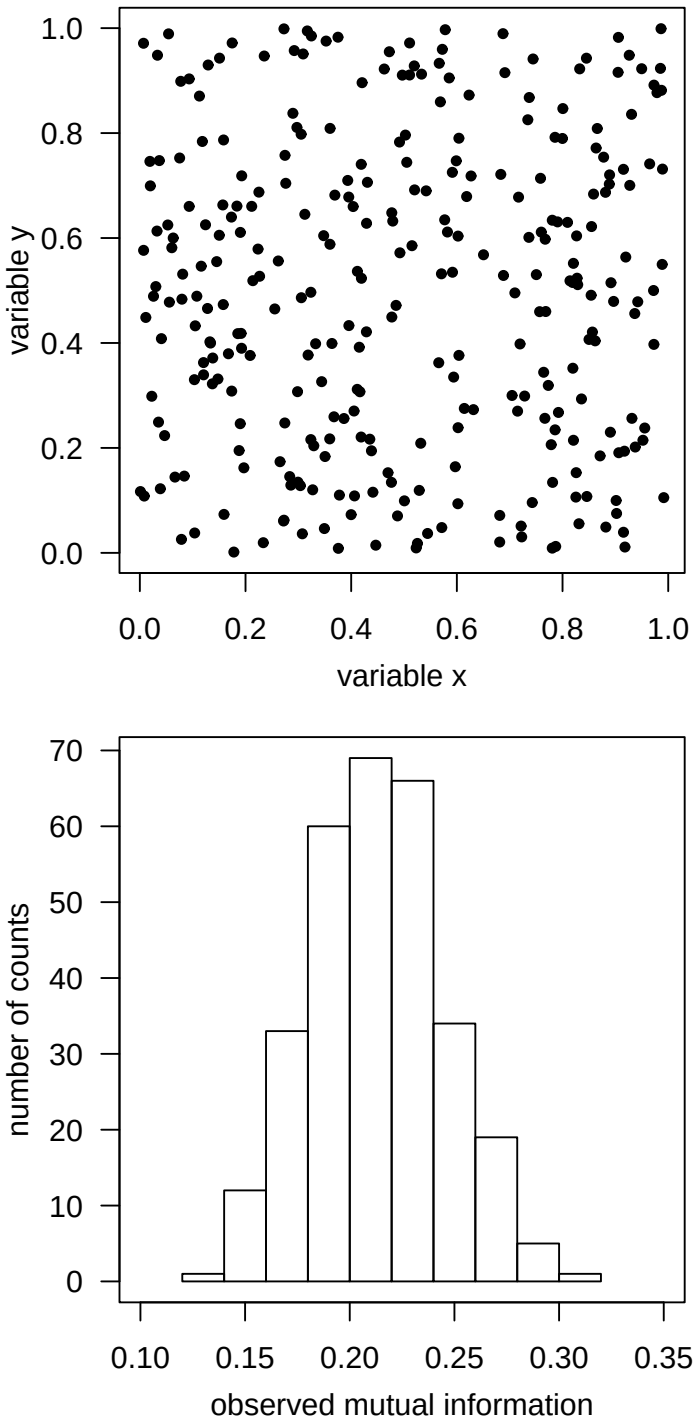

Figure 2.2: **Overestimation of mutual information.** The mutual information calculated from data is systematically overestimated. A dataset of $N = 300$ artificially generated, independent and equidistributed data points for two variables is generated (top). The mutual information obtained from 300 independent realisations using $M_x = M_y = 10$ bins is calculated and plotted as histogram (bottom).

$\hat{f}(x)$ is calculated by

$$\hat{f}(x) = \frac{1}{Nh^d} \sum_{i=1}^{N} K\left(\frac{x - x_i}{h}\right) \tag{2.14}$$

with d being the dimensionality. The parameter h is called *window width* or *smoothing parameter* and estimation procedures for its optimal choice have been proposed [69]

$$h_{optimal} \approx \left(\frac{4}{d+2}\right)^{1/(d+4)} N^{-1/(d+4)} \tag{2.15}$$

One example among several suggested types of kernel functions $K(x)$ [69] is a multivariate Gaussian function

$$K(x) = \frac{1}{(2\pi)^{d/2}} \exp\left(-\frac{1}{2} x^T x\right) \tag{2.16}$$

As illustrated in Figure 2.3, one may understand the kernel functions as little 'bumps' placed at the positions of each observation.

The probability density of Eq. (2.14) can be estimated with standard numerical integration methods and used for the calculation of the mutual information according to Eq. (2.13).

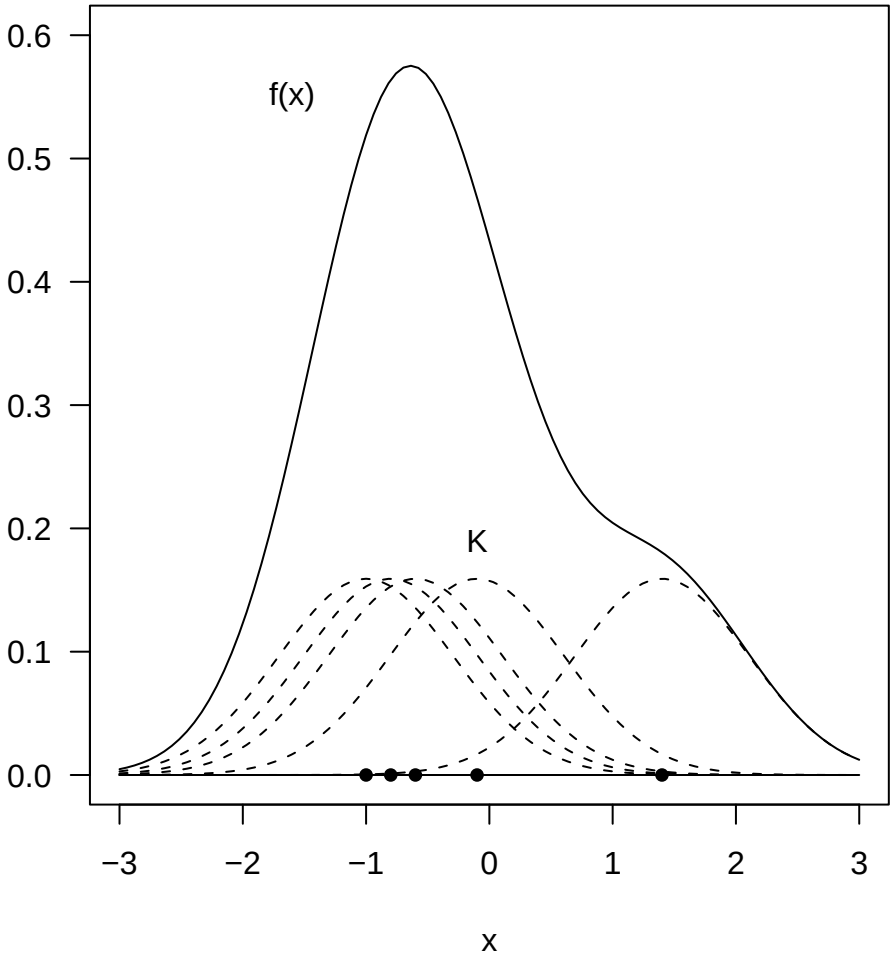

Figure 2.3: **Kernel density estimator.** A Gaussian kernel density estimator $\hat{f}(x)$ can be heuristically understood as placing Gaussian 'bumps' at the position of each observation. The estimator according to Eq. (2.14) is then given by the sum of all bumps.

2.2 Fuzzy mutual information

In a different approach for the estimation of the probability density, we present our extension to the classical binning strategy. Within this approach, data points are allowed to be assigned to several bins simultaneously. For this, the indicator function $\Theta(x)$ according to Eq. (2.11) is generalised to a set of polynomial B-spline functions. The shape of the B-spline functions is determined by the spline order k which thereby also determines the number of bins each data point is assigned to. A spline order $k = 1$ corresponds to the simple binning as described in the previous section: each data point is assigned to exactly one bin. For higher spline orders, $k \geq 2$, each data point is assigned to exactly k bins. The respective weights are given by the values of the B-spline functions at that data point. Figure 2.4 depicts examples for different spline orders.

2.2.1 B-spline functions

The first step in the definition of the B-spline functions is the definition of a knot vector t_i for bin i and one given spline order $k = 1 \ldots M - 1$ [70]

$$t_i := \begin{cases} 0 & \text{if} \quad i < k \\ i - k + 1 & \text{if} \quad k \leq i \leq M - 1 \\ M - 1 - k + 2 & \text{if} \quad i > M - 1 \end{cases} \tag{2.17}$$

where the spline order determines the degree and thereby the shape of the polynomial functions. The domain of the B-spline functions lies in the interval $z \in [0, M-k+1]$. To cover the range of the variable x, the new indicator function based on the B-spline functions needs to be linearly transformed to map the range of x. For an example with $M = 5$ bins and spline order $k = 3$, the domain of the B-spline function lies in the interval $[0, 3]$. To map this interval to the range of a variable $x \in [\min(x), \max(x)]$, the domain of the B-spline functions needs to be linearly transformed to the range of x.

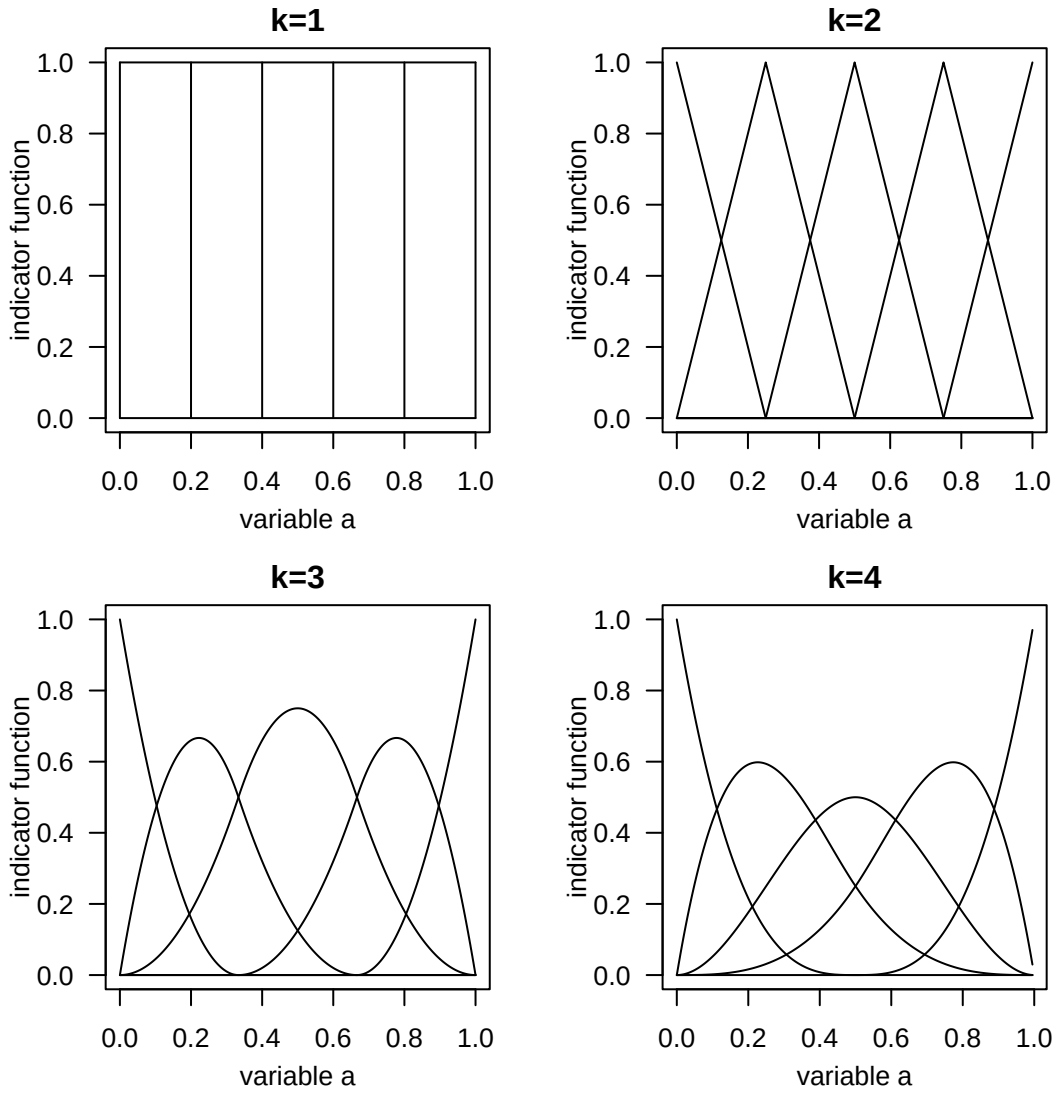

Figure 2.4: **B-spline functions.** For the calculation of mutual information, the continuous experimental data for the variable a needs to be binned. The indicator function of Eq. (2.11) counts the number of data points within each bin (example with $M = 5$ bins, top left). The generalised indicator function of Eq. (2.18) extends the bins to polynomial B-spline functions. The spline order k determines the shape of the B-spline functions. The bins now overlap in a way that each data point is assigned to exactly k bins. The weight of each data point to each of the bins is given by the intersection with the respective graphs of the B-spline functions at the data point. By definition, all weights contributing to one data point sum up to unity (example with $M = 5$ bins and spline order $k = 2 \dots 4$, top right, bottom left, bottom right, respectively).

The recursive definition of the B-spline functions reads [70]

$$\begin{aligned} B_{i,1}(z) := & \begin{cases} 1 & \text{if } t_i \leq z < t_{i+1} \\ 0 & \text{otherwise} \end{cases} \\ B_{i,k}(z) := & B_{i,k-1}(z)\frac{z-t_i}{t_{i+k-1}-t_i} + \\ & B_{i+1,k-1}(z)\frac{t_{i+k}-z}{t_{i+k}-t_{i+1}} \end{aligned} \tag{2.18}$$

An important property of B-spline functions is the implicit standardisation of coefficients: All weights belonging to one data point sum up to unity.

2.2.2 Algorithm

In the following, the calculation of the entropy using the B-spline approach is described in detail. First, the marginal entropies for both variables have to be calculated separately. Then, the joint entropy is calculated and from this the mutual information.

Marginal entropy

In the histogram binning of the simple binning approach, each data point of a variable x is assigned to exactly one bin by the indicator function of Eq. (2.11). Within the B-spline approach, this indicator function is extended. The new indicator function, which is given in Eq. (2.18), assigns each data point to more than one bin. For a spline order of k, each point is assigned to exactly k bins. Thus, a spline order $k = 1$ corresponds to the simple binning and the B-spline approach can be regarded as extension to this simple binning.

The indicator function of the B-spline approach is determined by the spline order k and the number of bins M. Its domain is also determined by these parameters to $[0, M - k + 1]$. To enable the application of this function to all values of x with

the domain $[\min(x), \max(x)]$, the indicator function $B_{i,k}(z)$ has to be modified into $\tilde{B}_{i,k}(x)$ with

$$z = x - \min(x)\frac{M_x - k + 1}{\max(x) - \min(x)} \tag{2.19}$$

The new indicator function covers exactly the domain of x.

Now, the proportion of each data point x_u to each of the bins a_i can be determined by $\tilde{B}_{i,k}(x)$. In a graphical interpretation, as shown in Figure 2.5 for exemplarily chosen data points, each data point intersects with and thereby is assigned to exactly k spline functions. All coefficients belonging to one data point are implicitly normalised as a property the the underlying B-spline functions.

From this point on, only the above determined coefficients are used for the calculation of the probability $p(a_i)$. For each bin a_i, a loop over all N data points averages all coefficients belonging to a_i

$$p(a_i) = \frac{1}{N}\sum_{u=1}^{N}\tilde{B}_{i,k}(x_u) \tag{2.20}$$

The marginal entropy $H(x)$ for variable x is then calculated according to Eq. 2.1.

Joint entropy

For the calculation of the joint entropy, the weighting coefficients $\tilde{B}_{i,k}(x_u)$ and $\tilde{B}_{i,k}(y_u)$ are determined for both variables x and y independently. The joint probabilities $p(a_i, b_j)$ are then calculated for each of the $M_x \times M_y$ 2-dimensional bins

$$p(a_i, b_j) = \frac{1}{N}\sum_{u=1}^{N}\tilde{B}_{i,k}(x_u) \times \tilde{B}_{j,k}(y_u) \tag{2.21}$$

The joint entropy can then be calculated according to Eq. 2.2.

Finally, the mutual information $MI(x, y)$ is calculated from the marginal entropies and the joint entropy according to Eq. 2.7. This procedure is summarised in Figure 2.6.

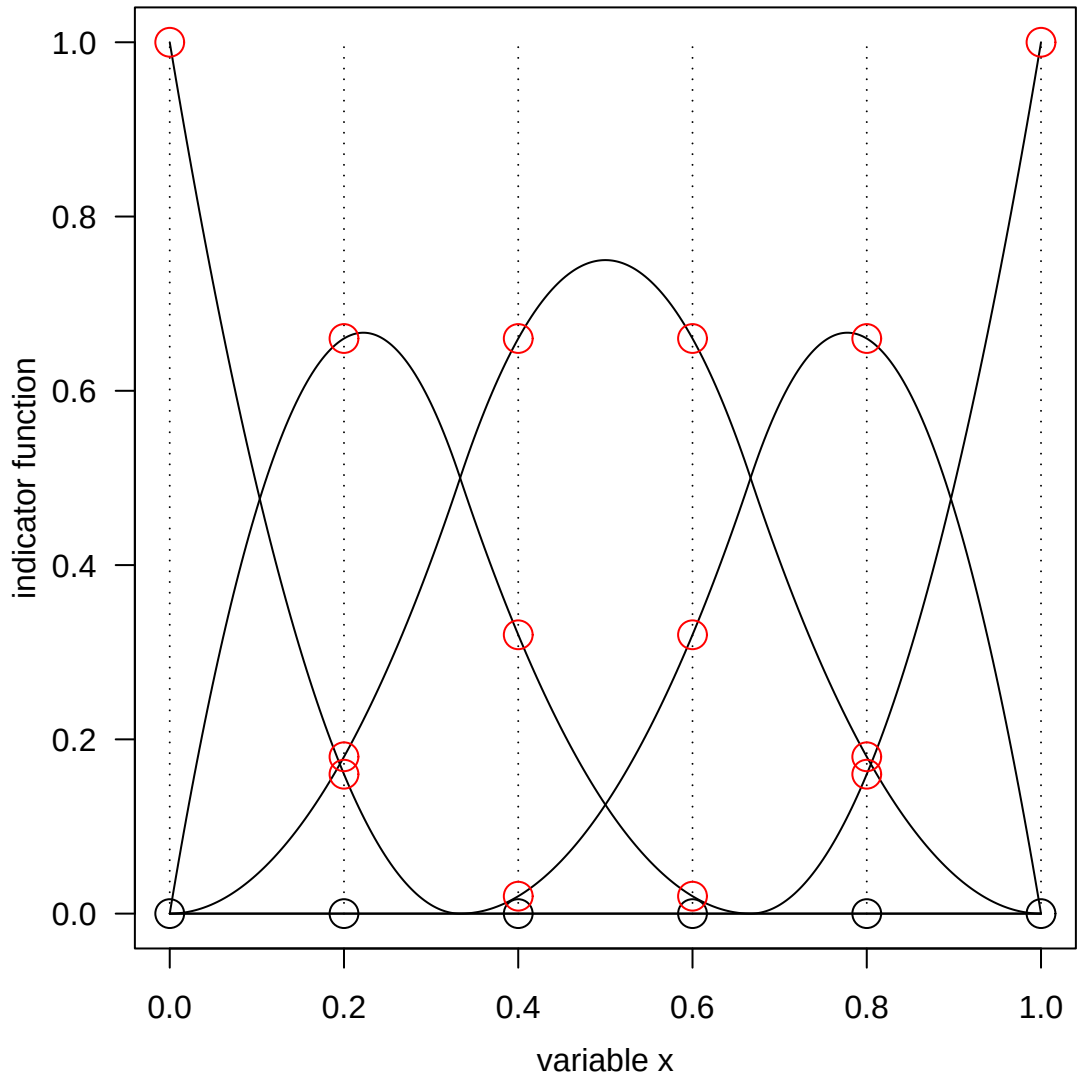

Figure 2.5: **Binning based on B-spline functions.** Extending the simple binning approach, each data point is assigned to more than one bin depending on the spline order k. For $k = 3$, each data point intersects with exactly three spline functions and the proportion to each 'bin' is represented by the intersection. One property of B-spline functions is the implicit normalisation: all proportions for one data point sum up to unity. For the two marginal data points ($x = 0.0$ and $x = 1.0$) just one intersection is shown since the other coefficients overlap with the data points itself.

INPUT

- Variables x and y with values x_u and y_u, $u = 1 \dots N$
- Bins a_i, $i = 1 \dots M_x$ and b_j, $j = 1 \dots M_y$
- Spline order k

OUTPUT

- Mutual information between variable x and y

ALGORITHM

1. Calculation of marginal entropy for variable x
   (a) Determine $\tilde{B}_{i,k}(x) = B_{i,k}(z)$ according to Eq. 2.19
   (b) Determine M_x weighting coefficients for each x_u from $\tilde{B}_{i,k}(x_u)$ in $\mathcal{O}(MN)$
   (c) Sum over all x_u and determine $p(a_i)$ for each bin a_i according to Eq. 2.20 with complexity in $\mathcal{O}(MN)$
   (d) Determine entropy $H(x)$ according to Eq. (2.1) in $\mathcal{O}(M)$ time
2. Calculation of joint entropy of two variables x and y
   (a) Apply steps 1 (a) and (b) to both variables x and y, independently
   (b) Calculate joint probabilities $p(a_i, b_j)$ for all $M_x \times M_y$ bins according to Eq. (2.21) in $\mathcal{O}(M^2N)$ time
   (c) Calculate the joint entropy $H(x, y)$ according to Eq. (2.2) in $\mathcal{O}(M^2)$ time
3. Calculate the mutual information $MI(x, y)$ according to Eq. (2.7)

Figure 2.6: **Calculation of mutual information with B-spline functions — Algorithm and complexity**

2.2.3 Exemplary calculation of mutual information

Based on a simple example of two artificially generated variables (see Figure 2.7), the mutual information is calculated from the simple binning approach, as well as from the B-spline approach. The same number of $M_x = M_y = 3$ bins is used for both algorithms and the spline order was set to $k = 2$.

Simple binning

The histograms of each of the two variables are composed of three bins. Due to the symmetry of data, all bins are equally populated, $p(a_1) = p(a_2) = p(a_3) = \frac{2}{6}$ for variable x and $p(b_1) = p(b_2) = p(b_3) = \frac{2}{6}$ for variable y. For the marginal entropies follows $H(x) = H(y) = -3 \times \frac{2}{6} \log \frac{2}{6} = \log_2 3 \approx 1.58$. The joint histogram contains just three populated bins with two values in each bin $p(a_1, b_3) = p(a_2, b_2) = p(a_3, b_1) = \frac{2}{6}$. For the joint entropy and the mutual information follows $H(x, y) = \log_2 3$ and $MI(x, y) = \log_2 3$.

B-spline approach

Following the work-flow of the previous subsection, the modified indicator function $\tilde{B}_{i,k}(x)$ is determined according to Eq. (2.19) to $\tilde{B}_{i,k}(x) = B_{i,k}(2x)$ (rule 1(a)). For each value x_u of variable x, three weighting coefficients are determined (rule 1(b)). For each of the three bins, probabilities are determined according to Eq. (2.21)(rule 1(c)). Weighting coefficients and probability values for variable x are summarised in Table 2.1. The analogous procedure is applied to variable y and marginal entropies are calculated $H(x) = H(y) = \log_2(10) - 0.6 \log_2(3) - 0.4 \log_2(4) \approx 1.57$. Both, $H(A)$ and $H(B)$, are slightly smaller than the entropies calculated from the simple binning.
The joint probabilities are calculated to be $p(a_1, b_1) = p(a_3, b_3) = 0$, $p(a_1, b_2) = p(a_2, b_1) = p(a_2, b_3) = p(a_3, b_2) = 0.56/6$, $p(a_1, b_3) = p(a_3, b_1) = 1.24/6$, $p(a_2, b_2) = 1.28/6$ (rule 2 (b)) resulting in $H(x, y) = 2.69$ and $MI(x, y) = 0.45$.

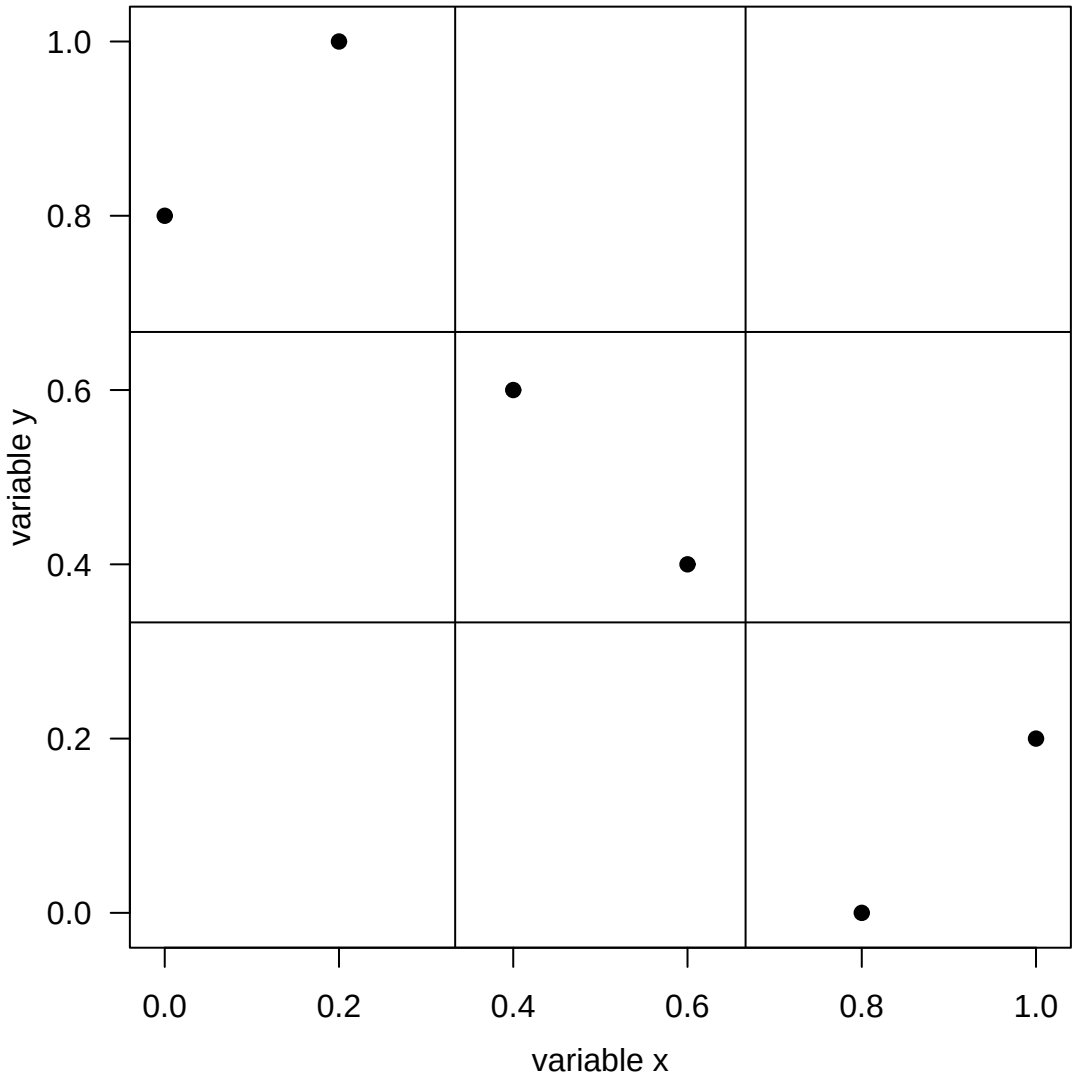

Figure 2.7: **Example: estimating mutual information from data.** For the calculation of the mutual information, the two artificially generated continuous variables x and y have to be binned. The simple binning approach leads to different joint entropies $H(x, y)$ and thereby to different values for the mutual information than the estimation with B-spline functions.

	$B_{i=1,k=2}(x_u)$	$B_{i=2,k=2}(x_u)$	$B_{i=3,k=2}(x_u)$
x_1	1.0	0.0	0.0
x_2	0.6	0.4	0.0
x_3	0.2	0.8	0.0
x_4	0.0	0.8	0.2
x_5	0.0	0.4	0.6
x_6	0.0	0.0	1.0
$p(a_i)$	1.8/6	2.4/6	1.8/6

Table 2.1: For the calculation of probabilities $p(a_i)$ according to the B-spline approach, M_x weighting coefficients are determined for each value x_u of variable x.

Compared to the simple binning, the joint entropy calculated from B-spline functions is significantly increased resulting in a decreased mutual information.

2.2.4 Improvements

In this section, some of the improvements of the estimation of mutual information using B-spline functions are illustrated. Properties arising from this approach are compared to the properties arising from the standard binning approach and from the application of kernel density estimators. Examples are chosen from variables of known distributions because the true mutual information for these examples is known and can be compared to the numerically estimated results. Additionally, we define a significance value to determine the ability of the different estimation procedures to differentiate between data drawn from a given distribution and the null hypothesis of statistical independence.

Sample size

As exemplarily shown in section 2.1.5 and discussed in literature [65, 66, 67, 68, 25], the mutual information is systematically overestimated for a finite size of N data

points. Moreover, the deviation of the true mutual information MI^{true} and the observed mutual information MI^{observed} scales linearly with the inverse size of the dataset for the estimation from bins

$$\left\langle MI^{\text{observed}} \right\rangle \approx MI^{\text{true}} + \frac{(M-1)^2}{2}\frac{1}{N} \tag{2.22}$$

The evaluation of the influence of the size of data on the mutual information is carried out on the basis of artificial data. Datasets of artificially generated equidistributed random numbers of different sizes are generated. The mutual information based on the simple binning, on the 'fuzzy' approach, as well as on kernel density estimators is calculated for all these datasets. From an ensemble of 600 generated datasets, the average mutual information is plotted over the inverse size of the datasets ($1/N$) (Figure 2.8, top). Additionally, the standard deviations are plotted (Figure 2.8, bottom).

It can be observed that the mutual information estimated from the simple binning ($k = 1$) shows a linear scaling in accordance with Eq. (2.22). With increasing size of the datasets ($1/N \to 0$) the mutual information approaches the true value of zero for random data. For the extension to B-spline functions (example with $k = 3$), the linear scaling is preserved and the mutual information also approaches zero for large datasets. It has to be noted that the slope is significantly decreased in contrast to the simple binning. The KDE approach, however, shows an asymptotic run with a linear tail for large datasets with intermediate values in-between the ones for $k = 1$ and $k = 3$.

More importantly, a similar result holds for the standard deviation of the mutual information. In accordance with literature [66, 71], the standard deviation for the simple binning ($k = 1$) scales linearly with $1/N$. This scaling still holds for the B-spline approach ($k = 3$) but the slope is decreased significantly. The standard deviations for KDE, again, lie in-between $k = 1$ and $k = 3$.

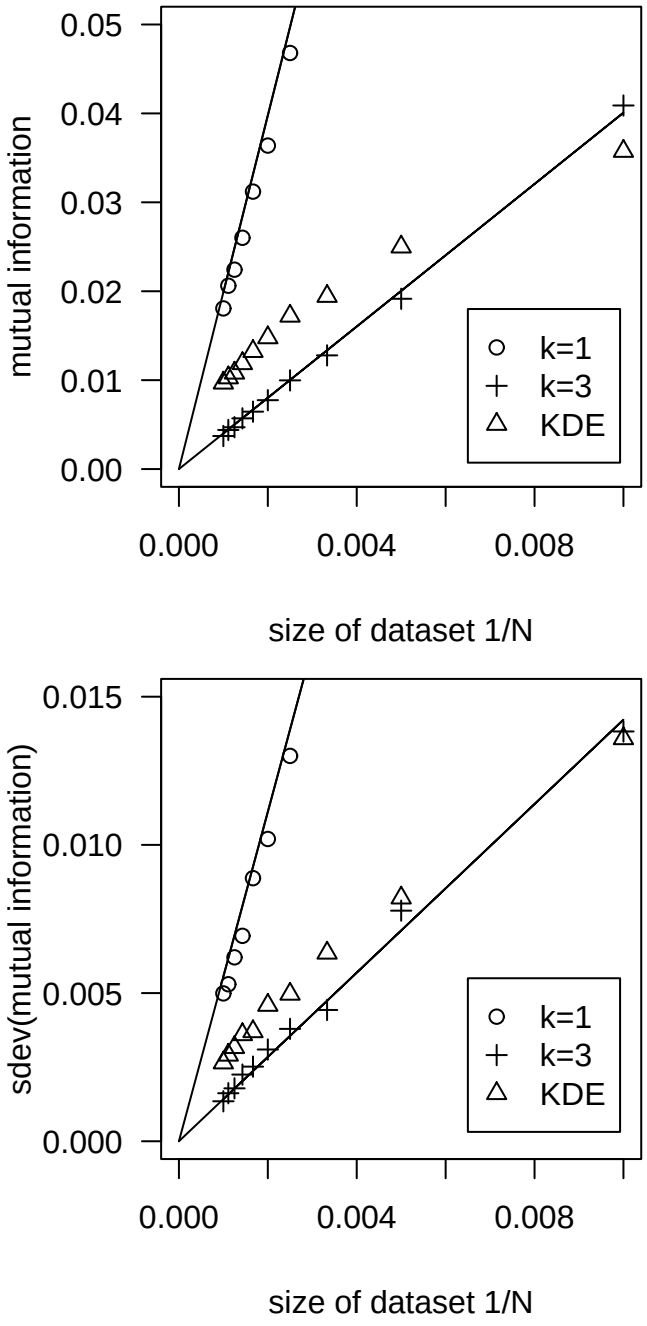

Figure 2.8: **Sample size.** The mutual information is calculated for artificially generated equidistributed random numbers for which the true mutual information is known to be zero. The average over an ensemble of 600 trials is shown as a function of the inverse size of the dataset $1/N$ (top). The standard deviation of the 600 trials for each dataset size is plotted over $1/N$ (bottom). The numerical estimation was done for two spline orders, $k = 1$ and $k = 3$ using $M = 6$, bins and for the kernel density estimation approach. The lines show a least squares fit fixed in the origin.

Spline order and number of bins

In the following, the influence of the spline order k and the number of bins M on the estimation of the mutual information is characterised. Data for this evaluation will be drawn from the distribution shown in Figure 2.1.

The interpretation of the results obtained from the subsequent analysis bases on the testing whether the calculated results are consistent with a previously chosen null hypothesis. Following an intuitive approach, the null hypothesis assumes statistical independence of variables. The mutual information calculated from experimental data is therefore tested against a surrogate dataset, which is consistent with the chosen null hypothesis of statistical independence. One commonly used way to generate such a dataset (see [25] for details and other methods) is by random permutations of the original dataset. A *significance* S can then be defined from the mutual information of the original dataset MI^{data}, the average value obtained from surrogate data $\langle MI^{\text{surr}} \rangle$, and its standard deviation σ^{surr}

$$S := \frac{MI^{\text{data}} - \langle MI^{\text{surr}} \rangle}{\sigma^{\text{surr}}} \tag{2.23}$$

Depending on the distribution of the calculated mutual information and the significance S, the null hypothesis can be rejected to a certain level α. For the ideal case of Gaussian distributed mutual information values, $S \geq 2.6$ can be treated as significant at a level of 99%. For distributions diverging from a Gaussian distribution, a more general reasoning was suggested [72, 73].

The influence of the spline order k on the estimated mutual information is evaluated on a dataset drawn from the distribution shown in Figure 2.1. In contrast to the previously chosen datasets drawn from equidistributed random numbers, this dataset is constructed to represent 'real' measurements on variables, e.g. on genes or on metabolites. For a dataset constructed in such a way consisting of 300 data points, the mutual information is calculated for a range of spline orders $k = 1 \ldots 5$ (Figure 2.9). The number of bins for the estimation was chosen to $M = 6$. From 300 shuffled realisations of the dataset, the mean and maximum mutual information are shown with standard deviation as error-bars.

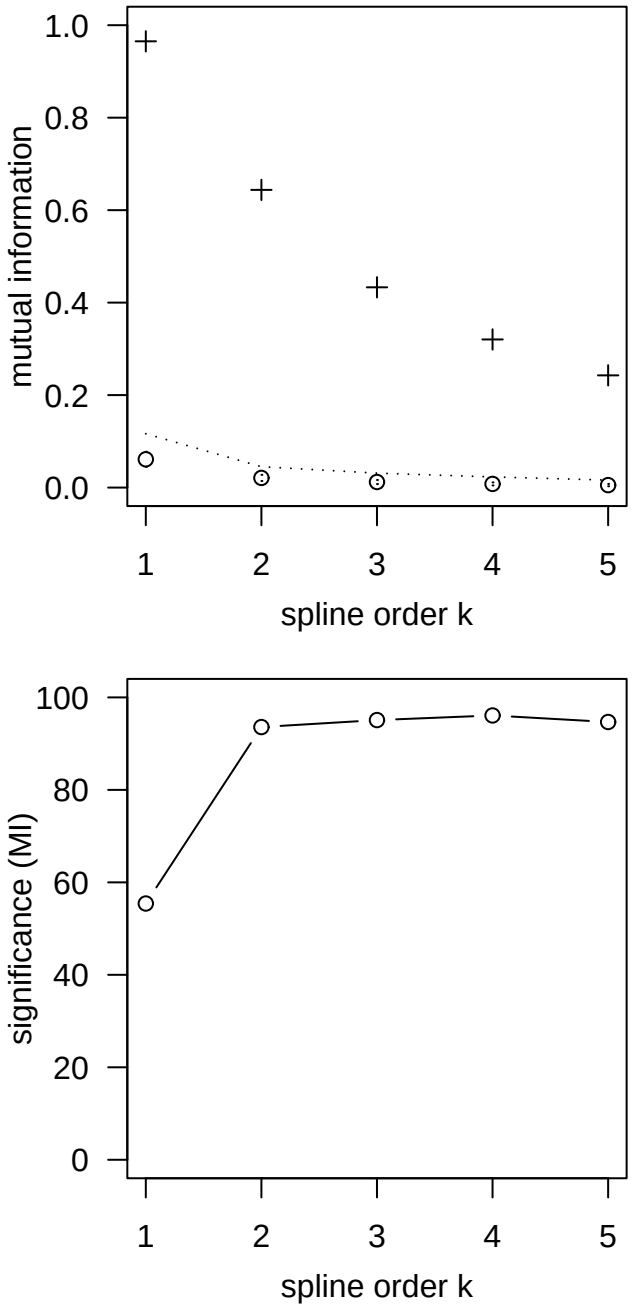

Figure 2.9: **Spline order.** For a dataset of 300 data points representing a 'real' measurement, drawn from the distribution shown in Figure 2.1 (top), the mutual information is calculated (top, crosses). The number of bins is fixed to $M = 6$. For 300 shuffled realisations of this dataset, the average mutual information (top, circles) together with standard deviations as error-bars is shown. The largest value found within the ensemble of shuffled data is drawn as dotted line (top). The significance according to Eq. (2.23) is calculated (bottom).

As can be observed from the difference of the largest values obtained from shuffled data (Figure 2.9, top, dotted line) to the values obtained from the 'real' measurements (Figure 2.9, top, crosses), the null hypothesis can be rejected for all shown spline orders. This result runs in accordance with the example shown in Figure 2.1.

To estimate the strength of the rejection, the significance according to Eq. (2.23) is calculated (Figure 2.9, bottom). It can be seen that the largest change in significance occurs in the transition from $k = 1$ (simple boxes) to $k = 2$ with an increase by roughly two-fold. The utilisation of more sophisticated functions ($k \geq 3$) does not further improve the significance. In the context of kernel density estimators, similar findings have been reported [69]. This finding might arise from the distribution of surrogate data: As can be seen in Figure 2.9 (top, circles), the standard deviation of surrogate data σ^{surr} is decreased for $k \geq 2$ compared to $k = 1$ leading to an increase of significance according to Eq. (2.23).

Based on the same dataset, the dependency of the mutual information on the number of bins M is evaluated. For the exemplarily chosen spline orders $k = 1$ and $k = 3$ the mutual information is calculated from 300 data points (Figure 2.10) drawn from the distribution shown in Figure 2.1. Again, from 300 shuffled realisations the mean and maximum mutual information are shown with standard deviation as error-bars.

The mutual information calculated from data, as well as from surrogate data, shows a robust run without strong fluctuations. From this it can be concluded that the choice of the number of bins does not affect the resulting mutual information notably as long as it is chosen to be within a reasonable range. Even though the absolute numbers are effected, especially the relation to surrogate data is of importance.

For this, the significances for both spline orders are calculated (Figure 2.11) according to Eq. (2.23) and compared to the significances obtained from the KDE approach. It can be observed that the significances of the mutual information calculated with B-spline functions increased roughly by two-fold compared to the simple binning. The significances obtained from KDE are not depending on the number of bins M and were determined to be similar to the significances estimated from the

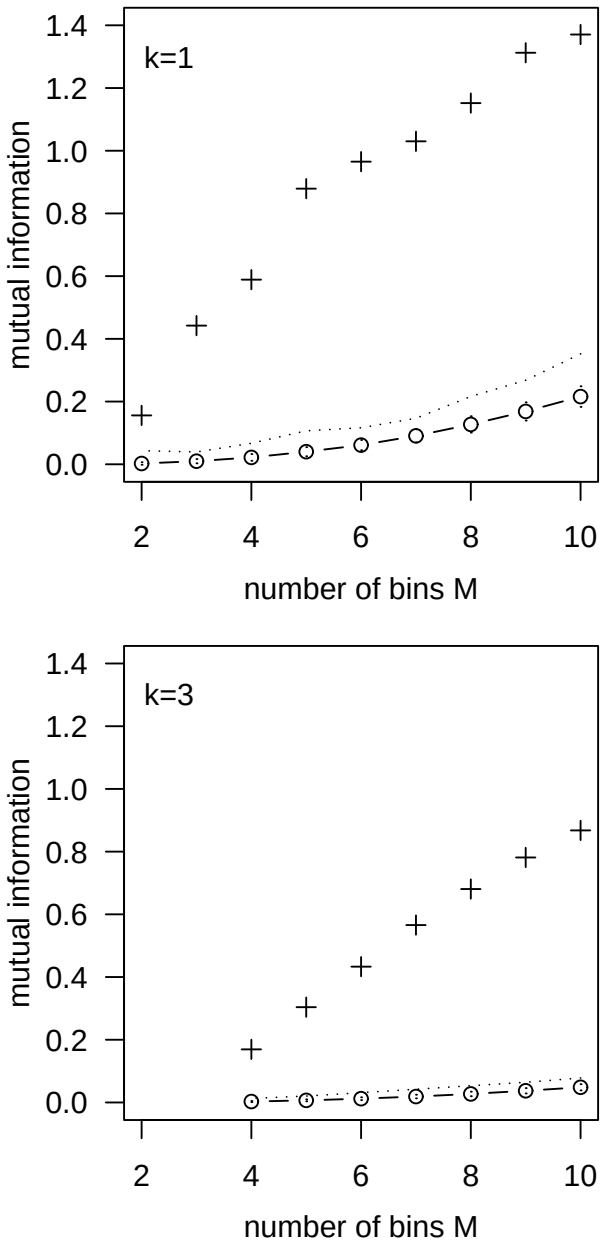

Figure 2.10: **Number of bins.** Drawn from the distribution shown in Figure 2.1, the mutual information is calculated from 300 data points for two spline orders $k = 1$ (top) and $k = 3$ (bottom) as a function of the number of bins M (crosses). Calculated from 300 surrogates the mean (circles), standard deviations (error-bars), and largest values (dotted line) are shown.

B-spline approach. The numerically expensive integration of KDE, however, limits the size of utilisable datasets. The KDE run time requirements were $\mathcal{O}(10^4)$ times higher than the ones from the B-spline approach. Strategies to simplify the integration step were proposed [25] but have to be used with caution since they assume particular properties of the distribution of experimental data that are in general not fulfilled.

After this introduction into the mutual information and our suggestions for an improved estimation, we turn towards the development of software tools.

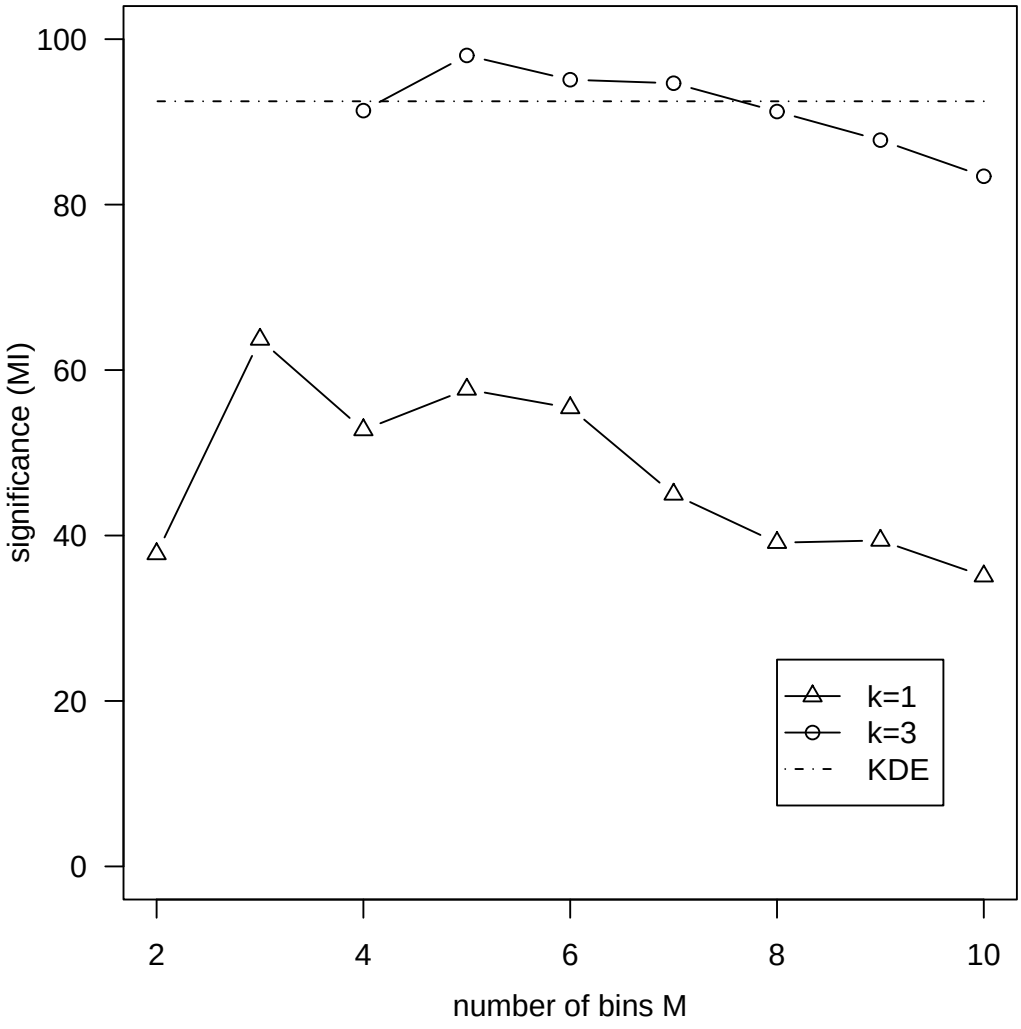

Figure 2.11: **Significance.** The significance S is shown as a function of the number of bins M for the two examples of Figure 2.10. For kernel density estimators (KDE), the significance is not depending on M and was estimated to $S = 92$.

2.3 Software development

Within the scope of the work presented here, two software packages have been developed. With the first package, the mutual information can be calculated for continuous data. It includes an implementation of the extension dicussed in this thesis. The second software is a web-based service for the analysis of data obtained from different measurement technologies.

2.3.1 Calculating the mutual information — *mis_calc*

The algorithm for the numerical estimation of the mutual information from B-spline functions is implemented into the software package *mis_calc* under the programming language C++.

mis_calc runs on the command line. The data has to be provided in the form of a data matrix. Each row, containing the data for one variable, is regarded as a vector of data and pairwise comparisons of all vectors are carried out. The mutual information for each comparison is calculated.

Besides the mandatory arguments like the bin number and the spline order, several optional arguments can be passed on the command line. The program supports different output formats, additional calculation of the Pearson correlation coefficient, the prior shuffling of data for significance analyses, as well as a multi-processor threading option. The progress of the calculation optionally can be visualised with a graphical animation which is based on the graphical library *qt*[1]. The implementation was carried out in close collaboration with Sebastian Kloska.

In the default settings, first the marginal entropies of all vectors are calculated and then used for the calculation of a particular pair of vectors. When the dataset contains undefined values, however, the joint entropy for two vectors might base on different number of entries. The marginal entropies for such vectors also need to be

[1] http://www.trolltech.com/qt/

recalculated so that the marginal, as well as the joint entropy are estimated from the same values. For such cases, *mis_calc* offers an *on-the-fly* option. This option has to be used with caution, since the mutual information depends on the size of the sample it is calculated from (see discussion about *finite size effect* below). Mutual information values calculated from different numbers of data points are no longer comparable unless a correction is appended.

The package was developed with the GNU C++ compiler[2] and the recent version, *mis_calc v6.3*, was compiled with *gcc v3.3.2* under the operating system Linux.

A natural approach for the calculation of the mutual information is the implementation of the measurements in the form of a two dimensional matrix (see Table 1.1). The object representation of the matrix should allow for the manipulation of data and should handle operations on the whole set or on subsets in a way that facilitates the solution of the given problem. Furthermore, we decided to design the system in a way that enables different threads to work on the solution in parallel. Since the calculation of mutual information for a single row is numerically independent from the calculation of all other pairwise comparisons of rows, *mis_calc* benefits very much from a multiprocessor system when designed in this fashion. The recent version runs in multiple instances. These instances might be distributed over several processors of one computer or over different computers where each instance communicates via the *CORBA*[3] interface with a master instance that tracks progress and status of the slave programs.

Details about the implementation of *mis_calc* are given in the appendix.

2.3.2 MetaGeneAlyse

The analysis of data demands software tools that comprise the algorithms and visualisation techniques required for the specific type of data under consideration. For this, we developed the software package *MetaGeneAlyse* [74] for the analysis of

[2] http://www.gnu.org/ software/gcc
[3] http://www.corba.org

data from gene expression measurements, metabolite measurements, and datasets merged from both types of data.

Overview

MetaGeneAlyse is a web-based service which allows for the analysis of integrated data containing gene-expression data and metabolite data. At the web front-end[4] (Figure 2.12, center), a tabular-separated dataset can be uploaded (Figure 2.12, top left). As file format we chose the general ASCII text table format containing genes and/or metabolites in rows and experiments in columns. As normalisation is an important preprocessing step towards the analysis of integrated data, several normalisation algorithms were implemented (Figure 2.12, top right), like normalisation to the maximum, mean, median, variance, standard deviation, vector norm, root mean square, z-score, and others. Analysing directly the normalised dataset can be done with k-means clustering and principal component analysis (Figure 2.12, bottom right). Different types of hierarchical clustering, which are based on the prior calculation of a distance matrix, are also implemented (Figure 2.12, bottom left). The underlying distance matrix is then calculated as first step using a distance measure (e.g. Euclidean or Manhattan distance, Pearsons correlation coefficient or mutual information). The distance matrices can also be downloaded to be post-processed by the user. Different file formats for the download are available to directly import the distance matrix into graph visualisation tools like Pajek[5]. Analyses on shuffled datasets are useful for the calculation of significance levels, therefore our service allows the shuffling of uploaded datasets. Documentation is available in HTML and PDF format and can be downloaded from the website.

Concept of data access

A common way of interaction with web-based services is via the authorisation of previously registered users. After a personal log on, each user gets access to her or

[4] http://metagenealyse.mpimp-golm.mpg.de
[5] http://vlado.fmf.uni-lj.si/pub/networks/pajek/

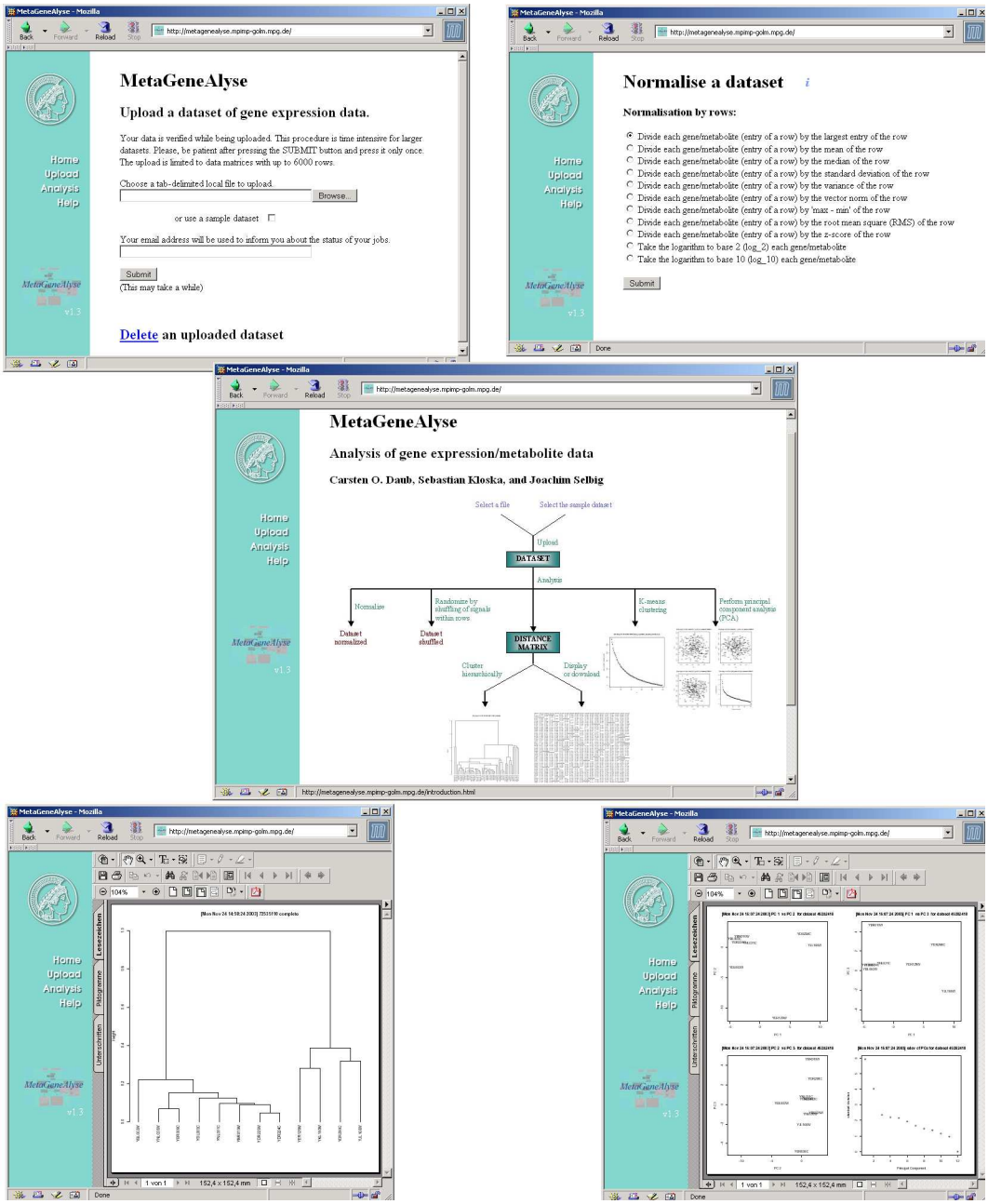

Figure 2.12: **MetaGeneAlyse Overview.** In the web-interface of MetaGeneAlyse (center), integrated datasets containing gene-expression and/or metabolite data can be uploaded (top left) and normalised (top right). Among various analysis methods, principal component analysis (bottom right) and hierarchical clustering (bottom left) are implemented.

his data. This approach, however, acts discouraging upon many users when they get in contacts with the tool. It also requires the implementation and support of a user administration. Therefore, MetaGeneAlyse realises a different concept: An 8 digit long random ID number is assigned to each intermediate result of the analysis procedure. At the dataset upload, a valid email address has to be provided for the identification of the owner the dataset (see Figure 2.12, top left). This email address is inherited by all subsequent analysis results. Thus, the user can access all intermediate result of her or his analysis and re-check or continue a previous analysis in a different way. With knowledge about the ID and the email address of the owner of a dataset, it also can be deleted.

2.4 Application on data

The evaluation of gene expression data [26] and metabolite data [27, 28, 29] is often carried out on the basis of clustering algorithms. In the context of gene expression data, mutual information has shown to reveal biologically relevant clusters of genes [24, 77].

In the first example, the global question is asked whether mutual information is able to detect non-linear correlations in large scale gene-expression datasets. After this, a clustering analysis is applied to a large scale gene expression dataset and the resulting clusters are evaluated by annotations of genes. Finally, an integrated dataset containing gene expression data and metabolite data for sulfur depletion time series experiments on *Arabidopsis thaliana* is analysed.

2.4.1 Global comparison of MI to Pearson correlation

Linear measures, such as the Pearson correlation coefficient, are among the most frequently used similarity measures even in recent publications [78]. In this example, non of the above mentioned analyses are repeated or deepened, but it is determined whether any correlation detected using mutual information would be missed by solely applying the Pearson correlation coefficient.

In the ideal case of Gaussian distributed data, the relationship between the mutual information and the Pearson correlation for genes X and Y reads [79]

$$R(X,Y) = \sqrt{1 - \exp(-2\,MI(X,Y))} \tag{2.24}$$

For this, the mutual information and the Pearson correlation are calculated for two large-scale gene expression datasets. From the pairwise comparison of all genes within a dataset, the tuple $(MI(X,Y), R(X,Y))$ is plotted (Figure 2.13) together with the prediction of Eq. (2.24).

The first dataset contains cDNA measurements for *Saccharomyces cerevisiae* for up to 300 experiments [80]. To avoid numerical effects arising from different numbers

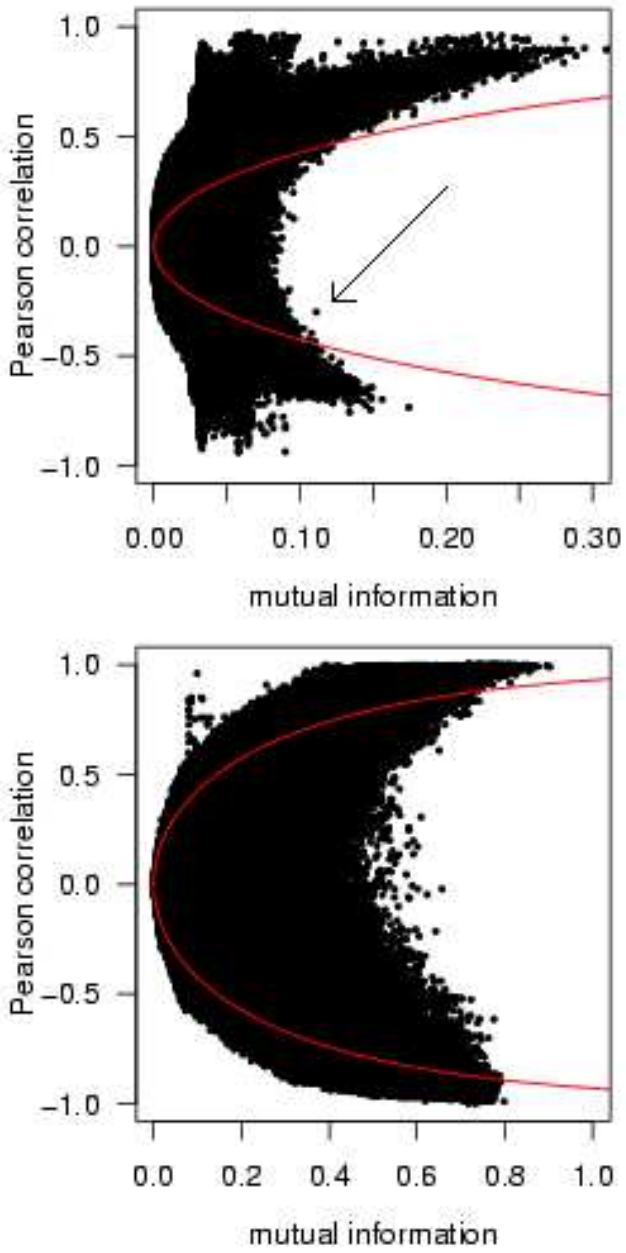

Figure 2.13: **Comparison of mutual information and Pearson correlation coefficient.** The Pearson correlation coefficient and the mutual information for all pairwise comparisons of genes for two large-scale gene expression datasets is shown together with the expected mutual information calculated from Eq. (2.24). For the first dataset (top) genes containing undefined values were omitted resulting in 5345 genes measured under 300 experimental conditions [80]. For the second dataset (bottom) containing 22608 genes measured under 102 experimental conditions [78], a representative fraction is shown.

of defined expression values (missing data points) for each gene, exclusively genes that are fully defined for all experimental conditions were utilised resulting in 5345 genes[6]. Details for this dataset are given in the appendix.

First, some well-known facts can be confirmed: the Pearson correlation distinguishes between positive and negative correlation, whereas the mutual information gives exclusively positive correlations. Positive Pearson correlations are more frequent than negative ones. Further, the relation of the Pearson correlation to the mutual information follows in principle the theoretical prediction but with large fluctuations. Many tuples showing high Pearson correlation and low mutual information can be detected. These gene-gene pairs, with high linear correlation but a low statistical independence, arise from outlying expression values, as exemplarily shown for a gene-gene comparison in Figure 2.14, A. In contrary, tuples with low Pearson correlation but high mutual information, thus indicating non-linear correlations, are not observed. The only remarkable tuple, marked with an arrow in Figure 2.13 and depicted in Figure 2.14, B, arises also from outlying values.

The second dataset contains cDNA measurements for 102 experiments on 22608 genes derived from 20 different human tissues [78]. In contrast to the first dataset, tuples with low Pearson correlation but high mutual information are indeed detected. For two exemplary chosen tuples, as depicted in Figure 2.14 C and D, clusters of data points (each data point refers to one experimental condition) can be clearly detected by eye. Such type of correlations are missed by analyses based exclusively on linear measures, such as the the analysis done in the original publication of this dataset.

In summary, the analysis confirms for the first dataset that the Pearson correlation cofficient does not miss any non-linear correlations. As a side effect, gene-gene pairs containing outlying values were detected. For the second dataset, however, a substantial amount of non-linear correlations was detected. Gene-gene pairs exemplarily chosen from this fraction show a clustering of data points (experiments).

[6]The datasets actually contains open reading frames (ORF). The difference between ORFs and genes in the case of yeast is that the genes additionally contain the promoter and terminator sequences. The term genes is used synonymously to the term ORF throughout this example

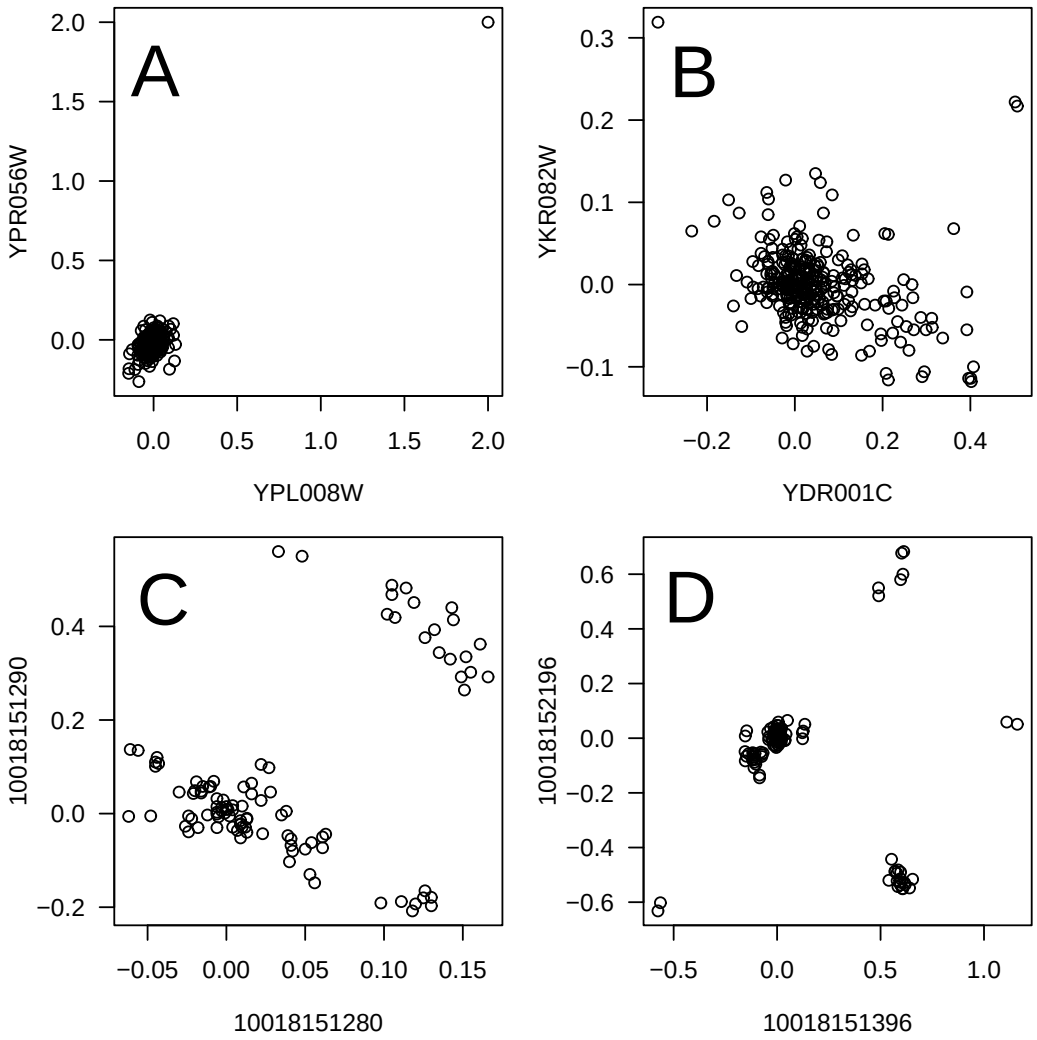

Figure 2.14: **Remarkable gene-gene comparisons.** Examples of gene-gene plots for two genes X and Y are shown for characteristic tuples $(MI(X,Y), R(X,Y))$ detected in Figure 2.13. For the first gene expression dataset under consideration [80], no non-linear correlations are detected. Moreover, tuples with high Pearson correlation and low mutual information, examples A and B, resulting from outlying values are detected. For the second dataset [78], however, tuples with low Pearson correlation and high mutual information are observed, see examples C and D. Such non-linear correlations are missed by solely using linear correlation measures.

Even though such patterns of expression values can be easily detected by eye, the enormous number of gene-gene comparisons (there are around 255 million pairwise comparisons for the second dataset) requires computational methods for the detection of such patterns.

2.4.2 Application of MI on a gene expression dataset

The clustering of co-expressed genes [26] is a frequently applied method for the evaluation of regulatory networks. Thereby, the choice of the underlying similarity measure is as crucial as the choice of the clustering method itself [50]. However, the appropriateness of similarity measures have not been systematically explored [81] and often they are used on an ad-hoc basis without justification.

Now, the global comparison of mutual information to a linear measure for the two gene expression datasets in section 2.4.1 is extended. Thereby, mutual information is used as a measure of similarity for the hierarchical clustering of expression data for one of the already introduced datasets [80]. Details for this dataset are given in the appendix. The resulting clusters of genes are evaluated on the basis of annotations of the Munich Information Center for Protein Sequences (MIPS)[7].

Even though it was shown in the last section (2.4.1) that this dataset does not contain non-linear correlations, the utilisation of mutual information as similarity measure might reveal additional functional relationships: Similarity measures evaluate and thereby rank the correlations. The Pearson correlation coefficient exclusively detects linear correlations and thereby tends to underestimate correlations aside linear ones.

To this end, a similarity matrix containing all pairwise comparisons of genes based on mutual information is calculated. We choose the number of bins to $M = 8$. This seems reasonable for the dataset under analysis containing 300 measurements for each gene (see also the discussion about a reasonable bin number in section 3.2). The spline order is set to $k = 2$. Following a simple clustering approach, all gene-gene associations below a certain threshold of similarity are removed. This procedure corresponds to the *single linkage* clustering. It chains two clusters together if they are connected with just one single association. Thereby, it tends to build few large clusters. On the other hand it is able to detect branched or bended structures.

The choice of an appropriate threshold has a large influence on the resulting clusters,

[7] http://mips.gsf.de/

size of clusters	181	84	32	30	12	10	8	7	6	4	3	2
frequency of clusters	1	1	1	1	1	1	1	2	3	2	8	28

Table 2.2: **Distribution of cluster sizes.** Application of single linkage clustering to a large-scale gene expression dataset [80] results in clusters of different sizes for the threshold shown in Figure 2.15.

as shown in Figure 2.15. For small thresholds, many associations are left and all genes fall into one cluster. With increasing threshold, more separated clusters are build. From a certain threshold on, the number of clusters again decreases because genes are left without any association. These genes are not regarded as clusters. The number of genes, as well as the number of associations used for the clustering continuously decrease with increasing thresholds. The connectivity, which is defined as the number of actual links compared the the number of potential links, increases with increasing the threshold.

For the subsequent analysis, the threshold leading to the maximum number of 50 clusters was chosen. The threshold is marked with a dotted line in Figure 2.15. Even though this choice is arbitrary, it is supported by the assumption that the biological interpretation on a small number of large clusters or on a large number of cluster containing just a few genes each might not be reasonable. The distribution of the clusters sizes is typical for single linkage clustering (see Table 2.2).

The Munich Information Center for Protein Sequences (MIPS) provided annotations for the whole genome of *Saccharomyces cerevisiae*. They are hierarchically organised in 30 main classes with different levels of accuracy (Figure 2.16)[8]. Each gene is annotated with one or several class identifiers and additionally with a free text description (Figure 2.17).

The generated clusters were evaluated on the basis of these biological annotations:

[8]MIPS does no longer provide this information as table but only in databases that are accessible at http://mips.gsf.de/genre/proj/yeast/index.jsp. The original tables can still be accessed via http://rsat.ulb.ac.be/rsat/data/genomes/Saccharomyces_cerevisiae/catalogs/ at the files mips_orf_class_description.tab and mips_functional_catalog_scheme.tab

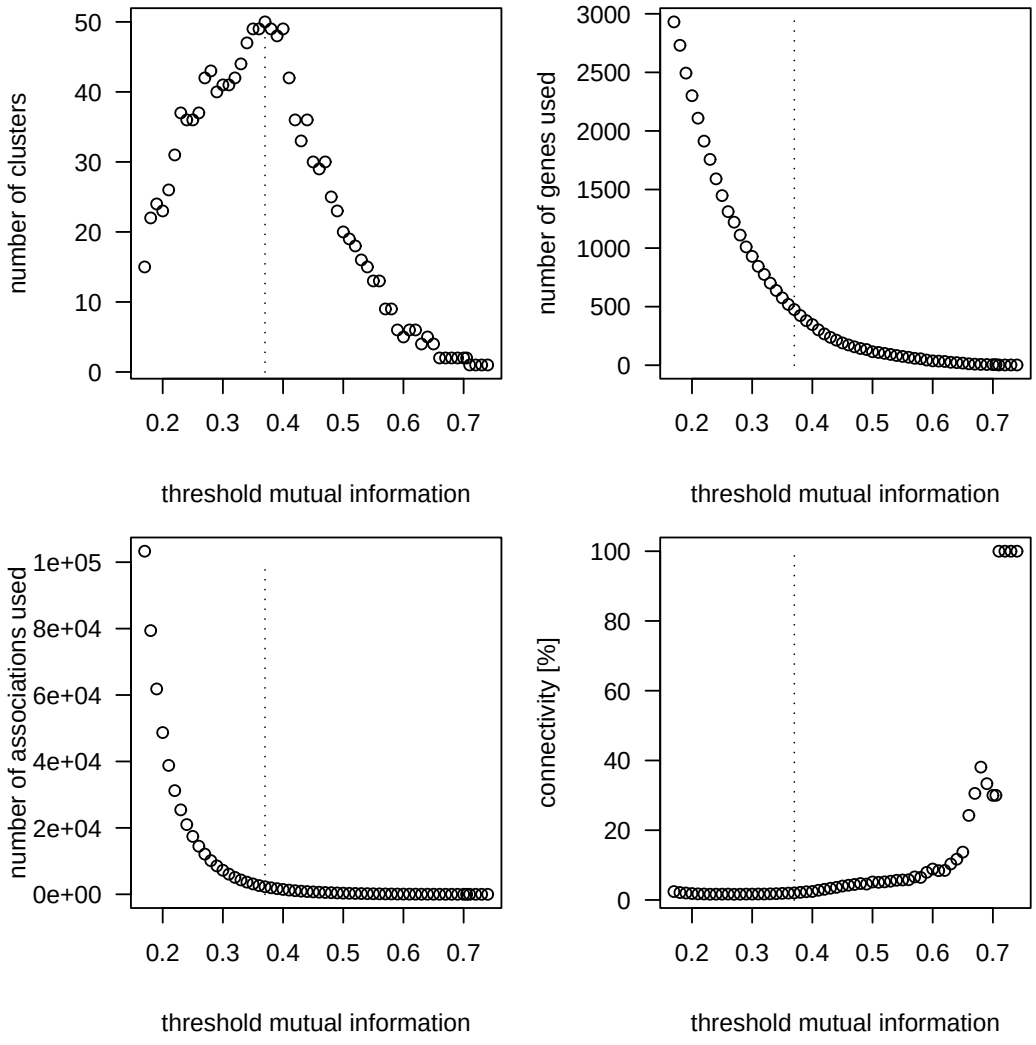

Figure 2.15: **Characterisation of clusters.** A large scale gene expression dataset [80] was hierarchically clustered based on mutual information as similarity metric. The clustering result depends on a threshold that is chosen for the single linkage clustering: the number of obtained clusters (left, top), the number of used genes (right, top), the number of associations used for the clustering (left, bottom), and the connectivity (right, bottom) are shown subject to the threshold.

```
01 METABOLISM
02 ENERGY
03 CELL CYCLE AND DNA PROCESSING
04 TRANSCRIPTION
05 PROTEIN SYNTHESIS
...
65 STORAGE PROTEIN
67 TRANSPORT FACILITATION
96 OUT OF USE (Historical categories, please update annotation
   for all ORFs in this categorie)
98 CLASSIFICATION NOT YET CLEAR-CUT
99 UNCLASSIFIED PROTEINS
-------------------------------------------------------------
1 METABOLISM
01.01 amino acid metabolism
01.01.01 amino acid biosynthesis
01.01.01.01 assimilation of ammonia,
            biosynthesis of the glutamate group
01.01.01.01.01 assimilation of ammonia
01.01.01.01.02 biosynthesis of the glutamate group
...
02 ENERGY
02.01 glycolysis and gluconeogenesis
02.01.01 glycolysis methylglyoxal bypass
02.01.03 regulation of glycolysis and gluconeogenesis
02.05 Entner-Doudoroff pathway
...
03 CELL CYCLE AND DNA PROCESSING
03.01 DNA processing
03.01.01 cellular DNA uptake
03.01.01.01 bacterial competence
...
```

Figure 2.16: **Annotation — listing of main classes.** The Munich Information Center for Protein Sequences (MIPS) provides annotations for the whole genome of *Saccharomyces cerevisiae*. The annotations are hierarchically organised in 30 main classes (cut-out shown in top) with different levels of accuracy (bottom).

```
...
YAL042w   08.07 S.cerevisiae;
          ERV46 - protein involved in vesicular transport between the
          endoplasmic reticulum and Golgi apparatus
YAL041w   14.04.03.01;14.04.03.03;03.03.01;62.02.05;13.11.03.07;
          62.02.05;14.01.03;40.03 S.cerevisiae;
          CDC24 - GTP/GDP exchange factor for Cdc
YAL040c   14.01;14.04.03.01;03.01.03;03.03.01;40.03 S.cerevisiae;
          CLN3 - cyclin, G1/S-specific
YAL039c   01.07.04;06.07;40.16 S.cerevisiae;
          CYC3 - holocytochrome-c synthase (cytochrome c heme lyase)
YAL038w   01.05.01;02.01;40.03 S.cerevisiae;
          CDC19 - pyruvate kinase
YAL037c-a 99 S.cerevisiae;
          putative ORF - identified by SAGE
YAL037w   99 S.cerevisiae;
          FUN11 - strong similarity to GTP-binding proteins
YAL036c   99 S.cerevisiae;
          FUN11 - strong similarity to GTP-binding proteins
...
```

Figure 2.17: **Annotation — listing of genes.** MIPS provides annotations according to functional classes and additionally in free format for the whole genome of *Saccharomyces cerevisiae*. A gene can be annotated to several classes.

cluster 1			cluster 2			cluster 3		
class	genes	total	class	genes	total	class	genes	total
1	5	84						
			2	1	32			
3	4	84						
4	24	84						
5	3	84	5	16	32	5	28	30
6	3	84	6	3	32	6	2	30
8	1	84						
10	1	84						
11	2	84						
			13	1	32			
14	2	84				14	1	30
30	1	84						
40	22	84	40	18	32	40	28	30
98	3	84	98	1	32			
99	41	84	99	12	32	99	1	30

Table 2.3: **Evaluation of clustering results.** The result of a clustering is evaluated on the basis of biological annotations which are organised in annotation classes.

the frequency of assignments of annotation classes to clusters were determined. For three of the largest clusters, containing 84, 31, and 29 genes, a detailed analysis was performed. The number of genes falling into each main class, as shown in Figure 2.16, are listed for each of the three clusters in Table 2.3.

For the first cluster, nearly half of the genes (41 out of 84) have not yet been classified (class `99`). Within the next two most populated classes, the class TRANSCRIPTION (`04`) is mainly represented by the subclass *RNA transcription* (`04.01`) with 18 of 24 genes (data not shown). The class SUB-CELLULAR LOCALISATION (`40`) is mainly represented by the subclass *nucleus* (`40.10`, data not shown) with 18 of 23 genes with an overlap to the *RNA transcription* subclass of 15 genes. Without consideration of the unclassified genes, the first cluster is mainly represented by genes participating in the **RNA transcription in the nucleus**.

YGL069c	99	questionable ORF
YPR099c	99	questionable ORF
YMR158w-b	99	questionable ORF
YKL169c	99	questionable ORF
YJL104w	99	weak similarity to C.elegans hypothetical protein F45G2.c
YPR100w	99	weak similarity to C.elegans hypothetical protein CEC25A1
YJR101w	99	RSM54 - weak similarity to superoxide dismutases
YKL195w	99	similarity to rabbit histidine-rich calcium-binding protein
YIL093c	99	RSM53 - weak similarity to S.pombe hypothetical protein SPBC16A3
YMR158w	99	weak similarity to E.coli ribosomal S8 protein
YGR021w	99	similarity to M.leprae yfcA protein
YHR116w	99	weak similarity to TRCDSEMBLNEW:AE003592_6 CG4186 D. melanogaster

Table 2.4: **Assigning functions to unclassified genes.** Within the second cluster containing 32 genes, 12 gene are annotated as *unclassified*. It is verified for these genes if their sequences overlap with already annotated genes of if other organisms contain similar sequences with annotations that match with annotations already found in this cluster.

The second cluster mainly contains genes annotated to PROTEIN SYNTHESIS - *ribosome biogenesis* (`05.01`) for 16 genes and SUB-CELLULAR LOCALISATION - *mitochondrion* (`40.16`) for 16 of a total of 18 genes. Thus, this cluster is mainly containing genes annotated as **ribosomal biogenesis in the mitochondrion**.

For the 12 unclassified genes in this cluster it was verified by their position in the genome whether they correspond to other already annotated genes (see Table 2.4). Four of the questionable genes (YGL069c, YPR099c, YMR158w-b, YKL169c) showed to strongly overlap with genes that already were part of the same cluster (YGL068w, YPR100w, YMR159c, YKL170w, respectively). Based on the protein sequences of the remaining 8 genes, BLASTP[9] searches [82] were performed. They aimed at finding similar genes in other species that share the same annotations as the genes already participating in this cluster. We accepted matches with an e-value below 10^{-4} as significant. For 6 of these 8 genes (YJL104w, YPR100w, YJR101w, YKL195w, YIL093c, YMR158w), the top scoring BLASTP results showed genes with free text annotations containing the keywords *mitochondria* or *ribosome* (one example of the BLASTP search for gene YMR158w is shown in table 2.5). For the remaining two genes, there was no BLASTP hit indicating ribosomal annotations found.

[9]http://www.ncbi.nlm.nih.gov/BLAST/

name	description	raw_score	evalue
gi-6323808-ref-NP_013879.1	Mitochondrial Ribosome Protein,...	306	8e-83
gi-19075281-ref-NP_587781.1	hypothetical protein [Schizosa...	118	3e-26
gi-32419391-ref-XP_330139.1	hypothetical protein [Neurospo...	97	8e-20
gi-22297638-ref-NP_680885.1	30S ribosomal protein S8 [Ther...	52	3e-06
gi-3122821-sp-O24702-RS8_SYNP6	30S ribosomal protein S8 gi...	50	1e-05
gi-32423687-gb-AAP81230.1	ribosomal protein S8 [Candidatus...	50	2e-05
gi-34764952-ref-ZP_00145288.1	SSU ribosomal protein S8P [F...	50	2e-05
gi-15599445-ref-NP_252939.1	30S ribosomal protein S8 [Pseu...	49	2e-05
gi-17231694-ref-NP_488242.1	30S ribosomal protein S8 [Nost...	48	4e-05
gi-19704951-ref-NP_602446.1	SSU ribosomal protein S8P [Fus...	48	5e-05
gi-16329928-ref-NP_440656.1	30S ribosomal protein S8 [Syne...	48	6e-05
gi-11467730-ref-NP_050782.1	ribosomal protein S8 [Guillard...	47	1e-04
gi-33152940-ref-NP_874293.1	30S ribosomal protein S8 [Haem...	47	1e-04
gi-11465767-ref-NP_053911.1	ribosomal protein S8 [Porphyra...	46	2e-04
gi-30352068-ref-NP_848095.1	ribosomal protein S8 [Adiantum...	45	3e-04
gi-23104453-ref-ZP_00090917.1	COG0096: Ribosomal protein S...	45	3e-04
gi-16125511-ref-NP_420075.1	ribosomal protein S8 [Caulobac...	45	3e-04
gi-23468111-ref-ZP_00123672.1	COG0096: Ribosomal protein S...	45	5e-04
gi-1173279-sp-P12879	RS8_BACSU 30S ribosomal protein S8 (BS...	45	5e-04
gi-32030989-ref-ZP_00133685.1	COG0096: Ribosomal protein S...	45	5e-04
gi-28202208-ref-NP_777449.1	ribosomal protein S8 [Anthocer...	45	5e-04
gi-32034703-ref-ZP_00134841.1	COG0096: Ribosomal protein S...	45	6e-04
gi-15603266-ref-NP_246340.1	RpS8 [Pasteurella multocida] ...	44	8e-04

Table 2.5: **Example of BLASTP result.** BLASTP searches were performed for genes that group together with genes annotated my MIPS as *ribosomal proteins* in the *mitochondrion*. For these *unclassified* genes, it was intended to find genes in other species with similar annotations.

The third cluster contains 30 genes of which 28 were annotated as PROTEIN SYNTHESIS - *ribosome biogenesis* (`05.01`) and SUB-CELLULAR LOCALISATION - *cytoplasm* (`40.03`). One of the 2 *unclassified* genes (YGL102c) is overlapping with another gene (YGL103w) which is annotated to (`05.01`) and (`40.03`). Thus, genes in this cluster are nearly exclusively annotated as **ribosomal biogenesis in the cytoplasm**.

It can be summarised that the hierarchical clustering of a large-scale gene expression dataset based on the mutual information as similarity measure leads to a meaningful grouping of genes according to their biological annotation. From the whole dataset containing 5345 genes, three subsets of genes were exemplarily identified (Figure 2.18). These subsets refer to genes taking part in the

- RNA transcription in the nucleus,
- protein synthesis in the mitochondrion, and
- protein synthesis in the cytoplasm.

The clustering method was exemplarily chosen without comprehensive evaluation. A different choice of this parameter will result in different clusters and thereby will reveal different aspects of the dataset.

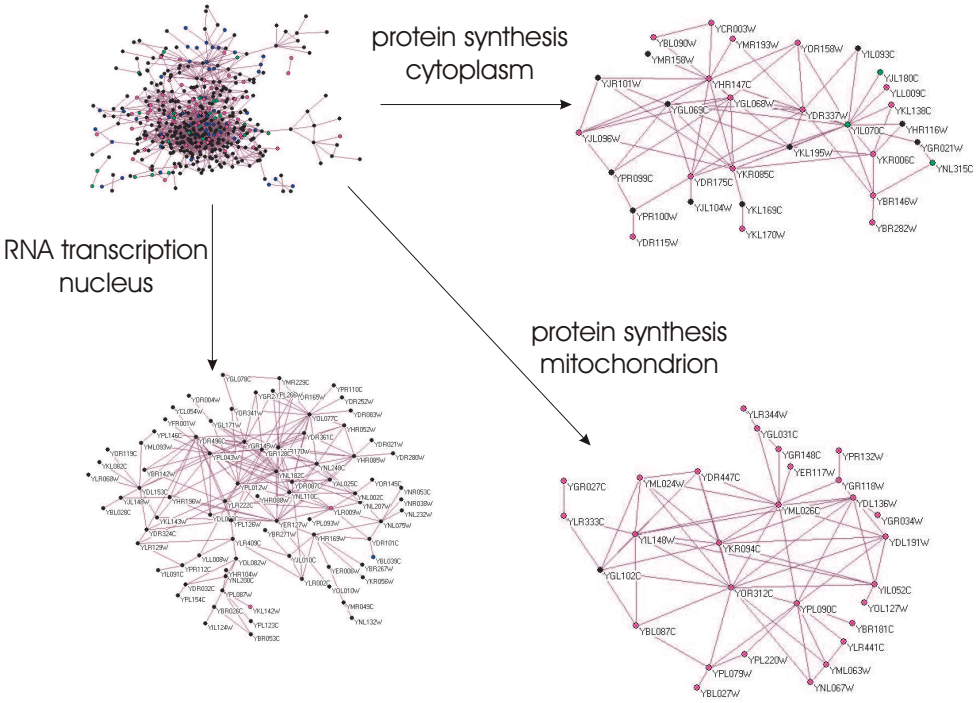

Figure 2.18: **Identification of clusters.** Hierarchical clustering based on mutual information as similarity measure is applied to a large-scale gene expression dataset [80]. Gene clusters are build for a set of exemplarily chosen parameters. The clusters can be characterised according the annotations of genes.

2.4.3 Analysis of informational fluxes in an integrative gene-metabolite network of sulfur stress response

Living organisms react to their changing environment and constitute complex systems of multiple informational fluxes interconnected in dense networks. Revealing such networks gives insight into the regulatory context of a biosystem and enables the definition of global models of biosystem functionality.

The regulations within a biosystem affect various levels, starting from the expression of genes to the production of proteins and metabolites. Towards the comprehensive description of the informational fluxes in a biosystem, the reconstruction of the underlying network needs to be based on data from different experimental techniques.

For this, we generate a gene-metabolite network of correlations reconstructed from an integrated dataset containing gene expression data for 6454 genes and metabolite data for 81 metabolites measured under the same experimental conditions [83]. *Arabidopsis thaliana* plants were grown under sulfur depletion conditions (see appendix) with the aim to unravel the informational flux response that is triggered by the depletion of sulfur as a major nutrient.

We are interested in the response of genes and metabolites to the sulfur stress conditions applied. Therefore we reduce in a first step the number of genes to those genes that show a high Pearson correlation to sulfur and sulfur-responding metabolites. Those metabolites are defined as sulfur-responding, which show significantly altered relative concentration levels in sulfur starved plants. Besides sulfur itself, these are glutathione, anthocyanins, allantoin, o-acetylserine, putrescine, raffinose, serine, shikimic acid, tryptophan, and uric acid. The comparison of these metabolites to all genes was repeated with randomised gene expression data to determine a threshold for significant metabolite-gene correlations. Only genes showing significant correlations to at least one of the sulfur-responding metabolites were used for further network reconstruction. The procedure is exemplarily shown for the sulfur-responding metabolite serine in Figure 2.19.

Based on this reduced dataset, containing the reduced number of genes and all

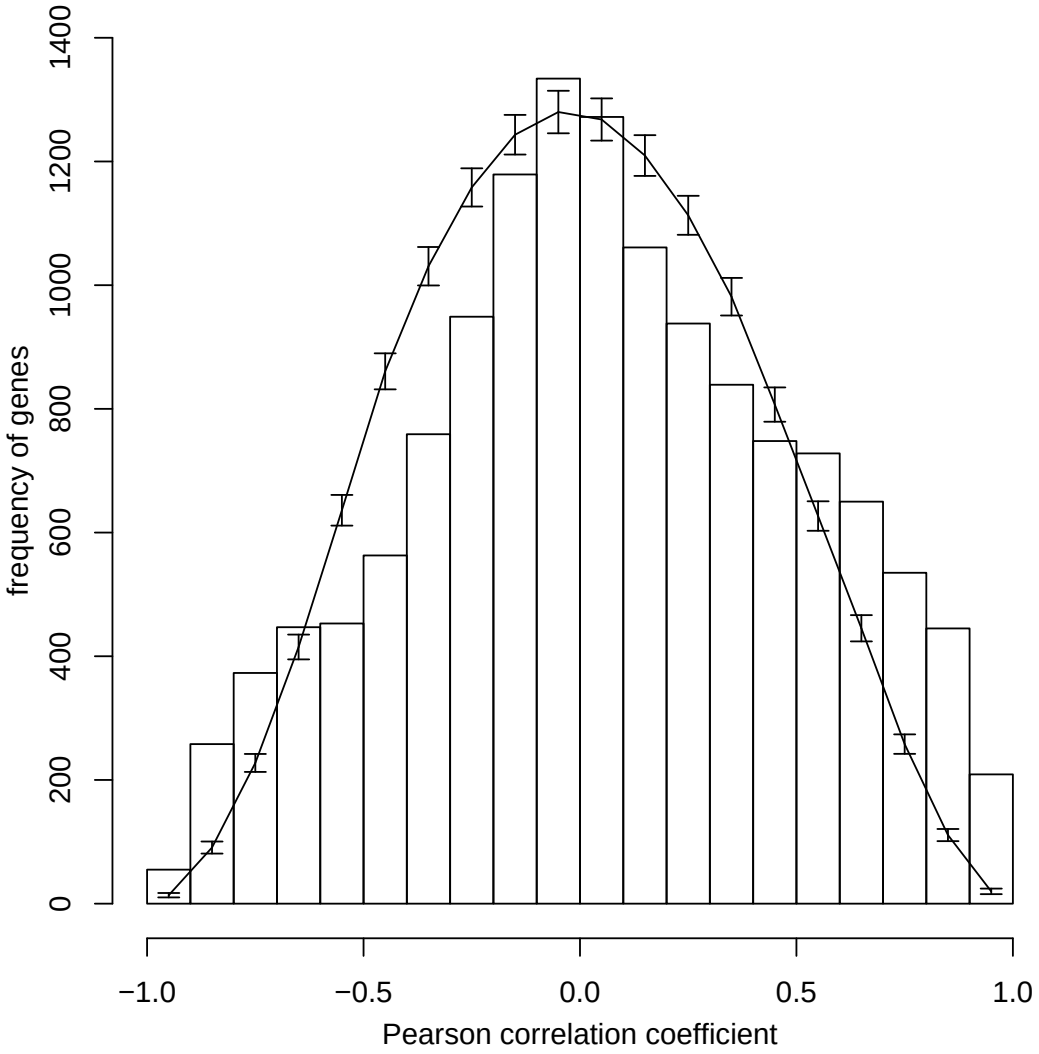

Figure 2.19: **Identification of sulfur correlated genes.** A subset off genes that show a significant Pearson correlation to sulfur and sulfur-related metabolites is determined as shown for the metabolite serine in this example. The histogram of Pearson correlation coefficients resulting from the comparison of serine to all genes is shown. The same comparisons to randomised gene expression data is shown as straight line. The error-bars denote the standard deviation obtained from 1000 realisations of shuffled gene expression data. Exclusively genes occurring in a histogram bin that contains at least twice as much associations from original data as from randomised data are chosen for further analysis. This corresponds to the first and the last two bins in the example shown.

metabolites, the pairwise comparison of all items is carried out. In addition to the Pearson correlation coefficient, we also calculate the mutual information for all pairs with the aim to exclude correlations arising from unevenly distributed data or outlying data points. A similar approach has been previously reported in literature [84]. For further analysis, both similarity measures were transformed to comply with the demands of a distance measure. For the Pearson correlation coefficient R, the transformed measure is obtained by

$$R_t = \sqrt{2\,(1 - abs(R))} \tag{2.25}$$

with $abs(R)$ denoting the absolute value of R. For the mutual information MI the transformation reads

$$MI_t = 1 - \frac{MI}{MI_{max}} \tag{2.26}$$

with MI_{max} denoting the maximum mutual information obtained from all pairwise comparisons.

The distance matrices obtained from the application of both measures, R_t and MI_t, are displayed in Figure 2.20 together with the results obtained from randomised realisations of the dataset. From visual inspection of Figure 2.20, we estimate an area of relevant associations containing only few associations obtained from shuffled data to $R_t < 0.45$ (corresponding to $R \geq 0.90$) and $MI_t < 0.30$. By this, a network is defined that contains 541 elements (genes and metabolites) with 5212 associations among them.

Network topology

The topology of a network can be regarded as a key aspect in the characterisation of the global network properties which can be directly predicted from its structure [85, 86, 87]. An important property of a network is the probability distribution of vertex connections $P(k)$, denoting the probability that a vertex interacts with k other vertices. The connectivities k of the network under consideration are distributed inhomogeniously with an average connectivity of $\bar{k} = 6.6$ (Figure2.21, top). Since the distribution of the connectivities follow a power law [88]

$$P(k) \propto k^{-\gamma} \tag{2.27}$$

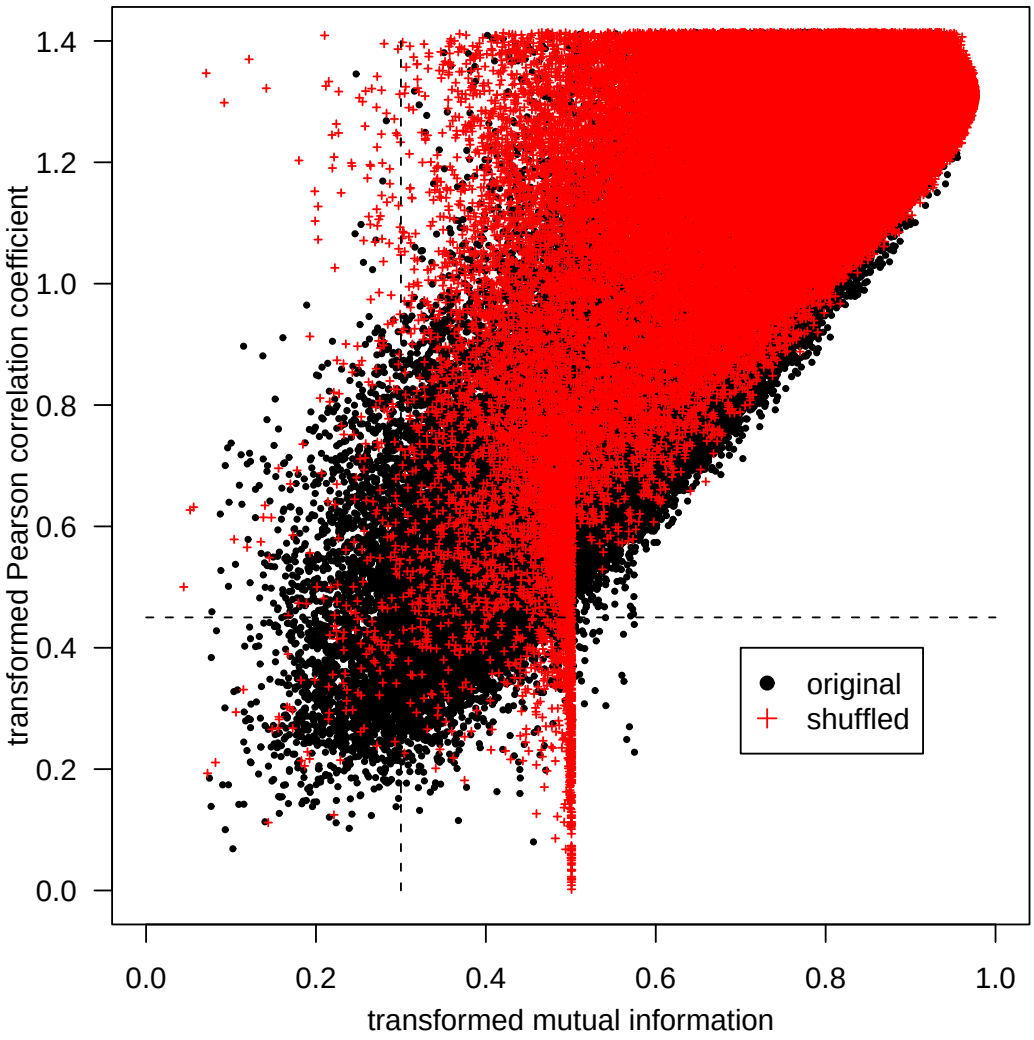

Figure 2.20: **Detection of relevant associations for network reconstruction.** For all pairwise comparisons of the reduced dataset, the Pearson correlation coefficient and the mutual information were calculated and displayed together with associations obtained from shuffled realisations of the dataset. Both similarity measures were transformed to fulfill the requirements of a distance measure with small values indication high similarities. Thresholds are defined by visual inspection (dashed lines) to minimise associations that are also shown from shuffled data. The peak around $MI_t \approx 0.5$ arises from gene/gene, metabolite/gene, or metabolite/metabolite associations containing outlying values. These associations would falsely be detected as high correlations using the Pearson correlation coefficient alone.

the parameter γ can determined to $\gamma \approx 2.3$ (Figure 2.21, bottom). It hereby fits into the range that has been observed in literature for other inhomogeniously-wired networks which are referred to as *scale-free* networks [88]. Such networks are characterised by a majority of nodes (genes or metabolites in the present study) showing just few connections to other nodes and few nodes showing a high connectivity to others. Networks showing this scale-free topology possess several universal characteristics like a high tolerance to errors due to high robustness. Nodes showing a high connectivity, however, are critically important for network stability and can be considered as putative controllers. On the other hand, they can also be regarded as sites of systems vulnerability.

Implementing a causal relationship

Both applied similarity measures, Pearson correlation coefficient and the mutual information, do not allow for the detection of the cause and the effect in the detected connection. For the network under consideration, this implies that the direction of informational flows between the elements is not defined. The experimental setup, however, is designed in a way that the primary cause of the systems excitement is *a priori* known: It arises from the depletion of sulfur in the surrounding medium. By this, we can assume that the changed sulfur levels can be regarded as the starting point of system excitation. For further analysis, sulfur is defined as the cause of excitation and all system responses to this excitation are regarded as effects. The reconstructed network (Figure 2.22) is based on the Pearson correlation coefficient as similarity measure. The mutual information is used as a 'filter' to remove associations arising from unevenly distributed data which correspond to outlying values.

Examples of biologically meaningful pathways

From the whole network, we extract in a first example routes of informational fluxes which are directed from sulfur towards sulfur-responding metabolites. We exclusively include connections (corresponding to edges in the graph) that link one of

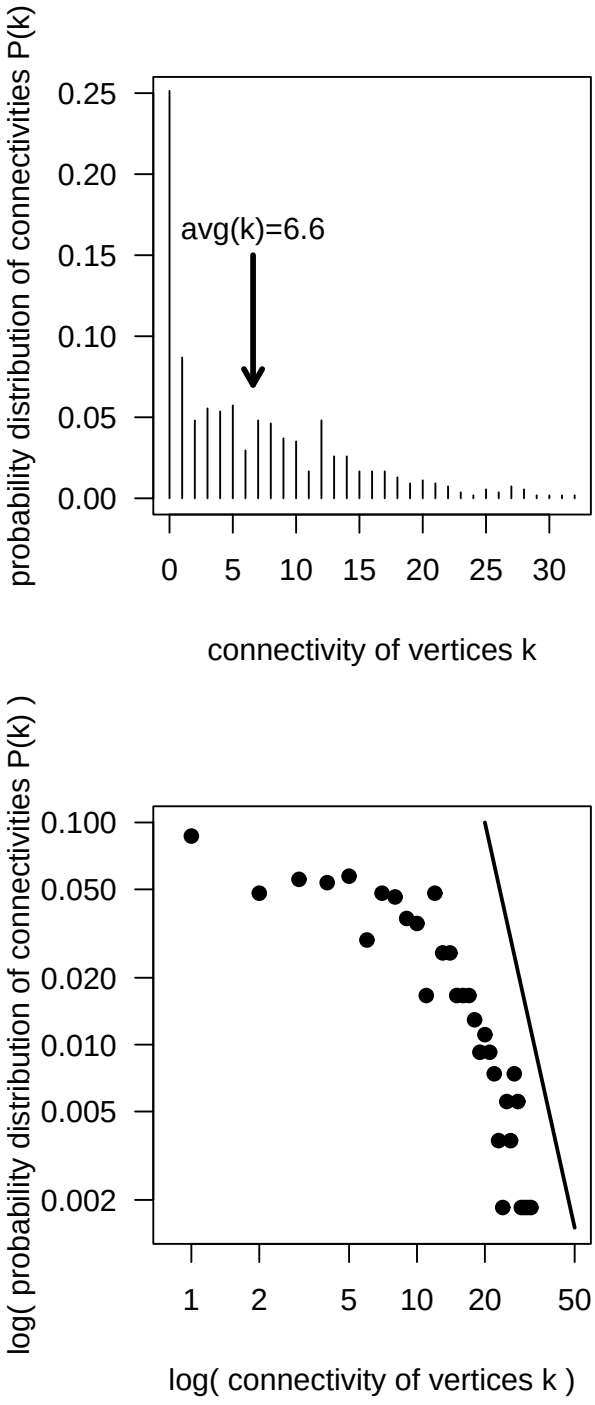

Figure 2.21: **Network properties.** The properties of a network can be predicted from its structure [85, 86, 87]. If the distribution of the connections per vertex $P(k)$ (top) follows a power law (Eq. 2.27), the topology of the network can be determined [88]. For the reconstructed network, the exponent of Eq. 2.27 is estimated by the slope of the linear right tail (bottom) to $\gamma \approx 2.3$.

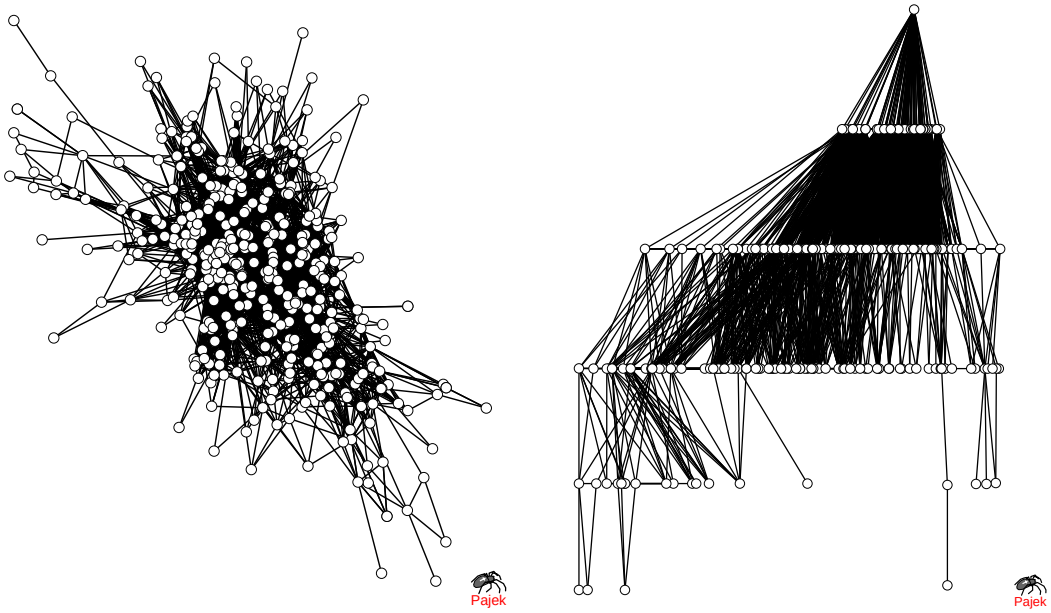

Figure 2.22: **Introduction of *cause-to-effect*.** The reconstructed network, based on the Pearson correlation coefficient as similarity measures, does not contain any causality (left). By taking the knowledge about the experimental conditions of sulfur depletion into account, sulfur can be regarded as starting point of initial system excitation. Thereby, sulfur can be placed in the center of the network representation (top node, right) and an *cause-to-effect* relationship can be introduced (from top to bottom, right).

the sulfur-responding metabolites to sulfur itself (directly or indirectly). From each of the sulfur-responding metabolites, we additionally include all connections that are directed downstream (in the sense of the informational flux), (Figure 2.23 A). A commonly observed response of plants to sulfur depletion is the accumulation of anthocyanins [83]. We find this confirmed in our reconstructed network by the position of the corresponding node (Figure 2.23, A). We observe only two edges for the node 'anthocyanins', both being upstream directed. This node can thereby be regarded as a physiological endpoint.

In a second example, we apply the same extraction to hormone-related metabolites and genes (Figure 2.23 B). Here, we detect a path to another well known physiological endpoint, the enhanced lateral root formation [83], represented by auxin related metabolites and genes. Starting from sulfur as the initial source of excitation, nitrilase 3 and the putative myrosinase-associated protein (At3g14210) are known to be involved in auxin biosynthesis. Downstream of the informational flux, we observe calmodulin 3, an auxin signal transduction factor, and the highly connected node IAA28, an auxin-related transcriptional factor which is mainly expressed in roots [89].

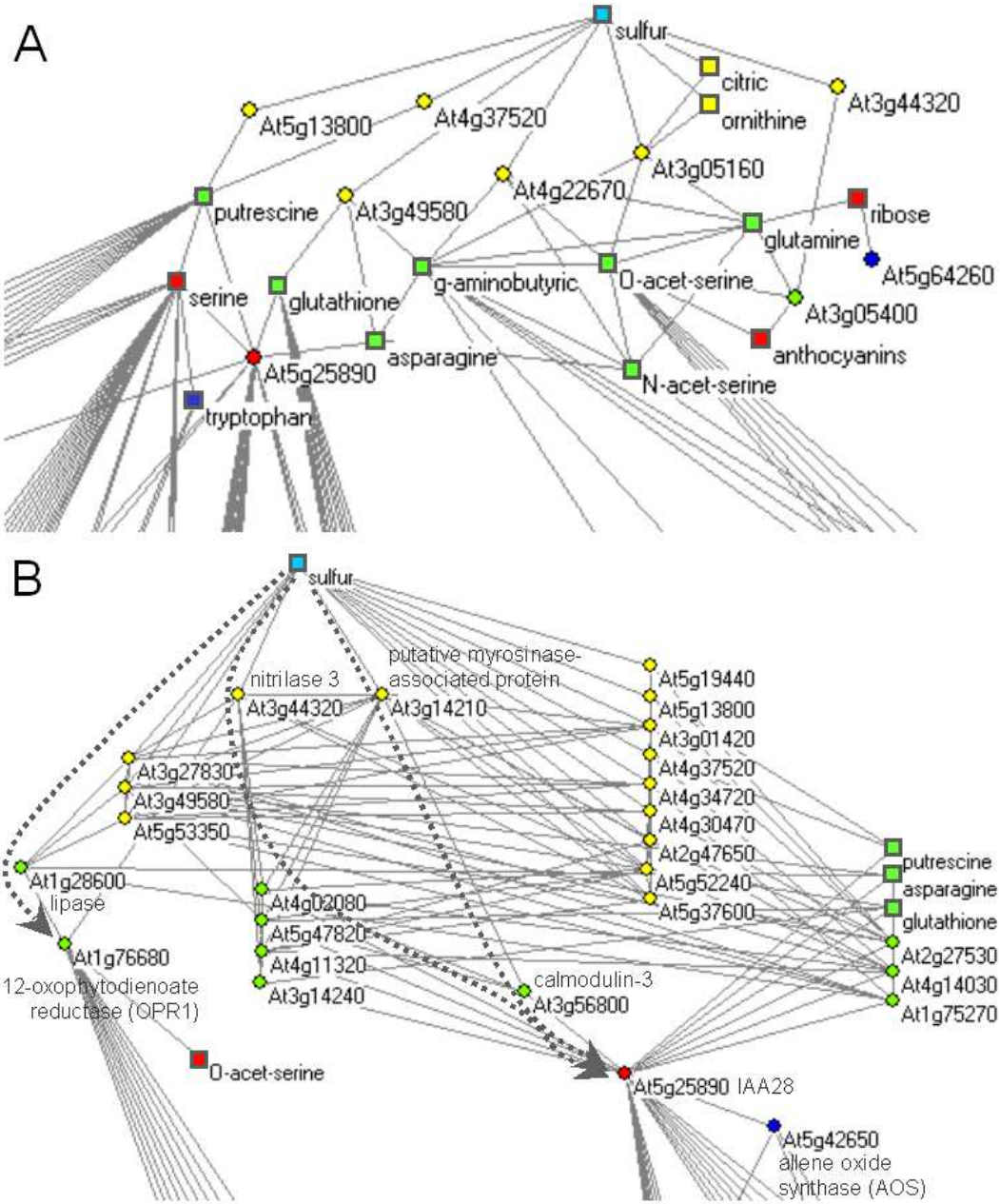

Figure 2.23: **Biologically meaningful network fragments.** From the network shown in Figure 2.22, biologically meaningful partitions referring to sulfur-responding metabolites (A, top) and hormone-related metabolites and genes (B, bottom) are extracted.

Chapter 3

Discussion

3.1 Mutual information

We introduced the concept of the mutual information and suggested its application as a measure of statistical independence between variables. In the context of the evaluation of complex biological networks, linear measures are commonly used on an *ad hoc* basis for the detection of correlations. From the biological point of view, also more general dependencies among biological measures, e.g. gene expression data or concentrations from protein or metabolite measurements, are supposable. We showed in an example based on artificial data that the mutual information extends these linear measures.

We continued with the definition of the mutual information in terms of discrete data and summarised some properties. This definition assumeed knowledge about the probability distribution of data. For the application to biological data, however, these distributions have to be estimation from continuous data. After introducing a simple *histogram approach* for this estimation, we presented and exemplified some of the known effects arising from this approach. In addition to the simple histogram binning, we summarised the estimation of the probability density on the basis of *kernel density estimators* (KDE) [59].

3.2 Fuzzy mutual information

After mentioning some of the drawbacks of the histogram binning, we proposed an extension of the bins to polynomial B-spline functions: Data points are no longer assigned to exactly one bin but to several bins simultaneously with a weight given by the B-spline functions. By definition, the weighting coefficients sum up to unity, which corresponds to an implicit normalisation. The algorithm is thereby reminiscent of kernel density estimators, it keeps the basic idea to associate data points to discrete bins. In contrast to KDE, the bins defined by B-spline functions are not placed at the positions of the data points but at the positions of the original rectangular bin. The spline order k determines the shape of the polynomial functions and specifies the number of non-zero coefficients. Due to the implicit normalisation, the probability to occupy a bin, $p(a_i)$, is obtained by the sum over all coefficients that belong to this bin, a_i. For KDE, the probability density has to be calculated by numerical integration methods which is a time consuming procedure. To clarify the application of the B-spline approach, we demonstrated the calculation of the mutual information for artificial example data in detail.

We showd that our approach improves the simple histogram binning method by comparing it to KDE. We provided a systematic comparison between these algorithms on the basis of artificial data with known distributions.

At first, we calculated the mutual information for uniformly distributed data of different sizes for which the 'true' value is known to be zero. In accordance with literature, we found for the simple histogram approach that the mutual information, as well as its standard deviation, scale linearly with the inverse size of the data. The same result holds for the estimation with B-spline functions, but with a 4-fold smaller slope for the mutual information as well as for its standard deviation. The KDE approach showed an asymptotic run with intermediate values for large datasets. For small datasets, the KDE approach gave even values slightly below the values obtained from the B-spline approach. In summary, the overestimation, which is depending on the size of the data, showed to be strongly reduced by the utilisation of B-spline functions for the estimation of mutual information compared

to the histogram binning and the KDE approach. Only for small datasets (≤ 100 values) the KDE approach gave slightly better results than the B-spline approach.

Then, we studied the influence of the parameters spline order k and number of bins M on the estimation of the mutual information. We generated an artificial dataset which was constructed in a way to represent a 'real' measurement on a biological system. We compared the different estimation algorithms in their ability to distinguish the given dataset from a dataset which was consistent with the null hypothesis of statistical independence. For this, we defined the significance S according to Eq. (2.23).

From the significances calculated from different spline orders it was observed that the null hypothesis was rejected for all spline orders. A nearly two-fold increase was shown for the step from $k = 1$ to $k = 2$, whereas for higher spline orders ($k \geq 3$) the significances stayed at the same level as for $k = 2$ and did not further improve the estimation for the distribution under consideration. Similar observations have been reported for KDE [69] where the particular choice of the kernel functions showed to be of minor influence for the result of estimation. In the given example, however, the calculations were carried out on the basis of one single dataset of a given size. The implications of the spline order on datasets of smaller sizes might be larger, since also overestimation effects have a larger influence. Further investigations with systematic evaluation based on data drawn from different distributions and different sizes need to be done.

The influence of the number of bins M on the significances was calculated for three estimators, $k = 1$, $k = 3$, and KDE for a range of bins $M = 2 \ldots 10$. A reasonable number of bins for the calculation of the mutual information was determined by the size of the data. The number of bins in the 2-D histogram used for the calculation of the joint entropy (see Figure 2.7) should be smaller than the number of data points. In a rough estimation it was suggested to use 3 to 10 times more data points than 2-D bins. This implies for the example dataset containing 300 data points, that a reasonable number of bins is in the range of $M = 6 \ldots 10$. Within this range it was observed that the significance was roughly doubled with the change of the spline order from $k = 1$ to $k = 3$. The discrimination of correlations from the hypothesis

of statistical independence was thereby significantly improved.

The KDE approach does not depend on the binning[1] and is thereby independent on the parameter M. The significance obtained from KDE lied in the range of the results obtained by $k = 3$. The runtime requirements of the KDE approach, however, were $\mathcal{O}(10^4)$ times higher than the ones of the B-spline approach. This limits the application of KDE to datasets of moderate sizes. For a large-scale gene expression dataset containing measurements for several thousand genes, the runtime exceeds a reasonable limit. Since the pairwise comparison of genes can be calculated in parallel, the utilisation of multi processor computers or computer clusters enables the processing of larger data matrices. Strategies for a simplification of the numerical integration step for KDE have been proposed [25]. The mutual information can be regarded as an average over a probability distribution and Eq. (2.13) can be written as

$$\hat{MI}(X,Y) = \left\langle \log \frac{\hat{f}(x,y)}{\hat{f}(x)\,\hat{f}(y)} \right\rangle \tag{3.1}$$

If the distribution is well represented by the dataset, the integration can be approximated by a summation over the data points

$$\hat{MI}(X,Y) = \frac{1}{N} \sum_{i=1}^{N} \log \left[\frac{\hat{f}(x_i,y_i)}{\hat{f}(x_i)\,\hat{f}(y_i)} \right] \tag{3.2}$$

This simplification has to be applied with caution since it assumes that all data points are independent realisations of the underlying distribution.

3.3 Software development

3.3.1 Calculating the mutual information — *mis_calc*

Within this work, we used the mutual information as a measure of similarity for gene expression data and metabolite concentration data. For this, we carried out a

[1]The numerical integration step in the KDE approach also bases on a binning procedure, but this binning is different from the binning discussed here.

comparative analysis between all genes and/or metabolites contained in the particular dataset under consideration. The number of compared attributes thereby lied in the range of up to several thousands and the number of pairwise comparisons up to several hundred millions.

To enable the application of the mutual information to large-scale datasets, as for example the datasets analysed in this work, we developed the software package *mis_calc*. It calculates the mutual information for the pairwise comparisons of the attributes of a data matrix. We incorporated the improvements described in this work and provided the possibility to calculate the mutual information on the basis of B-spline functions. By taking advantage of the independence of these comparisons among each other, the *mis_calc* software can be operated in a mode, where the comparisons can be calculated on different processors of one computer or even on different computers in parallel. Additionally, it was designed in a performance optimised way with a caching algorithm. For the underlying programming language, C++, compilers with effective optimisation routines are available.

Numerical programming environments, like Matlab and R, also allow for the calculation of the mutual information. They are very flexible and the utilisation of different estimation algorithms is relatively easy to realise. However, the run time requirements of these programming environments make the application to large datasets a very time consuming task. For datasets containing thousands of attributes, the runtimes exceed the limit for practical applicability. For the calculation of significance thresholds from shuffled data, this drawback becomes even more apparent.

From a programmers point of view, the *mis_calc* software package bases on a variety of self-made classes with different level of abstraction. The classes can be accessed via their application interfaces (API) and thereby can also be used for further software development. A description of the main classes is given in the appendix.

3.3.2 MetaGeneAlyse

Highly powerful techniques allow for the large-scale and parallel profiling of gene-expression and metabolite data from the same biological samples. The analysis of such integrated data enables the elucidation of different levels of cellular regulation and leads to new insights into genetic and physiological control.

We have developed *MetaGeneAlyse* [74], a web-based service for the analysis of integrated datasets containing gene-expression and metabolite data. *MetaGeneAlyse* incorporates a variety of normalisation algorithms that are essential for the preprocessing of data obtained from different measurement technologies. Various algorithms are available for data analysis and visualisation, enabling a researcher to easily obtain an overview over her or his data.

Several tools for the exploration of microarray data are freely available. They are usually offered as software tools for download and run on the users workstation. Our web-service, however, offers the possibility to run time-consuming calculations on the server-side. It is thereby possible to analyse even large data matrices that could not be processed with software running on the client-side.

3.4 Application on data

3.4.1 Global comparison of MI to Pearson correlation

Linear correlation measures are among the most frequently applied similarity measure in literature. They are often used on an ad hoc basis without discussion about their appropriateness in the given context. It thereby remains unclear if a considerable number of non-linear correlations might be missed. In our first application example we addressed the question whether previous analyses, based on linear correlation measures, sufficiently described the correlations within datasets or whether the mutual information is able to detect additional correlations that are not detected

by linear correlation measures, such as the Pearson correlation coefficient. The relation between the mutual information and the Pearson correlation can be explicitly given for Gaussian distributed data [79]. Both similarity measures were applied to two large-scale gene expression datasets [80, 78] with the intention to verify whether non-linear correlations, shown as deviations from the theoretical relation, can be detected.

At first it was confirmed that the Pearson correlation results in values in the range $[-1, 1]$ whereas the mutual information showed exclusively positive values. It also has to be noted that the theoretical relation shown in both applications is not corrected for the systematic overestimation of the mutual information. A correction would result in a horizontal shift of the theoretical prediction towards higher mutual information values. For our analysis we neglected this correction since we were interested in qualitative deviations.

Our observations showed for the first dataset, that it was fairly well described by the theoretical relation of the mutual information and the Pearson correlation. In particular, we did not detect gene-gene association (each association corresponded to one data point in Figure 2.13) with low Pearson correlation and high mutual information which would be missed by solely using the linear Pearson correlation. Other deviations from the theoretical relation were observed for low mutual information and high Pearson correlation values. Such associations arised from gene-gene comparisons containing outlying values as was exemplarily shown in Figure 2.14, A. The mutual information assigned low similarity values to these comparisons since the corresponding marginal entropies already resulted in low values. A similar approach has been reported [84] were the marginal entropies of genes were used as a filter to remove genes containing outlying values.

We concluded, that previous analyses on the first dataset under consideration based on linear correlation measures [80] did not miss any non-linear associations. This presents an important finding since it is by all means supposable that the regulations inherent in the genetic network under consideration show a more complex behaviour than the observed linear correlations. Even for one of the largest gene expression dataset at hand, insufficient data might complicate the detection of such

complex patterns of regulation. Alternatively, the biological mechanism underlying the regulatory networks might not lead to non-linear correlations. Experimental methods for probe and target preparation and measurement techniques also might complicate the detection of non-linear associations.

The second dataset also showed to follow the theoretical relation in principle. Again, outlying values with high Pearson correlations and low mutual information were detected. In contrast to the first dataset, this dataset contained tuples arising from low Pearson correlations and high mutual information. Detailed gene-gene plots for two interesting tuples are depicted in Figure 2.14 C and D. From these examples can be seen, that the data points for two genes form compact clusters. Without attempting to draw conclusions about the biological context of such clusters here, it might be reasonable to check whether this clustering arises from the choice of the oligonucleotide probe in the probe set.

Even though the examples shown in Figure 2.14 C and D are easily classified by visual inspection, the amount of pairwise comparisons from large-scale gene expression experiments are in an order of magnitude where comprehensive visual inspection is no longer applicable.

3.4.2 Application of MI to gene expression dataset

After the global comparison of the mutual information to the linear Pearson correlation, we used the mutual information as a basis for a hierarchical clustering analysis. The calculation of the mutual information, as well as the application of clustering procedures, require the choice of adequate parameters. In this example we did not attempt to carry out a comprehensive analysis with a variety of sets of parameters but showed the exemplary application for one set of parameters.

From our particular set of parameters starting with our choice of a dataset, the number of bins and the spline order for the calculation of the mutual information, the clustering algorithm we applied, the threshold we used for the building of clusters and the database we used for evaluating the cluster results, we obtained a biologically

interesting result. We were able to identify three large gene cluster where each of the clusters contained genes that were annotated to just a few of the available annotation classes. Even more interestingly, two of the clusters separated genes, that take part in the protein synthesis, according to their sub-cellular localisation, the cytoplasm and the mitochondrion. So far not annotated genes within one of these clusters showed sequence similarity to similar annotated genes in other organisms.

We also addressed the question whether similar result can be obtained by the utilisation of the Pearson correlation coefficient as similarity measure (data not shown). The similarity matrix, containing the pairwise comparisons of all genes of the dataset under consideration, is not directly comparable to the similarity matrix based on the mutual information. For this, we carried out the same characterisation of cluster results on the basis of the Pearson correlation coefficient as similarity measure, as is done in Figure 2.15 (data not shown). By this, we were not able to detect any clusters that were comparable to the clusters we found by the utilisation of the mutual information as similarity measure.

The utilisation of different parameters for the calculation of the mutual information and different clustering algorithms might lead to other complementary results. A comprehensive testing of parameters and the evaluation of their results were not in the scope of this work but could present a direction for future work.

3.4.3 Analysis of informational fluxes in an integrative gene-metabolite network of sulfur stress response

Plant growth and development are dependent on a variety of biotic and abiotic factors. Since plants are immobile, they have evolved a variety of physiological mechanisms to attain the nutrients they need. For situations of limited nutrient supply, plants are forced to respond with adaptive strategies to assure their survival. Under the limitation of the essential nutrient sulfur, *Arabidopsis thaliana* plants respond to this stress condition [83]. The reaction involves the sensing of the depletion, transduction of the perceived signal to control points of regulation,

and the triggering and silencing of gene expression and thereby the production of proteins and metabolites.

Our approach aimed at the revealing of informational fluxes that are triggered by the excitation of sulfur depletion on the genetic, as well as on the metabolic level. We intended to verify, whether well known stress response changes in expression levels and in the metabolic content of *Arabidopsis thaliana* can be interconnected by an integrated analysis. We extended a previous study [83] in which the reaction to sulfur depletion was measured on the gene expression level. For this, we additionally measured metabolite concentrations of probes that were sampled under the same experimental conditions as for the expression data.

The strategy of our analysis based on the assumption, that the sulfur-depletion conditions constitute an external excitation. This excitation is a trigger for the plant and the excitation propagates through the signalling network. Therefore, we focusd our analysis on potentially relevant genes that respond to the sulfur starvation conditions: for a chosen set of sulfur-responding metabolites we neglected all genes that did not show correlations to at least one of these sulfur-responding metabolites. Here, we used the linear Pearson correlation coefficient as a measure of correlation. This subset of genes, together with all measured metabolites, build the bases for further analysis.

In principle, the reconstruction of the signalling network based on the Pearson correlation coefficient as similarity measure. To exclude false positive correlations arising from unevenly distributed data, we additionally calculated the mutual information for all comparisons of genes and metabolites and compared the results obtained from both similarity measures (Figure 2.20). We thereby took our results from one of the previous examples (section 2.4.1) into account where we detected situations with outlying values. By this, we used the mutual information as a filter for false positive correlations. The significance of correlations was estimated by visual inspection from the superimposed comparison with similarities calculated from randomised datasets. An area, defined by thresholds for the Pearson correlation coefficient and for the mutual information, containing just few correlations arising from randomised data was chosen (Figure 2.20).

One important aspect in the characterisation of a network is its topology, from which general properties of the system can be predicted [85, 86, 87]. The topology of the resulting network was characterised as *scale-free* which confirms the findings for other biological networks [88]. Due to the high redundancy of connections within such networks, local errors or changes rarely lead to the loss of the global information carrying ability of the networks and result in robustness. Highly connected nodes, however, are critically important for network stability. In the network under consideration, sulfur represents such a highly connected node and thereby supports our approach that the excitation of the plant by sulfur depletion triggers an informational stress response flux.

With focus on the sulfur-responding metabolites, we were able to verify the already described accumulation of anthocyanins in *Arabidopsis thaliana* under sulfur stress conditions [83]. The node 'anthocyanins' in the reconstructed network (Figure 2.23, A) showed only two edges, both were positioned between sulfur and anthocyanins and though were directed upstream in the sense of the informational flux. Among all metabolites, this was the only one that did not possess any link to parallel or downstream nodes. Due to its position in the graph, we considered the node 'anthocyanins' as physiological endpoint in the path of informational flux leading to the accumulation of anthocyanins in the plant.

By focusing on hormone-related network elements, another highly connected node was identified, the transcriptional factor IAA28 (Figure 2.23, B). Starting from sulfur, several interlacing redundant paths passed nitrilase 3 and a putative myrosinase-associated protein, both involved in auxin biosynthesis and then lead via the auxin signal transduction factor calmodulin 3 to the auxin regulated transcriptional factor IAA28, which is known to be mainly expressed in roots. Analysis of IAA28 expression history in the Stanford Microarray Database[2] provided us the experimental evidence, that IAA28 is involved in auxin signalling. Auxin is known to control latent root formation and the latent root formation is known to be triggered by sulfur starvation. By this, the detected auxin path extended previous observations regarding the transcriptional factor IAA28, which acted as a highly connected 'hub' in the reconstructed network.

[2]http://genome-www.stanford.edu/microarray

Summary

The advent of high-throughput technologies opens new perspectives for the understanding of living organisms on a molecular level. The variety of complementary data produced by these technologies provides the challenge of multifunctionality and implies the presence of regulatory networks as opposed to isolated linear pathways of causality. In a *systems biology* approach, the evaluation of an organism as a whole and the understanding of underlying regulatory networks is addressed.

Towards the understanding of these networks, data from different experimental sources needs to be integrated. Such sources range from gene expression data, to data from protein and metabolite concentration measurements. The application of data preprocessing steps is necessary to enable the integration and the combined analysis of such data.

The evaluation of complex regulatory networks underlying molecular processes poses a major challenge to current research. A commonly used approach is the clustering of features on the basis of a similarity measure. In this context, the choice of an adequate similarity measure, as well as the choice of the clustering method itself is crucial for the results obtained. Information theoretic concepts, such as the mutual information, were used to extend conventional similarity measures. Mutual information presents a general measure of statistical independence and is thereby able to detect any type of functional relationships, extending the potentialities of commonly used linear measures.

Since the mutual information is defined for discrete data, its application to con-

tinuous biological data requires adaptive procedures. In this work, we present an algorithm for estimating the mutual information from continuous data and compare our approach to previously existing algorithms.

The clustering of a large-scale dataset demands the pairwise comparisons among all features of the dataset. For a number of features, in the order of thousands, the number of comparisons becomes very large. To enable the application of the mutual information to such datasets, we developed a performance optimised software package that also incorporates the algorithm presented in this work.

We have designed and implemented *MetaGeneAlyse*, a web-based service that allows the upload of data and its analysis with various methods including the presented algorithm for the calculation of the mutual information. Due to the implementation of adequate normalisation routines, *MetaGeneAlyse* enables the processing of data that is measured with different experimental techniques and thereby complies with the requirement of a systems biological approach. Since all calculations are done on the server-side, even large analyses can be carried out independently of the technical prerequisites of the researcher.

The application to publicly available datasets shows, that the mutual information detects dependencies that are missed by linear correlation measures. Furthermore, biologically relevant clusters were revealed that, again, could not be found by the utilisation of linear measures. For an integrated in-house dataset containing gene expression data and metabolite concentration data for stress response experiments, the reconstructed signalling network showed agreements with physiological findings.

Zusammenfassung

Die Entwicklung von Hochdurchsatzmesstechnologien eröffnet neue Perspektiven für das Verständnis von lebenden Organismen auf einem molekularen Niveau. Die Vielseitigkeit der komplementären Daten, die durch solche Technologien hervorgebracht werden, eröffnen neue Herausforderungen. Sie implizieren die Existenz von übergreifenden regulativen Netzwerken, im Gegensatz zu isolierten linearen Pfaden von Ursächlichkeiten. Im *systembiologischen* Ansatz wird die Betrachtung eines Organismus als Ganzes und das Verständnis des zugrunde liegenden regulativen Netzwerkes betrachtet.

Für das Verständnis solcher Netzwerke müssen aus unterschiedlichen Meßtechnologien stammende Daten integriert werden. Als Beispiele können Genexpressionsdaten, und Daten von Protein- und Metabolitkonzentrationsmessungen angeführt werden. Für die Integration und Auswertung dieser unterschiedlichen Daten werden bestimmte, die Daten vorverarbeitende Schritte, angewendet.

Die Auswertung von komplexen regulativen Netzwerken, die molekularen Prozessen zugrunde liegen, stellen eine Herausforderung für die aktuelle Forschung dar. Ein üblicher Ansatz für die Rekonstruktion von Netzwerken besteht dabei in der Gruppierung der Daten. Die Grundlage solcher Gruppierungen bilden Ähnlichkeitsmaße zwischen den zu gruppierenden Objekten des Datensatzes. Dabei haben sowohl die Auswahl eines geeigneten Ähnlichkeitsmaßes, als auch die Auswahl eines Gruppierungsalgorithmus selbst, einen großen Einfluß auf das Ergebnis der Gruppierung. In diesem Zusammenhang wird das informationstheoretische Konzept der wechselseitigen Information verwendet. Die wechselseitige Information stellt ein allgemeines

Maß für statistische Unabhängigkeit dar und erweitert das Potential von häufig verwendeten linearen Ähnlichkeitsmaßen.

Die wechselseitige Information ist ursprünglich für die Anwendung auf diskrete Daten definiert worden. Für ihre Anwendung auf kontinuierliche biologische Daten muß das Konzept angepasst werden. In der vorliegenden Arbeit stellen wir einen Algorithmus für die Anwendung der wechselseitigen Information auf kontinuierliche Daten vor und vergleichen diesen mit bereits existierenden Algorithmen.

Der Gruppierung von Objekten eines Datensatzes liegt der paarweise Vergleich zwischen allen diesen Objekten zugrunde. Für große Datensätze mit tausenden von Objekten wird die Anzahl der zu vergleichenden Paare sehr groß. Um die wechselseitige Information als Grundlage für die Gruppierung von Objekten verwenden zu können, haben wir ein geschwindigkeitsoptimiertes Programm erstellt, das den in dieser Arbeit vorgestellten Algorithmus beinhaltet.

Wir haben den web-basierten Dienst *MetaGeneAlyse* konzipiert und programmiert. Er ermöglicht das Hochladen und Auswerten von Daten mit verschiedenen Methoden, auch mit der in dieser Arbeit vorgestellten Methode. Durch die Möglichkeit der Anwendung verschiedener Normalisierungsmethoden auf die Daten können aus unterschiedlichen Meßtechnologien stammende Daten ausgewertet werden. Dies entspricht der Idee des systembiologischen Ansatzes. Bei der Benutzung von *MetaGeneAlyse* werden alle Berechnungen auf einem Servercomputer durchgeführt. Dies ermöglicht die Auswertung von großen Datensätzen, unabhängig von den technischen Voraussetzungen des jeweiligen Benutzers.

Die Anwendung der wechselseitigen Information als Ähnlichkeitsmaß auf öffentlich zugängliche biologische Daten zeigt Abhängigkeiten, die bei der Verwendung von linearen Maßen übersehen worden wären. Desweiteren finden wir biologisch relevante Gruppierungen, die wir ebenfalls bei der Verwendung von linearen Maßen nicht finden. Für einen im Institut erstellten integrierten Datensatz, der sowohl Genexpressions- als auch Metabolitkonzentrationsdaten für Stressreaktionsexperimente enthält, werden für das rekonstruierte Signalnetzwerk Übereinstimmungen mit physiologischen Erkenntnissen festgestellt.

Appendix

Software development

Calculating mutual information — *mis_calc*

Implementation details

The *mis_calc* software package was developed under the programming language C++. In the next subsections, we give an overview of the main classes and their methods use by *mis_calc*.

class SignalSpace *SignalSpace* represents an object which primarily implements the basic algorithms for storing and manipulating a two dimensional representation of the data. In addition to this, it offers methods to calculate single or pairwise entropies, mutual information and other statistical values (e.g. Pearson correlation coefficient). Since the calculation of the mutual information *MI* benefits heavily when storing preliminary results, *SignalSpace* provides a transparent caching mechanism for such values.

Basic Concepts of class *SignalSpace* After instantiation of a *SignalSpace* object, it can be used to hold signal vectors. A signal vector is a one dimensional

sequence of values which are regarded as measurements made for the same feature (gene, protein, etc.) under different conditions. It can also be associated with a unique name (e.g. Accession number). Signal vectors are represented by the inner class *SignalSpace::vector* which will be discussed in the next subsection. Most of the numerical operations applied to the *SignalSpace* are implemented by calling the appropriate methods on the *SignalSpace::vector* object.

Before working with a *SignalSpace*, the matrix has to be filled with data to be analysed. After calling the *open* method, which sets the *SignalSpace* into its initial state, one might add single vectors to the *SignalSpace* by applying the *addRow* method or by reading a complete data set from a *std::istream* via a call to *read*. Each signal vector may consist of variable length and may also contain undefined values (e.g. in cases where the measurement failed).

After filling the *SignalSpace* with signal vectors, one has to call the *close* method before performing any further calculations on the data. The *close* method declares the data set to be complete and triggers the calculation of those values which can be cached.

On a closed *SignalSpace* one might perform *MI* calculations by calling the *mi* method or retrieve other correlation factors like *pearson* and *entropy*.

We did not want to be restricted to only pairwise calculation of statistical values, but wanted the design to allow the solution of higher dimensional problems. Therefore, we decided to implement a generic addressing of vector sets. To most of the statistical methods implemented by *SignalSpace*, indices are passed as an $std :: vector < int >$ from the standard template library *STL*. In this way, the method $SignalSpace::entropy(const\ vector < int > \&)$ works on two vectors for which the entropies are calculated on the basis of the same entries, or calculates a solution for three or more vectors.

Entropies and Mutual Informations within class *SignalSpace* Entropies are calculated based on the described algorithm in which the range of a signal vector

is divided into an arbitrary number M of equally sized intervals. The algorithm uses a B-spline weight function, as implemented by the class *Spline* to calculate the probability of finding a given signal inside a given interval. by this, it is possible that a single signal can be found in a number of intervals with a different probability.

Beside the number of intervals M one has to provide the spline weight order k which has to be in the range $1 \leq k \leq M - 1$. The larger k, the larger the number of intervals for which the probability is not 0. If $k = 1$, the modified method behaves similar to the simple binning algorithm for MI calculation. M and k can be set by calls to the methods *setIntervals* and *setSplineWeight* respectively. Calculation of the MI can afterwards be performed by calling the *mi* method with an array of indices identifying the vectors for which to calculate the mutual information.

Since the mutual information can be defined as $MI(A, B) = H(A) + H(B) - H(A, B)$, one can cache the single vector entropies $H(A)$ and $H(B)$ beforehand in order to perform this time consuming step once for all vectors held by *SignalSpace*. The same holds true for the distribution vectors which only depend on the individual signal vectors and the parameters M and k. These are needed for the calculation of single entropies, as well as for the joined entropy $H(A, B)$. The caching mechanism is implemented by filling up a vector with single vector entropies for all signal vectors and by initialising a matrix with probability values for each of the signal vectors. The so called distribution space gets filled whenever the *SignalSpace* gets closed and holds valid data.

This caching strategy performs best if all vectors of the *SignalSpace* have the same length and do not contain too many undefined values (*nil*) since the cached values for single entropies might become invalid if vectors with undefined values have to be compared. If, for example, we deal with two vectors A and B, $A = \{0.9, 1.2, nil, 3.8, 9.5\}$ and $B = \{1.8, nil, 4.2, 9.3, 8.2\}$ we might only work on those three indices for which both vectors hold valid entries and the previously cached single vector entropies $H(A)$ and $H(B)$ are invalid since they are based on 4 valid entries for each vector.

Example The following example code reads in a set of measurement from standard input, sets $M = 10$ and $k = 3$, and calculates the joined entropy of two vectors.

```
// create a new SignalSpace with M=10 intervals and spline order k=3

SignalSpace ss;

ss.setIntervals(10);
ss.setSplineWeight(3);

// initialise SignalSpace before adding new rows
ss.open();

// Read in the whole signale space from standard input
ss.read(std::cin)

// close the signalspace. This initialises
// the internal caching mechanisms

ss.close();

// create an array describing two dimensions in vector space
std:: vector<int> v(2);

// we ask for the indices 55 and 28 of the vector space, blindly
// assuming that they actually exist
v[0]=55;
v[1]=28;

// calculate the joined entropy of dimension 55 & 28
// and dump it

std::cout << "Joined Entropy of vector pair "
          << "(" << v[0] << "; " << v[1] << ") "
          << "is " << ss.entropy(v)
          << std::endl;
```

class SignalSpace::vector Signal vectors are implemented by the inner class *SignalSpace::vector* which represents a specialised indirect offspring of the *std* :: *vector* $< C >$ template class found within the *STL*. Beside the methods inherited from this super class (e.g. iteration, assignment, size management), *SignalSpace::vector* supports the concept of undefined values (*nil*) and implements various mathematical

and statistical functions. For the implementation and proper handling of *nil* values, *SignalSpace::vector* holds the variables in an container object (*SignalSpace::Signal*) which for most applications is opaque when accessing the *SignalSpace::vector*.

The *SignalSpace::Signal* object implements the basic arithmetic operations and other function (e.g. output operators) one would like to apply to its contents (*double*) representing the signals within the *SignalSpace*.

Example The example code creates two instances of *SignalSpace::vector* (*sv1,sv2*) and fills them up with 0 or *nil* values. After this, we iterate through *sv1* using an index and assign random numbers to the *Signal* objects. Additional objects are added via the *push_back* method inherited from $std :: vector < C >$. We then perform a vector multiplication and dump out the result, this time using an *iterator* in order to traverse through the vector.

```
// create a new SignalSpace::vector holding 100 entries
// implicitely filled with 0 values.
SignalSpace::vector sv1(100);

// create a second SignalSpace::vector holding 100 entries
// explicitely filled with nil Signals
SignalSpace::vector sv2(100,SignalSpace::Signal::nil);

// iterate through vector #1 and fill it with random numbers
for(int i=0;i<sv1.size();i++) {
  sv1[i] = (double)::rand() / (double)RAND_MAX;
}

// push a nil value on to the end of the vector
sv1.push_back(SignalSpace::Signal::nil);

// push a Signal of value 5 on sv2
sv2.push_back(5);

// ask if entry #99 is currently undefined
if(sv1[99].isNil()) {
  std::cerr << "entry #99 is undefined" << std::endl;
}
```

```
 // Multiply each component of s1 with the corresponding component
 // from s2. This throws an exception if the vectors do not have
 // the same size.
 // It could also be written s1 *= s2. Furthermore we
 // provide + - / as well as /= += and -=.

 s1 = s1*s2;

 // using an iterator
 SignalSpace::vector::iterator i;

 for(i=sv1.begin();i!=sv1.end();i++) {
  std::cerr << *i << std::endl;
 }
}
```

class SignalSpace::Signal The container class *SignalSpace::Signal* holds a *double* value and implements all necessary arithmetical operators one might want to apply on its contents. It furthermore defines a special value *nil* which may also be assigned to *SignalSpace::Signal* and invalidates the contents, flagging it as *not computable*. Trying to compute with *nil* values always results in an *SignalSpace::Signal* object representing *nil*. Comparison with such values or casting to the contents class is not allowed and will result in an exception. To avoid such failures *SignalSpace::Signal* provides the method *isNil* which enables applications to check the state of the variable.

Example The following example code creates three different *SignalSpace::Signal* objects initialising them with either *nil* or 0. Afterwards, we perform a direct assignment, multiplication and various comparisons.

```
// create a new Signal which is currently nil
SignalSpace::Signal s1;

// create a second signal with explicit value 0
SignalSpace::Signal s2=0;
```

```
// create a third one with explicitely setting
// the value to nil (undefined)
SignalSpace::Signal s3=SignalSpace::Signal::nil;

// change value of s1 to 5;
s1=5;

// Multiply to Signal objects. This would trigger
// an exception if one operand would be nil
s1 = s1 * s2;

try {
 // test s1 against s2
 if(s1<s2) {
  std::cerr << "s1 (" << s1 << ") is smaller than s2 (" << s2 << ")"
            << std::endl;
 } else {
  std::cerr << "s2 " << s2 << ") is smaller than s1 (" << s1 << ")"
            << std::endl;
 }

 // do the same for s1 and s3
 // This will throw an exception since comparison with
 // nil is not allowed

 if(s1<s3) {
  std::cerr << "s1 is smaller than s3" << endl;
 } else {
  std::cerr << "s3 is smaller than s1" << endl;
 }
} catch (logic_error & le) {
 std::cerr << "got an exception " << le.what() << std::endl;
}
```

class Spline The *Spline* class serves as a helper for the *SignalSpace* and provides the B-Spline weight function in the form of the *Weight* method. The standard creator of *Spline* takes the two arguments M and k representing the number of intervals and spline weight order. After instantiation, a call to *Weight(int n,double*

t) might be used to retrieve the value of the B-Spline weight function for interval M and parametric value t, which in our case represents a single *Signal* within a *SignalSpace::vector* remapped into the interval $0 \leq t \leq n - k + 1$.

class MiFactory With the *SignalSpace class*, one already is able to perform MI calculation on a set of signal vectors. In a minimal implementation, a *SignalSpace* can be filled up with data, as shown in the previous example. Afterwards, we only need to address the unique vector pairs within the *SignalSpace*, i.e. the upper right triangle of a $l \times l$ matrix, where l denotes the number of *SignalSpace::vector*s held within the *SignalSpace*, and call the *mi* method in order to obtain MI values for the whole *SignalSpace*. This could be implemented in the form of two simple iterations:

```
SignalSpace ss;

// Fill up the SignalSpace as shown in the previous example
....

// Iterate through all rows
for(int i=0;i<ss.Rows()-1;i++) {
 // Iterator through all rows > i
 for(int j=i+1;i<ss.Rows();i++) {
   std::cout << "MI (" << ss.getRowName(i) << "," << ss.getRowName(j)
     << " happens to be " << ss.mi(i,j) << std::endl:
   }
}
```

However, since the calculation of each MI value is numerically independent of all the others and very time consuming, it is desirable to implement a multi-threaded version of *mis_calc* which takes advantage of a multi processor environment. For this purpose, we implemented the class *MiFactory* which as an offspring of *SignalSpace* provides all the previously described functionality and adds methods for resource locking and distributed computing.

The standard constructor *MiFactory(int w=1)* takes a single argument which defines the maximal number of threads assigned to the computation. After using methods

inherited from *SignalSpace* to fill up the matrix, one has to call the *run* method in order to calculate *MI* values for all vector pairs. Progress and error states can afterwards be monitored with the *getState* method. If the *MiFactory* turns *idle*, results might be retrieved with the *getMi* function.

The *run* method divides the upper triangle of the resulting matrix into w portions of equal size and feeds these subsets of *SignalSpace*s as parameters to *MiFactory::Thread* objects which implement code that can be executed concurrently within the *POSIX threads*[3] environment.

Example A multi-threaded version of the previous example might look like:

```
// create a new factory with a maximum of 8
// threads running in parallel
MiFactory mf(8);

// Define intervals and spline order
mf.setIntervals(10);
mf.setSplineWeight(3);

// Stuff in data from file './bla'
mf.open():
ifstream ifi("/bla");
mf.read(ifi);
ifi.close()
mf.close();

// start the job; We select calculation of the
// pearson correlation coeff. and the mutual information

mf.start(MiFactory::calc_pearson|MiFactory::calc_mi);

// periodically check if the factory came to a hold
for(;;) {
 // do something usefull in between
 ....
```

[3]http://www.iso.org

```
   if(mf.getStatus()==MiFactory::finished ||
    mf.getStatus()==MiFactory::error) {
    break;
   }
   // sleep 5 seconds waiting for the next iteration
   ::sleep(5);
  }

  if(mf.getState()==MiFactory::error) {
    std::cerr << "Something fishy happened here" << std::endl;
    ::exit(1);
  }

 // retrieve the desired results

 for(int i=0;i<mf.Rows();i++)    {
  for(int j=i+1;mf.Rows();j++) {
   std::cout << "MI(" << mf.getRowName(i) << "," << mf.getRowName(j)
             << ") "  << " happens to be " << mf.mi(i,j) << std::endl;
  }
 }
```

class MiFarm Another concept of distributed computing is the usage of a number of independent computers (i.e. clusters) working on the solution. This architecture is also very interesting for the given problem since after an initial distribution of the input set, no further communication except minimal progress and status reports and the final result set is needed. Although the result sets might be rather huge, this still represents only a moderate overhead in terms of network communication. We therefore implemented a preliminary third version which uses the *CORBA* API in order to implement distributed computing across the network.

The object representation is implemented in the class *MiFarm*, which is a subclass of *MiFactory*. Just as *MiFactory*, the *MiFarm* object has to be started with the *run* method in order to calculate the MI values for the underlying *SignalSpace*. *MiFarm* also divides the solution space into equally sized portions and distributes the problem to a set of slave threads running on the same machine. Instead of computing the MI, however, each thread takes a set of slave computers and transmits the involved

SignalSpace::vector objects via the *CORBA* communication protocol. After starting the *mis_calc* algorithm on the remote machines, each slave thread tracks progress and error states and reports back to the *MiFarm* object.

From an applications point of view the API did not change at all. The fact that the program is running on different computers is totally hidden from the program. And the *MI* calculation might be implemented in the very same way as shown in the previous example with the only exception that one has to instantiate an object of type *MiFarm* instead of *MiFactory*.

class MisCorba::Factory, MisCorba::Client and MisCorba::Server Within *MiFarm*, the details of cluster communication are implemented in three classes *MisCorba::Client*, *MisCorba::Factory* and *MisCorba::Server* which are only used from within methods of the *MisFarm* object in order to hide the *CORBA* interface from higher level functions.

MisCorba::Factory implements the stubs of a *CORBA* interface as defined in the *CORBA* specific *IDL* language. In principle this represents a collection of methods mirroring the methods of *MiFactory* where the arguments have to be transformed to satisfy the needs of a *CORBA* compliant application. *MisCorba::Client* also mirrors methods of the *MiFactory* object on the client side encapsulating the transport layer of the *CORBA* API. Each program that likes to interface with a remote *MisCorba::Factory* should use an instantiation of *MisCorba::Client* which almost can be used like a local *MiFactory* object. The *MisCorba::Client* tries to make connection to the remote *MisCorba::Factory* and translates arguments and results in a way that hides the middle-ware layer of *CORBA*.

Finally, the *MisCorba::Server* is dispatching *MisCorba::Factory* instances to clients, trying to start a job on a remote machine. Each computer, that is intended as a slave node within the cluster, needs one instance of the *mis_client-server* program running which mainly implements the *MisCorba::Server* object and advertises the existence of such a service via the *CORBA* name service.

MetaGeneAlyse

Implementation details

MetaGeneAlyse runs on a multiprocessor Linux server[4] under the web-server Apache[5] with mod-perl enabled. It primarily consists of a compilation of Perl[6] scripts for the interaction with the user. The interaction of the Perl scripts with the web-server is mainly realised with the perl CGI module[7]. The methods of the CGI module handle actions such as the HTML-forms for the data uploads and the passing of arguments to other Perl scripts. For some analysis and visualisation steps, the statistics software package R[8] is internally used. Fast analysis steps are processed on-the-fly, whereas computationally expensive steps, like the calculation of a large distance matrix, are handled by a job queuing system. For this, run-time optimised C++ programs are utilised. Several log files inform about all user actions, error messages send to users, and the status of the job queue.

Freely available tools for data exploration which run under Java on the user-side have been reported [75, 76]. An advantage of MetaGeneAlyse is the calculation of large distance matrices that gets computationally expensive and requires a large working memory for larger datasets, such datasets could not be analyzed with software running on client-side workstations.

[4]Fujitsu Siemens N800, 8 processors, 6 GB RAM
[5]http://httpd.apache.org
[6]http://www.perl.org
[7]http://stein.cshl.org/WWW/software/CGI
[8]http://www.r-project.org

Data examples

Yeast gene expression dataset by Hughes et al.

The first dataset used in the global comparison of the Pearson correlation to the mutual information (section 2.4.1) was obtained from the supplemental material of the publication [80]. A two-color cDNA hybridisation assay was used to generate 300 gene expression profiles for *Saccharomyces cerevisiae*. Transcript levels of a mutant or compound-treated culture where compared to that of a wild-type or mock-treated culture. The 300 profilings contain 276 deletition mutants, 11 tetracycline-regulatable alleles of essential genes, and treatments with 13 well-characterised compounds.

All experiments were performed under a single condition to allow for direct comparison of the behaviour of all genes in response to all mutations and treatments.

Our analyses were carried out on the basis of the logarithms of ratios between the treated probes and the wild-type or mock-treated cultures.

Human cancer dataset by He et al.

A dataset containing large-scale gene expression data from human cancer tissues [78] was used as the second dataset in the global comparison of the Pearson correlation to the mutual information (section 2.4.1). Hybridisations on 204 in-situ synthesised oligonucleotide (60mers) arrays were performed with two-color cDNA samples derived from 20 different human tissues and cell lines. The design of the $\sim$ 24k oligonucleotides reports $\sim$ 2500 known genes.

Arabidopsis thaliana sulfur depletion dataset by Nikiforova et al.

Total RNA and hydrophilic metabolites from the same samples were obtained from transcript profiles by array hybridisations on nylon membranes [83] and metabolite profiles by gas chromatography - mass spectrometry (GC-MS). The dataset contains relative transcript amounts for 6454 non-redundant genes and relative concentrations for 81 non-redundant chemical compounds. For conditions of constitutive and induced sulfur starvation, data for 8 experimental time points was measured referring to a time course of 6 to 13 days.

Software used

The following software packages were used for the analyses reported in this work.

- Apache web-server (Version1.3.26 under Unix with mod_perl/1.27): Open source web-server program. In the context of the MetaGeneAlyse analysis tool, which is described in this work, the Apache web-server processes Perl scripts to dynamically generate HTML web pages. It is publicly available[9].
- Matlab (Version 6.5.0): Interpreted numerical programming environment. The MathWorks, Inc., MA, USA
- MetaGeneAlyse (Version 1.3): Web-based tool for the analysis of gene expression and/or metabolite data[10]. This tool was developed by Carsten O. Daub and described in detail in this thesis.
- mis_calc (Version 0.63): Software package for the calculation of mutual information. It was developed by Sebastian Kloska in cooperation with Carsten O. Daub.

[9] http://httpd.apache.org
[10] http://metagenealyse.mpimp-golm.mpg.de

- Octave (Version 2.1.44): Interpreted numerical programming environment, similar to Matlab. It is publicly available[11].
- Pajek (Version): Software for analysis and visualisation of large networks. It is publicly available[12].
- Perl (Version 5.6.1 to 5.8.0, **P**ractical **E**xtraction and **R**eport **L**anguage): Perl is a language optimised for scanning arbitrary text files, extracting information from those text files, and printing reports based on that information. Perl was initially developed by Larry Wall and is publicly available[13].
- R (Version 1.5.1 to 1.8.1): Language and environment for statistical computing and graphics. It is publicly available[14].

[11] http://www.octave.org
[12] http://vlado.fmf.uni-lj.si/pub/networks/pajek
[13] http://www.perl.com
[14] http://www.r-project.org

Bibliography

[1] L. von Bertalanffy (1969) *General Systems Theory, Foundations, Development, Applications*. George Braziller, New York/NY

[2] L. Hood, T. Ideker, and T. Galitski (2001) A new approach to decoding life: Systems biology. *Annual Review of Genomics and Human Genetics* **2**, 343-372

[3] H. Kitano (2002) Systems biology: A brief overview. *Science* **295**, 1662-1664

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

[5] M. Schena, D. Shalon, R. W. Davis, and P. O. Brown (1995) Quantitative Monitoring of Gene Expression Patterns with a Complementary DNA Microarray. *Science* **270**, 467-470

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

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

[8] S. Braxton, T. Bedilion (1998) The integration of microarray information in the drug development process. *Curr. Opin. Biotechnol.* **9**, 643-649

[9] C. Debouck, P. N. Goodfellow (1999) DNA microarrays in drug discovery and development. *Nat. Genet.* **21**, 48-50

[10] S. F. Dobrowolski, R. A. Banas, E. W. Naylor, T. Powdrill, and D. Thakkar (1999) DNA microarray technology for neonatal screening. *Acta Paediatr. Suppl.* **88**, 61-64

[11] O. Fiehn, J. Kopka, P. Dörmann, T. Altmann, R. N. Trethewey, and L. Willmitzer (2000) Metabolite profiling for plant functional genomics. *Nat. Biotechnol.* **18**, 1157-1161

[12] W. Weckwerth (2003) Metabolomics in systems biology. *Annu. Rev. Plant Biol.* **54**, 669-689

[13] J. Lill (2003) Proteomics tools for quantitation by mass spectrometry *Mass Spectr. Rev.* **22**, 182-194

[14] O. Fiehn, and W. Weckwerth (2003) Deciphering metabolic networks. *Eur. J. Biochem.* **270**, 579-588

[15] R. L. Somorjai, B. Dolenko, and R. Baumgartner (2003) Class prediction and discovery using gene microarray and proteomics mass spectroscopy data: curses, caveats, cautions. *Bioinformatics* **19**, 1484-1491

[16] S. G. Hilsenbeck, W. E. Friedrichs, R. Schiff, P. O'Conell, R. K. Hansen, C. K. Osborne, and S. A. Fuqua (1999) Statistical analysis of array expression data as applied to the problem of tamoxifen resistance. *J. Natl. Cancer Inst.* **91**, 453-459

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

[18] J. Taylor, R. D. King, T. Altmann, and O. Fiehn (2002) Application of plant metabolomics to plant genotype discrimination using statistics and machine learning. *Bioinformatics* **18**, 241S-248S

[19] J. MacQueen (1967) Some methods for classification and analysis of multivariate observation. In L. M. Le Cam, and J. Nyeman (eds), *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, Vol. I, University of California Press.

[20] S. Tavazoie, J. D. Hughes, M. J. Campbell, R. J. Cho, and G. M. Church (1999) Systematic determination of genetic network architecture. *Nature Genet.* **22**, 218-285

[21] Y. Cheng, and G. M. Church (2000) Biclustering of expression data. *Proc. Int. Conf. Intell. Syst. Mol. Biol.* **8**, 93-103

[22] A. Tanay, R. Sharan, and R. Shamir (2002) Discovering statistically significant biclusters in gene expression data. *Bioinformatics* **18**, S136-S144

[23] L. Hunter, R. C. Taylor, S. M. Leach, and R. Simon (2001) GEST: a gene expression search tool based on a novel Bayesian similarity metric. *Bioinformatics* **17**, S115-S122

[24] A. J. Butte, and I. S. Kohane (2000) Mutual information relevance networks: functional genomic clustering using pairwise entropy measurements. *Pac. Symp. Biocomp.* **5**, 427-439

[25] R. Steuer, J. Kurths, C. O. Daub, J. Weise, and J. Selbig (2002) The mutual information: detecting end evaluating dependencies between variables. *Bioinformatics* **18**, 231S-240S

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

[27] W. Weckwerth, V. Tolstikov, and O. Fiehn (2001) Metabolomic characterisation of transgene potato plants using GC/TOF and LC/MS. In *Proceedings of the 49th ASMS Conference on Mass Spectrometry and Allied Topics.* pp. 1-2

[28] F. Kose, W. Weckwerth, T. Linke, and O. Fiehn (2001) Visualizing plant metabolomic correlation networks using clique-metabolite matrices. *Bioinformatics* **17**, 1198-1208

[29] W. Weckwerth, and O. Fiehn (2002) Can we discover novel pathways using metabolite analysis? *Curr. Opin. Biotechnol.* **13**, 156-160

[30] R. Steuer, J. Kurths, O. Fiehn, and W. Weckwerth (2003) Observing and interpreting correlations in metabolomic networks. *Bioinformatics* **19**, 1019-1026

[31] K. Y. Yeung, W. L. Ruzzo (2001) Principal component analysis for clustering gene expression data. *Bioinformatics* **17**, 763-774

[32] O. Alter, P. O. Brown, and D. Botstein (2000) Singular value decomposition for genome-wide expression data processing and modeling. *Proc. Natl. Acad. Sci. USA* **97**, 10101-10106

[33] N. S. Holter, M. Mitra, A. Maritan, M. Cieplak, J. R. Banavar, and N. V. Fedoroff (2000) Fundamental patterns underlying gene expression profiles: Simplicity from complexity. *Proc. Natl. Acad. Sci. USA* **97**, 8409-8414

[34] Y. Yamanishi, J.-P. Vert, A. Nakaya, and M. Kanehisa (2003) Extraction of correlated gene clusters from multiple genomic data by generalized kernel canonical correlation analysis. *Bioinformatics* **19**, 323i-330i

[35] W. Liebermann (2002) Linear modes of gene expression determined by independent component analysis. *Bioinformatics* **18**, 51-60

[36] M. Scholz, S. Gatzek, O. Fiehn, and J. Selbig (2003) Metabolomics: Detecting biological parameters by independent component analysis. *submitted to Bioinformatics*

[37] M. P. S. Brow, W. N. Grundy, D. Lin, N. Cristianini, C. W. Sugnet, and T. S. Furey (2000) Knowledge-based analysis of microarray gene expression data by using support vector machines. *Proc. Natl. Acad. Sci. USA* **97**, 262-267

[38] T. S. Furey, N. Cristianini, N. Duffy, D. W. Bednarski, M. Schummer, and D. Haussler (2000) Support vector machine classification and validation of cancer tissue samples using microarray expression data. *Bioinformatics* **16**, 906-914

[39] L. A. Soinov, M. A. Krestyaninova, and A. Brazma (2003) Towards reconstruction of gene networks from expression data by supervised learning. *Genome Biology* **4**:R6

[40] J. Khan, J. S. Wei, M. Rigner, L. H. Saal, M. Ladanyi, F. Westermann, F. Berthold, M. Schwab, C. R. Antonescu, C. Peterson, and P. Meltzer (2001) Classification and diagnostic prediction of cancer using gene expression profiling and artificial neural networks. *Nat. Med.* **7**, 673-679

[41] C. H. Yeang, S. Ramaswamy, P. Tamayo, S. Mukherjee, R. M. Rifkin, M. Angelo, M. Reich, E. Lander, J. Mesirov, and T. Golub (2001) Molecular classification of multiple tumor types. *Bioinformatics* **17**, S316-S322

[42] P. Tamayo, D. Solnim, J. Mesirov, Q. Zhu, S. Kitareewan, E. Dimitrovsky, E. S. Lander, and T. R. Golub (1999) Interpreting patterns of gene expression with self-organizing maps: methods and application to hematopoietic differentiation. *Proc. Natl. Acad. Sci. USA* **96**, 2907-2912

[43] A. P. Dempster, N. M. Laird, and D. B. Rubin (1977) Maximum likelihood from incomplete data via the EM algorithm. *Nature Genet.* **22**, 218-285

[44] J. C. Bezdek (1981) Pattern recognition with fuzzy objective function algorithms. Plenum Press, New York.

[45] E. Mjolsness, R. Castaño, and A. Gray (1999) Multi-parent clustering algorithms for large-scale gene expression analysis. *Technical report JPL-ICTR-99-5* Jet Propulsion Laboratory Section 367. http://www-aig.jpl.nasa.gov/public/mls/papers/emj/multiparentPreprint.pdf

[46] P. Cheesemann, and J. Stutz (1996) Bayesian classification (autoclass): theory and results. In U. M. Fayad, G. Piatetsky-Shapiro, P. Smyth, R. Uthurusamy (eds), *Advances in Knowledge Discovery and Data Mining* AAAI Press/MIT Press, http://ic-www.arc.nasa.gov/ic/projects/bayes-group/images/kdd-95.ps

[47] X. Wen, S. Fuhrman, G. S. Michaels, D. B. Carr, S. Smith, J. L. Baker, and R. Somogyi (1998) Large-scale temporal gene expression mapping of central nervous system development. *Proc. Natl. Acad. Sci. USA* **95**, 334-339

[48] A. Ben-Dor, R. Shamir, Z. Yakhini (1999) Clustering gene expression patterns. *J. Comput. Biol.* **6**, 281-297

[49] T. Hastie, R. Tibshirani, M. B. Eisen, A. Alizadeh, R. Levi, L. Staudt, W. C. Chan, D. Botstein, and P. O. Brown (2000) 'Gene shaving' as a method for identifying distinct sets of genes with similar expression profiles. *Genome Biol.* 1:research0003.1-0003.21

[50] P. D'haesleer, S. Liang, and R. Somogyi (2000) Genetic network inference: from co-expression clustering to reverse engineering. *Bioinformatics* **16**, 707-726

[51] D. R. Bickel (2003) Robust cluster analysis of microarray gene expression data with the number of clusters determined biologically. *Bioinformatics* **19**, 818-824

[52] P. D'haesleer, X. Weng, S. Fuhrman, and R. Somogyi (1997) Information processing in cells and tissues. *Plenum*, 203-212

[53] G. S. Michaels, D. B. Carr, M. Askenazi, S. Fuhrmann, X. Wen, and R. Somogyi (1998) Cluster analysis and data visualization of large-scale gene expression data. *Pac. Symp. Biocomp.* **3**, 42-53

[54] R. Herwig, A. O. Schmitt, M. Steinfath, J. O'Brien, H. Seidel, S. Meier-Ewert, H. Lehrach, and U. Radelof (2000) Large-scale clustering of cDNA-fingerprinting data. *Geneome Res.* **9**, 1093-1105

[55] Y. Moon, B. Rajagopalan, and U. Lall (1995) Estimation of mutual information using kernel density estimators. *Phys. Rev. E* **52**, 2318-2321

[56] B. T. Korber, R. M. Farber, D. H. Wolpert, and A. S. Lapedes (1993) Covariation of mutations in the V3 loop of human immunodeficiency virus type 1 envelope protein: An information theoretic analysis. *Proc. Natl. Acad. Sci. USA* **90**, 7176-7180

[57] J. Gorodkin, L. J. Heyer, S. Brunak, G. D. Stormo, X. Wen, and R. Somogyi (1997) Display the information contents of structural RNA alignments: the structure logos. *Comput. Appl. Biosci.* **13**, 583-586

[58] S. Liang, S. Fuhrman, and R. Somogyi (1998) Reveal, a general reverse engineering algorithm for inference of genetic network architectures. *Pac. Symp. Biocomp.* **3**, 18-29

[59] A. M. Fraser, H. L. Swinney (1986) Independent coordinates for strange attractors from mutual information. *Phys. Rev. A* **33**, 2318-2321

[60] P. Thénevaz, and M. Unser (2000) Optimization of mutual information for multiresolution image registration. *IEEE Trans. Image Processing* **9**, 2083-2099

[61] D. P. Ellis, and J. A. Bilmes (2000) Using mutual information to design feature combinations. *Proc. ICSLP-2000*

[62] C. O. Daub, R. Steuer, J. Selbig, and S. Kloska (2003) Estimating mutual information using B-spline functions — an improved similarity measure for analysing gene expression data. *submitted to BMC Bioinformatics*

[63] C. E. Shannon (1948) A mathematical theory of communication. *The Bell System Technical Journal* **27**, 623-656

[64] A. N. Kolmogorov (1968) Logical basis for information theory and probability theory. *IEEE Trans. Information Theor.* **14**, 662-664

[65] H. Herzel, A. O. Schmidt, and W. Ebeling (1994) Finite sample effects in sequence analysis. *Chaos, Solitons and Fractals* **4**, 97-113

[66] H. Herzel and I. Grosse (1995) Measuring correlations in symbol sequences. *Physica A* **216**, 518-542

[67] I. Grosse (1996) Estimating entropies from finite samples. In J. A. Freund (ed.) *Dynamik, Evolution, Strukturen* Dr. Köster, Berlin

[68] M. S. Roulston (1999) Estimating the error on measured entropy and mutual information. *Physica D* **125**, 285-294

[69] B. W. Silverman (1986) Density estimation for statistics and data analysis. *Chapman and Hall, London*

[70] C. DeBoor (1978) A practical guide to splines. *Springer, New York*

[71] H. Herzel and I. Grosse (1997) Correlations in DNA sequences: The role of protein coding segments. *Phy. Rev. E* **55**, 800-810

[72] T. Schreiber, and A. Schmitz (2000) Surrogate time series. *Phyica D* **142**, 346-382

[73] J. Theiler, S. Eubank, A. Longtin, B. Galdrikian, and J. D. Farmer (1992) Testing for nonlinearity in time series: the method of surrogate data. *Phyica D* **58**, 77-94

[74] C. O. Daub, S. Kloska, and J. Selbig (2003) MetaGeneAlyse: analysis of integrated transcriptional and metabolite data. *Bioinformatics* **19**, 2332-2333

[75] B. Dysvik, and I. Jonassen (2001) J-express: exploring gene expression data using Java. *Bioinformatics* **17**, 369-370

[76] A. Sturn, J. Quackenbush, and Z. Trajanowski (2002) Genesis: cluster analysis of microarray data. *Bioinformatics* **18**, 207-208

[77] G. T. Klus, A. Song, A. Schick, M. Wahde, and Z. Szallasi (2001) Mutual Information Analysis as a Tool to Assess the Role of Aneuploidy in the Generation of Cancer-Associated Differential Gene Expression Patterns. *Pac. Symp. Biocomp.* **6**, 42-51

[78] Y. D. He, H. Dai, E. E. Schadt, G. Cavet, S. W. Edwards, S. B. Stepaniants, S. Duenwald, R. Kleinhanz, A. R. Jones, D. D. Shoemaker, and R. B. Stoughton (2003) Microarray standard data set and figures of merit for comparing data processing methods and experiment design. *Bioinformatics* **19**, 956-965

[79] R. Steuer, C. O. Daub, J. Selbig, and J. Kurths (2003) Measuring distances between variables by mutual information. *accepted by Proceedings of 27th Annual Conference of GfKl*

[80] T. R. Hughes, M. J. Marton, A. R. Jones, C. J. Roberts, R. Stoughton, C. D. Armour, H. A. Bennett, E. Coffey, H. Dai, Y. D. He, M. J. Kidd, A. M. King, M. R. Meyer, D. Slade, P. Y. Lum, S. B. Stepaniants, D. D. Shoemaker, D. Gachotte, K. Chakraburtty, J. Simon, M. Bard, and S. H. Friend (2000) Functional Discovery via a Compendium of Expression Profiles. *Pac. Symp. Biocomp.* **6**, 42-51

[81] A. Brazma, J. Vilo (2000) Gene expression data analysis. *FEBS Lett.* **480**, 17-24

[82] S. F. Altschul, T. L. Madden, A. A. Schäffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman (1997) Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res.* **25**, 3389-3402

[83] V. Nikiforova, J. Freitag, S. Kempa, M. Adamik, H. Hesse, and R. Höfgen (2003) Transcriptome analysis of sulfur depletion in Arabidopsis thaliana: interlacing of biosynthetic pathways provides response specificity. *Plant J.* **33**, 633-650

[84] A. J. Butte, T. Tamayo, D. Solnim, T. R. Golub, and I. S. Kohane (2000) Discovering functional relationships between RNA expression and chemotherapeutic susceptibility using relevance networks. *Proc. Natl. Acad. Sci. USA* **97**, 12182-12186

[85] R. Albert, H. Jeong, and A. L. Barabasi (2000) Error and attack tolerance of complex networks. *Narure* **406**, 378-382

[86] H. Jeong, B. Tombor, R. Albert, Z. N. Oltval, and A. L. Barabasi (2000) The large-scale organization of metabolic networks. *Narure* **407**, 651-654

[87] J. Stelling, S. Klamt, K. Bettenbrock, S. Schuster, E. D. Gilles (2002) Metabolic network structure determines key aspects of functionality and regulation. *Narure* **420**, 190-193

[88] A. L. Barabasi, and R. Albert (1999) Emergence of Scaling in Random Networks. *Science* **286**, 509-512

[89] L. E. Rogg, J. Lasswell, and B. Bartel (2001) A Gain-of-Function Mutation in IAA28 Suppresses Lateral Root Development. *Plant Cell* **13**, 465-480