Systematic Interpretation of High-Throughput Biological Data

Kurt Fellenberg
Ruhr-Universität Bochum
Germany

1. Introduction

MATLAB has evolved from the command-line-based ``MATrix LABoratory" into a fully-featured programming environment. But is it really practical for implementing a larger software package? Also if it is intended to run on servers and if Unix is preferred as a server operation system? What if there are more problem-related statistical methods available in R? Positive answers to these and more questions are shown in example discussing the ``Multi-Conditional Hybridization Processing System" (M-CHiPS). Here, as well, the name is not entirely descriptive because apart from the classical microarray hybridizations it takes data from e.g. antibody array incubations as well as methylation or quantitative tandem mass spectrometry data by now. The system was implemented predominantly in MATLAB. It currently contains more than 13,000 hybridizations, incubations, gels, runs etc. comprising all common microarray transcriptomics platforms but also genomic chip data, chip-based methylation data, 2D-DIGE gels, antibody arrays (both single and dual-channel), and TMT 6-plex MS/MS data. Apart from tumor biopsies, it contains also data about model organisms, e.g. *Trypansosoma brucei*, *Candida albicans*, and *Aspergillus fumigates*, to date 11 organisms in total.
While data stemming from e. g. Microarray and Mass Spectrometry platforms need very different preprocessing steps prior to data interpretation, the result can generally be regarded as a table with its columns representing some biological conditions, e.g. various genotypes, growth conditions or tumor stages, just to give some examples. Also, in most cases, each row roughly represents a "gene", more precisely standing for its DNA sequence, methylation status, RNA transcript abundance, or protein level. Thus, quantitative data stemming from different platforms and representing the status of either the transcriptome, methylome or the proteome can be collected in the very same format (database structure, MATLAB variables). Also, the same set of algorithms can be applied for analysis and visualization.
However, the patterns comprised by these large genes × conditions data tables cannot be understood without additional information. The behaviours of some ten thousands of genes need to be explained by Gene Ontology terms or transcription factor binding sites. And often hundreds of samples need to be related to represented genotypes, growth conditions or disease states in order to interpret these data. In addition to the signal intensities, M-CHiPS records information about the protocols involved (to track down systematic errors), sample biology and clinical data. Risk parameters such as alcohol consumption and

smoking habit are stored along with e.g. tumor stage and grade, cytogenetical aberrations, and lymphnode invasion, just to provide few examples. These additional data can be of arbitrary level of detail, depending on the field of research. For tumor biopsies, recently 119 such clinical factors plus 155 technical factors are accounted for, just to give one example. All these data are acquired and stored in a statistically accessible format and integrated into exploratory data analysis. Thus, the expression patterns are related to (and interpreted by means of) the biological and/or clinical data.
Thus the presented approach integrates heterogenous data. But not only are the data heterogenous. The high-throughput data as well as the additional information are stored in a data warehouse currently providing an analysis platform for more than 80 participants (www.m-chips.org) of different opinions about how they want to analyze their data. In subsection 4.2.3, the chapter will contrast providing a large multitude of possible algorithms to choose from to common view and use as a communication platform and user friendliness in general. As a platform for scientists written by scientists, it equally serves the interests of the programmers to code their methods quickly in the programming language that best suits their needs (4.2.4). Apart from MATLAB, M-CHiPS uses R, C, Perl, Java, and SQL providing the best environment for fast implementation of each task. The chapter discusses further advantages of such heterogeneity, such as combining the wealth of microarray statistics available in R and Bioconductor, with systems biology tools prevalently coded in MATLAB (4.1.4). It also discusses problems such as difficult installation and distribution as well as possible solutions (distribution as virtual machines, 4.2.4).
The last part of the chapter (section 5) is dedicated to what can be learned from such biological high-throughput data by inferring gene regulatory networks.

2. High-throughput biological data

Bioinformatics is a relatively new field. It started out with the need for interpreting accumulating amounts of sequence data. Thus the analysis of gene and/or protein sequences is what one may call ``classical'' bioinformatics. While sequence analysis still provides ample opportunity for scientific research, it is nowadays only one out of many bioinformatics subfields. Structure prediction attempts to delineate three-dimensional structures of proteins from their sequences. Microscopic and other biological or clinical (i.e. computer tomographical) images are used to model cellular or physiological processes. And quantitative, so called ``omics'' data record the status of many to all genes of an organism in one measurement. The status of a gene can be measured on different regulatory levels, corresponding to different processes involved in gene expression. While **genomics** refers to the abundance and the sequence of all genes, **epigenomics** data record e.g. the genes' degree of methylation (determining if a gene can be transcribed or not). Transcription of a gene means copying its information (stored as DNA sequence in the nucleus of the cell) into a data medium (much like a DVD or other media) that can leave the cell nucleus. This medium transports the information into the surrounding cytoplasm (where the hereby encoded protein is produced). It is called "messenger RNA" or "transcript". Transcript levels are reflected by (quantitative) **transcriptomics** data. Presence of the transcript is a prerequisite for producing the encoded protein in a process called translation. However, regulatory mechanisms governing this process as well as different decay rates both for different transcripts and for different proteins interfere with a direct proportional relationship of transcript and protein levels in most cases. Protein levels (i.e. the actual results of gene expression) are recorded by **proteomics** data. Each of these "omics" types characterizes a certain level of gene expression. There are more kinds of "omics" data, e.g. **metabolomics** data recording the status of the metabolites, small molecules that are intermediates of the biochemical reactions that make up the metabolism. However, the following examples will be restricted to gene expression, for simplicity.

All of the above-mentioned levels of gene expression have been monitored already prior to the advent of high-throughput measuring techniques. The traditional way of study, e.g. by southern blot (genomics), northern blot (transcriptomics), or western blot (proteomics), is limited in the number of genes that can be recorded in one measurement, however. High-throughput techniques aim at multiplexing the assay, amplifying the number of genes measured in parallel by a factor of thousand or more, thus to assess the entire genome, methylome, transcriptome, or proteome of the organism under study. While such data bear great potential, e.g. for understanding the biological system as a whole, large numbers of simultaneously measured genes also introduce problems. Forty gene signals provided by traditional assays can be taken at face value as they are read out by eye (without requiring a computer). In contrast, 40,000 rows of recent quantitative data tables need careful statistical evaluation before being interpreted by machine learning techniques. Large numbers of e. g. transcription profiles necessitate statistical evaluation because any such profile may occur by chance within such a large data table.

Further, even disregarding all genes that do not show reproducible change throughout a set of biological conditions under study, computer-based interpretation (machine learning) is simply necessary, because the number of profiles showing significant change (mostly several hundreds to thousands) is still too large for visual inspection.

3. Computational requirements

With the necessity for computational data analysis, the question arises which type of computing power is needed. In contrast to e.g. sequence analysis, high-throughput data analysis does not need large amounts of processor time. Instead of parallelizing and batch-queuing, analysis proceeds interactively, tightly regulated, i.e. visually controlled, interpreted, and repeatedly parametrized by the user. However, high-throughput data analysis cannot always be performed on any desktop computer either, because it requires considerable amounts of RAM (at least for large datasets). Thus, although high-throughput data analysis may not require high-performance computing (in terms of "number crunching"), it is still best run on servers.

Using a server, its memory can be shared among many users logging in to it on demand. As detailed later, this kind of analysis can furthermore do with access to a database (4.3), webservice (4.2.1), and large numbers of different installed packages and libraries (4.1.3). Many of these software packages are open source and sometimes tricky to install. Apart from having at hand large chunks of RAM, the user is spared to perform tricky installations and updates as well as database administration. Webservers, database servers, and calculation servers sporting large numbers of heterogeneous, in part open-source packages and libraries are traditionally run on Unix operation systems. While in former times a lack of stability simply rendered Windows out of the question, it is still common belief among systems administrators that Unix maintenance is slightly less laborious. Also, I personally prefer Unix inter-process communication. Further it appears desirable to compile MATLAB code such that many users can use it on the server at the same time without running short of

licenses. Both licensed MATLAB and MATLAB compiler are available for both Windows and Unix. However, there are differences in graphics performance.
In 1998, MATLAB was still being developed in/for Unix. But times have changed. Graphics windows building up fast in Windows were appearing comparably slow when run under Unix ever since, suggesting that it is now being developed in/for Windows and merely ported to Unix. Performance was still bearable, however, until graphical user interface (GUI) such as menus, sliders, buttons etc. coded in C were entirely replaced by Java code. The Java versions are unbearably slow, particularly when accessed via secure shell (SSH) on a server from a client. For me that posed a serious problem. Being dependent on a Unix server solution for above reasons, I was seriously tempted to switch back to older MATLAB versions for the sole reason of perfect GUI performance. Also, I did not seem to be the only one having this problem. Comments on this I found on the internet tended to reflect some colleagues' anger to such extend that they cannot be cited here for reason of bad language. As older versions of MATLAB do not work for systems biology and other recent toolboxes, version downgrade was not an option. It therefore appeared that I had no choice other than to dispense with Unix / ssh. But what to do when client-side calculation is not possible for lack of memory? When switching to Windows is not intended?
A workaround presented itself with the development of data compression (plus caching and reduction of round trip time) for X connections designed for slow network connections. NX (http://www.nomachine.com) transports graphical data via the ssh port 22 with such high velocities that it nearly compensates for the poor Unix-server MATLAB-GUI performance. It was originally developed and the recent version is sold by the company Nomachine. There is also an open-source version maintained by Berlios (which unfortunately didn't work for all M-CHiPS functions in 2007). Needless to mention that I do hope that the Java GUI will be revisited by the Mathworks developing team in the future. But via NX, server-side Linux MATLAB graphics is useable. A further advantage of NX is that the free client is most easily set up on OSX or Windows running on the vast majority of lab clients as well as on the personal laptop of the average biologist. In this way, users can interact as if M-CHiPS were just another Windows program installed on their machine, but without tedious installation. Further, NX shows equally satisfying performance on clients old and new, having large or small memory, via connections fast and slow, i.e. even from home via DSL.

4. Data diversity and integration

Abovementioned configuration allows to provide MATLAB functions as well as other code to multiple users, e.g. within a department, core facility, company, or world-wide. As described, life scientists can use this service without having to bother with hardware administration, database administration, update or even installation. For these reasons, software as a service (SAAS) is a popular and also commercially successful way e.g. to deliver microarray analysis algorithms to the user. However, different users have different demands. The differences can roughly be categorized into being related to different technical platforms used for data acquisition (such as microarrays or mass spectrometry), related to different fields of research (plants or human cancer), or preference of certain machine learning methods.

4.1 Technical platforms

There is a multitude of different high-throughput techniques for acquiring "omics" data. As explained in section 2, following examples focus on the different regulatory levels of gene expression. In order to provide an outline of the technical development, microarray platforms are discussed in more detail.

4.1.1 Microarrays

Biological high-throughput quantification started out in the 1990s with the advent of cDNA microarrays. Originally, in comparison to recent arrays very large nylon membranes were hybridized with radioactively labelled transcripts. Within shortest time, microarrays became popular. Although (and possibly because few people were actually aware of this at that time) data quality was abysmally poor. The flexibility of the nylon membrane as well as first-version imaging programs intolerant of deviations from the spotting grid caused a considerable share of spots being affiliated to the wrong genes. Also, although radioactivity actually shows a superior (wider) linear range of measured intensities when compared to the recently used fluorescent dyes, it provided only for a single channel. Thus each difference in the amount of spotted cDNA, for example due to a differing concentration of the spotted liquid as caused by a newly made PCR for spotting a new array batch, directly affected the signal intensities. This heavily distorted observed transcription patterns. Nowadays, self-made microarrays are small glass slides (no flexibility, miniaturization increases the signal-to-noise ratio), hybridized with two colors (channels) simultaneously. The colors refer to two different biological conditions labelled with two different fluorescent dyes. RNA abundances under the two conditions under study compete for binding sites at the same spot. Ratios (e.g. red divided by green) reflecting this competition are less dependent on the absolute number of binding sites (i.e. the amount of spotted cDNA) than the absolute signal intensities of only one channel. While even modern self-made chips still suffer from other systematic errors, e.g. related to the difference between individual pins used for spotting or related to the spatial distribution throughout the chip surface, commercially available microarrays mostly do not show any of these problems any more. Furthermore, modern commercial arrays show lower noise levels in comparison to recent self-made arrays (and these in turn in comparison to previous versions of self-made arrays), thus increasing reproducibility.

But even more beneficial than the substantial increase in data quality since 1998 is the increase in the variety of what can be measured. While at first microarrays were used only for recording transcript (mRNA) abundance, all levels of regulation mentioned in section 2 nowadays can be measured with microarrays. **Genomic** microarrays can be used to assess DNA sequences, for example to monitor hotspots of HIV genome mutation enabling the virus to evade patients' immune systems (Gonzalez et al., 2004; Schanne et al., 2008). **Epigenomic** microarrays that assess the methylation status of so-called CpG islands in or near promoters (regulatory sequences) of genes are used e.g. to study epigenetic changes in cancer. **Transcriptomic** (mRNA detecting) microarrays are still heavily used, the trend going from self-made arrays (cDNA spotted on glass support) to commercial platforms comprising photo-chemically on-chip synthesized oligomeres (Affimetrix), oligomeres applied to the chip surface by ink jet technology (Agilent), or first immobilized on tiny beads that in turn are randomly dispersed over the chip surface (Illumina), just to provide a few examples. Recently, the role of transcriptomic microarrays is gradually taken over by so-called next generation sequencing. Here, mRNA molecules (after being reversely transcribed into cDNA molecules) are sequenced. Instances of occurrence of each sequence are counted, providing a score for mRNA abundance in the cell. While sequencing as such is a long-established technique, throughput and feasibility necessary for transcriptomics use

by ordinary laboratories has been achieved only few years ago. Nevertheless, this technique may well supersede transcriptomic microarrays in the near future. **Proteomic** microarrays are used to assess abundances of the ultimate products of gene expression, the proteins. To this end, molecules able to specifically bind a certain protein, so-called antibodies, are immobilized on the microarray. Incubating such a chip with a mixture of proteins from a biological sample labelled with a fluorescent dye, each protein binds to its antibody. Its abundance (concentration) will be proportional to the detected fluorescent signal.
Unfortunately, the affinities of antibodies to their proteins differ considerably from antibody to antibody. These differences are even more severe than the differences in the amount of spotted cDNA abovementioned for transcriptomic cDNA microarrays. Thus the absolute signals can not be taken at face value. However, as for the transcriptomic cDNA arrays, a possible solution is to incubate with two different samples, each labelled with a different color (fluorescent dye). The ratio of the two signal intensities (e.g. a protein being two-fold upregulated in cancer as compared to normal tissue) for each protein will be largely independent of the antibody affinities. More than two conditions (dyes) can be measured simultaneously, each resulting in a so-called "channel" of the measurement.

4.1.2 Other platforms

The general categorization into single-channel and multi-channel data also applies to other technical platforms. There are, for example, both single-channel and multi-channel quantitative mass spectrometry and 2D-gel data. Using 2D-gels, a complex mixture of proteins extracted from a given sample is separated first by charge (first dimension), thereafter by mass (second dimension). In contrast to the microarray technique, the separation is not achieved by each protein binding to its specific antibody immobilized on the chip at a certain location. Instead, proteins are separated by running through the gel in an electric field, their velocity depending on their specific charge, and their size. As for microarrays, the separation results in each protein being located at a different x-y-coordinate, thus providing a distinct signal. A gel can be loaded with a protein mixture from only one biological condition, quantifying the proteins e.g. by measuring the staining intensity of a silver staining, resulting in single-channel data. For multi-channel data, protein mixtures stemming from different biological conditions are labelled with different fluorescent dyes, one color for each biological condition. Thus, after running the gel, at the specific x-y-location of a certain protein each color refers to the abundance of that protein under a certain condition. Unlike with microarrays, there is no competition for binding sites at a certain location among protein molecules of different color. Nevertheless, data of different channels are not completely independent.
In general, regardless of the technique, separate channels acquired by the same measurement (i.e. hybridization, incubation, gel, run, ...) share the systematic errors of this particular measurement and thus tend show a certain degree of dependency. They should therefore not be handled in the same way as single-channel data, where each "channel" stems from a separate measurement. Data representation (database structure, MATLAB variables, etc.) and algorithms need to be designed accordingly. Fortunately, independent of the particular platform, the acquired data are always either single- or multi-channel data. In the latter case, different channels stemming from the same measurement show a certain degree of dependency. This is also true for all technical platforms.
As a last example of this incomplete list of quantitative high-throughput techniques assessing biological samples, I will briefly mention a technique that, albeit long

established for small molecules, only recently unfolded its potential for high-throughput quantitative proteomics. Mass spectrometry assesses the mass-to-charge ratio of ions. To this end, proteins are first digested into smaller pieces (peptides) by enzymes (e.g. trypsine), then separated (e.g. by liquid chromatography) before being ionized. Ionization can be carried out e.g. by a laser beam from a crystalline matrix (matrix-assisted laser desorption/ionization, abbreviated MALDI) or by dispersion into an aerosol from a liquid (eletrospray ionization, ESI). Movement of these ions in an electric field (in high vacuum) is observed in order to determine their mass-to-charge ratio. This can be achieved simply by measuring the time an ion needs to travel from one end of an evacuated tube to the other (time of flight, TOF), or by other means (e.g. Quadrupole, Orbitrap). The detection works via induced charge when the ion hits a surface (at the destination end of the flight-tube in case of TOF) or e.g. via an AC image current induced as oscillating ions pass nearby (Orbitrap).

Unlike e.g. for antibody microarray data where each protein can be identified through its location on the array, for mass spectrometry the quantification must be accompanied by a complex identification procedure. To this end, ions of a particular mass-to-charge ratio are fragmented by collision with inert gas molecules (mostly nitrogen or argon). The fragments are then subjected to a second round of mass spectrometry assessment (tandem mass spectrometry or MS/MS). The resulting MS2 spectrum contains enough information to identify the unfragmented peptide ion, in a second step eventually enabling to deduce the original protein. Like other techniques, quantitative mass spectrometry can be used to execute single-channel measurements (label-free) or to produce multi-channel data, measuring several biological conditions (up to 6 e.g. via TMT labelling) at the same time.

4.1.3 Data integration

Above examples illustrate that the input into any comprehensive software solution is highly diverse. For cDNA microarrays alone several so called imaging software packages exist (e.g. Genepix, Bioimage, AIS and Xdigitize) that convert the pixel intensities of the scanned microarray image into one signal intensity per gene. Also, specialized software is available for the equivalent task in case of 2D-gels (e.g. Decider) and for protein identification in case of mass spectrometry (Mascot, Sequest), just to name few examples. Thus, the first step necessarily means to parse different formats for import. Furthermore, different platforms require different preprocessing steps which deal with platform-specific systematic errors. While local background subtraction may alleviate local spatial bias in different areas of a microarray, mass spectrometry spectra may require isotope correction and other measures specific for mass spectrometry. Any comprehensive software solution necessarily needs to provide a considerable number of specialized algorithms in order to parse and preprocess each type of data.

On the positive side, there are also certain preprocessing steps required for all platforms alike. Normalization of multiplicative and/or additive offsets between different biological conditions is generally required, since pipetting errors or different label incorporation rates affect the overall signal intensities obtained for each biological sample. Also, more than half of the genes of higher organisms tend to be not expressed to a measurable amount in a typical multi-conditional experiment (with the exception of studying embryonic development). Thus, for each dataset, regardless of the technique it is acquired by, genes whose signal intensities remain below the detection limit throughout all biological conditions under study can (and should) be filtered out. Regarding the fold-changes (ratios with a certain biological reference condition in the denominator) instead of absolute signal intensities is common practice for microarray and other high-throughput data. A gene switching from a signal of 0.001 (e.g. in normal tissue) to 0.002 (in cancer) would be otherwise interpreted as being two-fold upregulated, although meaningful signals may start only in the range of 10^3.

Further, measurements need to be performed repeatedly in order to assess the reproducibility of a signal. Repetitions are cost- and labor-intensive. There have been many attempts to compute a p-value from one single measurement alone, more than ten years ago for microarray measurements as well as recently in the mass spectrometry field (Zhang et al., 2006). However, distributions of gene abundance signals tend to vary, e.g. with signal intensity. For each one-measurement statistical tests I know of, a quantile-quantile plot revealed that its distribution assumption does not hold. Thus these tests do not at all yield proper p-values. While this is inconvenient for the wet-lab life scientist, it simplifies data integration for the bioinformatician. For few, i.e. in the range of three to four repetitions, the significance of gene signals can be tested e.g. using the limma package (by Gordon Smyth; for reference see Smyth, 2005), which is based on a very reasonable distribution assumption. This method, albeit originally developed for microarray data, seems to work properly for mass spectrometry data, as well. If there are six or more repeatedly performed measurements per condition (there usually are not), permutation tests such as the Significance Analysis of Microarray Data (SAM; Tusher, 2001) should be used. The latter can do without any distribution assumption, extracting the distribution from the data (by randomly permutating the measurements many times). Thus, there is no reason not to use it e.g. also for mass spectrometry data. At the very moment, six repetitions represent considerable costs here. However, this may become feasible in the future.

Thus, data integration can be achieved by a limited set of platform-specific preprocessing steps before data are collected into a genes × measurements matrix variable. The last step of preprocessing, the normalization, can be applied to data stemming from any platform in the same manner. There are different normalization algorithms such as loglinear normalization (Beißbarth, 2000), locally weighted scatterplot smoothing (LOWESS; Cleveland, 1979), quantile normalization (Bolstad, 2003), or variance stabilization (Huber, 2002) that can be applied under different circumstances (e.g. variance stabilization when differing variances for low and high signal intensities pose a problem). However, the choice depends less on the particular platform, but on particularities of the data and in part on personal preference (which will be discussed in detail in 4.2.3). The way a normalization is iterated to produce directly comparable numbers throughout all measurements of a multiconditional experiment depends on the data being single- or multichannel. In the latter case each non-control channel is fitted to the local control channel of the same measurement (hybridization, incubation, run, ...). In the former case (single channel), each measurement is fitted to the median of the repeatedly performed measurements of the control condition. Either single- or multi-channel data will be obtained from any technical platform. Thus, one of the two above ways to iterate normalization methods will be applicable – in combination with any of the above normalization methods, regardless of which technical platform the data stem from.

The resulting normalized data can universally be stored in another genes × measurements matrix of the same size. Thereafter, single genes are filtered out that show low signals throughout the conditions under study, or insignificant (e.g. irreproducible) change. Sometimes it pays off to discard a single (outlying) measurement instead of too many genes. In each case, the filtering process results in yet another genes × measurements matrix, but one of considerably reduced size, this time. Thus, preprocessing results stemming from a plethora of different technical platforms are stored in a common format.

4.1.4 Coding requirements

Above examples illustrate the demands placed on any comprehensive software solution. It needs to provide a multitude of both platform-specific and ubiquitously applicable algorithms. For any larger collection of interacting functions, one should think about ways to intelligently structure such code in order to properly develop a larger software package. Aspects range from re-using code, object oriented programming, and providing to the users a quick (one-click, automated) way of reporting bugs, to implementing a quickly adaptable menu structure, and using a concurrent version system.

In addition to simply being large, such a collection of algorithms tends to be under constant development. Better versions of already comprised algorithms will appear, new functions will need to be added regularly. Without extensive manpower, satisfactory maintenance of the system is only feasible when original source code (delivered along with the published method) can be plugged. Unfortunately, the vast majority of abovementioned algorithms for parsing data from different microarray platforms, preprocessing, and statistics is written in R. The programming environment R (http://www.r-project.org), an open-source version of S+, is the „natural habitat" of the statistician. In contrast to Matlab it provides tailored data types facilitating the handling of factors and levels, and more than one type of for missing values (NaN), to provide only two examples. For microarray and other high-throughput biological data, there is a comprehensive open-source R toolbox called Bioconductor (www.bioconductor.org), providing a collection of recently 460 R packages written and maintained by scientists all over the world, free to use. Several collegues (bioinformaticians) who were programming in Matlab in the 1980s switched to R for one or the other of above advantages since then.

However, Bioconductor can be regarded a platform made by bioinformaticians for bioinformaticians. Command-line style invocation and parametrization tends to „unhinge" many biologists who prefer clearly laid out menus, buttons and sliders, interactive graphics, in short a program that can be entirely operated by mouse click. Graphical user interfaces (GUI) as well as interactive graphs (e.g. returning x and y coordinates upon mouse click into the figure), although by now possible also in R, are the traditional domain of Matlab. In my opinion, implementation of both is considerably easier and thus faster than in R even to date. Moreover, in the field of systems biology (the science of modeling, simulating, and predicting the interplay of genes as a whole) the trend appears to be vice versa, with more tools being coded in Matlab than in R. In order to combine systems biology as well as fast implementation of user-friendly GUI and interactive graphs of Matlab with the statistical treasure trove readily available in R, both environments need to be interfaced. This can be achieved e.g. by the R.matlab-package maintained by Henrik Bengtsson (http://cran.**r**-project.org/web/packages/R.matlab). Entirely written in R, it provides (amongst other options for establishing a connection) two functions converting variables dumped into a matlab workspace (.mat) file into an R workspace and vice versa. The slight performance disadvantage of writing to and reading from hard disk is more than compensated for by perfect safety and reliability (no memory manipulation, no segmentation faults).

Furthermore, the interface is most easy to use (invoking one function for reading, one for writing) and appears to convert Matlab variables of each class (at least all classes we tested, incl. e.g. structs) into the best corresponding R data type. In order to plug algorithms written in R in a multi-user scenario, M-CHiPS dumps only the required variables into a .mat file located in the /tmp folder, its filename comprising the user name (so to prevent collision with other users' actions). Then an R shell is invoked by Unix command that reads the variables, invokes the R function to perform, and stores the result in another .mat file. For seamless inter-process communication via hard disk, it is advisable to await complete writing of a file by the other process (e.g. by using a different result filename for the R to Matlab direction and waiting for the R process to delete the first file as a signal that it finished writing the second one). As both processes run on the same machine (meaning the same hard drive buffer), this procedure is reasonably fast. The time needed for transporting the data to and from R is negligible in comparision to the runtime of any R code that was interfaced to M-CHiPS, as long as only the required variables are transferred instead of transferring the whole workspace.

4.2 Machine learning methods

As shown above, preprocessing of biological high-throughput quantifications generally starts with parsing the particular format of an input file. The imported data sometimes have to undergo platform-specific preprocessing steps, always followed by normalization and filtering. The preprocessing ends with a data table of reduced size, rows representing genes, columns representing measurements. Normalization has taken care of systematic differences among measurements (caused e.g. by differing label incorporation rates) rendering all numbers in the data table directly comparable. The data are now ready for "high-level analysis" (as opposed to preprocessing).

4.2.1 Common workspace organization

From before normalization and onwards, the data are also kept in common format, regardless of the acquisition platform. Thus, high-level analysis can take place on a commonly structured workspace. The M-CHiPS workspace documentation (http://mchips.org/workspace.html) provides an example of how such a workspace may look like. Fig. 1 shows the format of selected variables. Variable names are provided in blue color. Variable prim shows the typical genes × measurements matrix format, while stain and multichannel are vectors keeping track of which measurement belongs to which biological condition and to which hybridization (incubation, gel, or MS run), respectively. The names are related to the first task they were introduced for, i. e. color-coding (staining) graphical objects according to the biological conditions they belong to, and affiliating the different channels to multi-channel hybridizations, respectively. The name "prim" is explained later. Variables ngen and nexp are both scalars holding the numbers of genes and measurements (1998 referred to as "experiments" before the latter term was transferred to the multi-conditional dataset as a whole), respectively. The variable prim records the data before normalization. Normalized data and ratios are stored in separate variables. At the risk of occupying too much memory, keeping both the raw data and normalized intensities is necessary because analysis is an iterative procedure rather than following a fixed workflow. Both ratios and absolute intensities (as well as ranks) may be subjected to analysis algorithms in any temporal order. Even the raw data may be needed at a later stage, e. g. if the resulting plots reveal saturation effects. In this case, data will be re-filtered for saturation which is best detected by assessing the raw data.

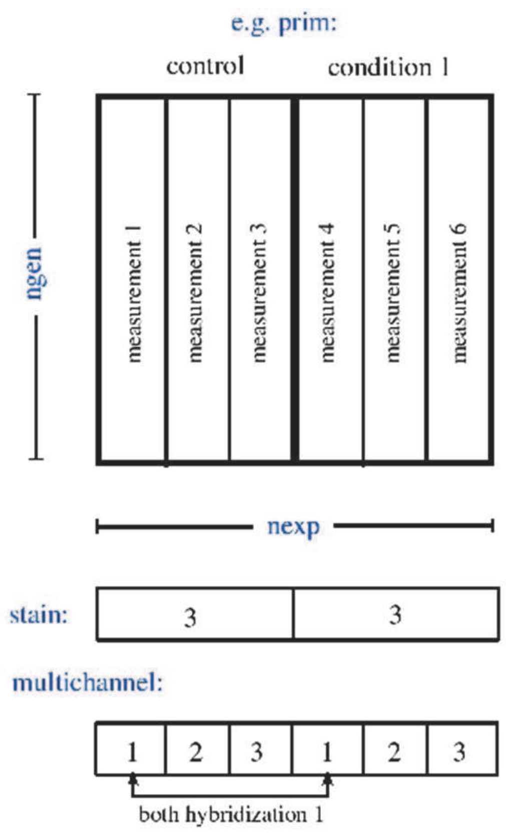

Fig. 1. Selected variable formats, variable names are shown in blue.

Because filtering can be repeated with different parameters at any time, rows representing genes that have been filtered out cannot be removed from the variables, either. Index variable “geneorder” holds the row numbers of the genes that “survived” the current filtering, in a sequence determined by a current sorting criterion. Other indexes record which genes (or measurements) have been selected by the user. Tab. 1 provides an exemplary list of variables, sorted by content and format.
The name “prim” stands for the primary of e.g. two spot sets on a microarray that may result from spotting each gene in duplicate. Thus, the purpose of prim (and secu) is to separately hold each single set of signals (available for all genes) that stems from the same channel or from the same single-channel measurement. Such sets stemming from the same channel share the same measuring procedure including sampling, labelling, as well as hybridization, incubation, or the like, thus being highly dependent. As a result, they cannot serve as independent repetitions for statistical tests (i.e. they ought to be averaged beforehand). Nevertheless, for the M-CHiPS workspace, such sets are kept separate because they can provide an “atomic” unit of variance in plots whose axes are dimensionless. A lack of units is typical for the entire field. Raw data are acquired by e.g. scanners or mass detectors in arbitrary machine units. Already at the stage of data acquisition, these machine units tend to be incomparable to those of data produced by other machines. Later on during data analysis, the signal intensities are often converted into ratios or distance measures for

which the unit cancels out. In such cases, the difference between primary and secondary spots on an array stands for the minimal distance beyond which biological differences cannot be resolved even after optimizing the wet-lab protocol. Simply put, whenever a difference between two conditions is not considerably larger than this minimal distance, the conditions cannot be distinguished by the technique.

- Data
  - ngen x nexp data matrices:
    - prim - raw data (primary spot set)
    - secu - raw data (secondary spot set)
    - fitprim - normalized absolute (or estimated) intensities (primary spot set)
    - fitsecu - normalized absolute (or estimated) intensities (secondary spot set)
    - flatprim - normalized (linear) ratios (primary spot set)
    - flatsecu - normalized (linear) ratios (secondary spot set)
    - there are distances and ranks, as well
    - gen (containing the spot numbers) as well as experim (containing the measurement IDs) are in the same format for easy handling)
  - ngen x ncon (quality-) matrices:
    - pvalue - two-class pvalues
    - fpvalue - multi-class (i.e. only one per gene, but same size for compatibility)
    - minmaxseparation - min/max separation (Beißbarth, 2000)
    - stddevseparation – standard deviation separation (Beißbarth, 2000)
  - data annotation:
    - ginfo - gene annotations
    - anno - experiment annotations
- Metadata
    - family - array family (chip type)
    - experim - measurement IDs, multichannel - hybridizations
    - backgroundsubtraction
    - normalization
    - pvalues
    - filtermode1, filtermodebytimepoints1, filtermode2, filtermodebytimepoints2, filtermode3, filtermodebytimepoints3 - filter constraints
    - geneorder - filter result: index of filtered genes (sorted by sortmode)
    - ca.meta - correspondence analysis (input, type, ...)

Table 1. Incomplete workspace list, variable names are shown in blue

In addition to gene × measurement data matrices, Tab. 1 lists matrices comprising only as many columns as there are biological conditions (comprising quality scores such as p-values) as well as differently structured variables holding gene and experiment annotations that will be discussed below (4.3.1). Last but not least, all analysis steps – including the selection of raw data, all preprocessing steps as well as "high-level analyses" are recorded along with their parametrization. Thus, the entire analysis procedure is comprehensively documented by metadata in order to be able to reproduce the result. Metadata are also listed in the header of result reports that can be shared via internet by mouse click. Result reports are protected by passwords that are made available to collaborators. To date, they can consist of a shortlist of user-selected genes with signal intensities and ratios color-coded according to statistical significance (HTML), or of a complete list of raw signals plus nearly all computed values (tab delimited spreadsheet), or of MATLAB figures amended with explanatory text (HTML).

4.2.2 Machine learning

The MATLAB figures depicted in such result reports are generally produced by machine learning techniques. Machine learning can be divided into supervised and unsupervised learning. Supervised learning, also called classification, takes as input a grouping of objects of a so-called "training set". Such a training set may e.g. consist of tumor samples for which the exact tumor type (class) is known. The method learns properties within the data (e.g. affiliated to the expression profiles of certain genes) that can serve to discriminate these classes. Any such property (e.g. high expression of a certain gene) should be homogenously present within a certain class, but absent in at least one other class, so to distinguish the two classes. The entirety of learned properties is called a "classifier". The classifier can be used to sort a new cancer sample of yet unknown class affiliation into the correct class. While this is practised with data acquired from tailor-made cancer microarrays in order to provide physicians with additional information for their decisions on cancer therapy, its use is largely restricted to the clinics or any other area of application providing established and clear-cut classes to affiliate to.

Basic research often lacks such established classes or any other prior knowledge about the acquired data. At first, new hypothesis need to be generated by a more exploratory approach. Unsupervised learning does not need any group affiliations beforehand. Solely on the basis of the data, unsupervised machine learning methods extract clusters of objects whose quantified signals behave similar within the cluster but different in comparison to objects of a different cluster. Naming all unsupervised methods already being applied to the analysis of biological high-throughput data is beyond the scope of this chapter. It would result in an outline of applied statistics. Additionally, a few methods have been newly developed specifically for microarray data. Interestingly, the first clustering method applied to microarray data by Eisen and co-workers (Eisen, 1998), hierarchical clustering, is still most widely used. Because each following method had to be proven superior in one or the other aspect in order to get published, this appears tantamount e.g. to using Windows 3.1.

The one aspect all unsupervised methods have in common is that a scoring for similar (or dissimilar) behaviour has to be defined beforehand. While some algorithms are traditionally run on one particular so-called "distance measure", others operate on a wide variety of distance measures. In principle, each algorithm can be adapted to run on each distance measure. This is just not supported by each piece of software and might require altering the original source code.

4.2.3 User preferences and user-friendliness

User preferences are diverse, with each user preferring a particular method she or he is used to. Having seen many different data sets visualized by one and the same preferred method, a user is able to assess data quality as well as predominant variances and coherences with one quick glance. Being used to a method, interpretation of the produced plots takes only a minimum of time. Also, the hypothesis generated hereby tend to prove correct the more often, the more plots of the kind the user has studied. Thus, an experienced user will prefer her or his particular method for good reason. In other words, lack of one particular method represents a good reason for not using a software. Therefore, all commercial as well as the vast majority of academic packages aim at implementing the entire set of unsupervised methods as comprehensively as possible. Abovementioned common organization of workspace variables facilitates plugging of a new method. Interfacing to R commonly enables to do so even without translating the code.

But is "the more" always "the better"? Obviously, more methods increase chances to find one's preferred method. However, this applies only to the experienced user. For users new to the field, an overwhelming number of possibilities represents more of a curse rather than being an asset. For hierarchical clustering alone, they need to choose between various distance measures (Euklidean, Mahalanobis, City Block, X^2, Correlation, ...), in combination with either single or complete or average linkage. Above six distance measures multiplied with three different ways to update the distance table upon merging two clusters result in 18 possibilities to parametrize only one method. Each way recognizes different properties of the data, and will thus visualize different patterns. Thus, the 18 different parametrizations will produce 18 different results. Multiplied by a considerable number of different methods, the user will be faced with an overwhelming number of different results, not knowing which to select for further investigation. Making such an unpleasant situation even worse, analysis of biological high-throughput data holds in stock a number of pitfalls to the inexperienced. To pick one from the examples already provided, Euclidean distance, albeit simple and commonly known (representing the "every-day distance" computed by Pythagoras), and therefore often listed as first item of pull-down menus, is by no means suitable for clustering absolute signal intensities in a biological context. Here, it is useful to sort e.g. a transcription factor whose abundance regulates the expression of a set of target genes into the same cluster as its targets. Biological conditions in which the transcription factor is highly abundant will show high abundance also for the target genes, and vice versa. However, while the transcription factor shares the expression behaviour (shape of curve) with its targets, its absolute expression level (amplitude of curve) will be much lower. Therefore, Euklidean distance will affiliate transcription factor and targets into different clusters (of low and high expression, regardless of relative trends). Other distance measures such as the correlation distance (computed as one minus the correlation coefficient) will cluster together similarly shaped (i.e. correlating) expression profiles regardless of the absolute level of expression and are therefore more useful for biological research.

In order to grant usability also to the inexperienced, a variety of measures can be taken. Different options, either for choosing one of many possible algorithms for a certain task, or for parametrizing it, should be always accompanied by one default suggestion which works reasonably well for most types of data. Sharing a common method among different users also facilitates communication among collaborating scientists. Data and results can be shared without explaining the process from the former to the latter. Along the same lines, abovementioned automated web distribution of result reports facilitates using the system as a communication platform.

It goes without saying that usability of a software package is further enhanced by a clear menu structure guiding the user to the methods provided for a certain task with one default method clearly tagged. However, there are also more complex tasks for which providing one default workflow is insufficient. "Experience" e.g. of how a proper signal intensity threshold can be estimated and applied in order to filter out inactive genes or which reproducibility measure makes sense for a certain number of repetitions (see 4.1.3) can be handed on from the programmer to the user in form of a wizard. In M-CHiPS, the filter wizard, after assessing the data, selects from the multitude of possible measures (here filters as well as filter parameters) a subset that, according to more than thirteen years of filtering high-throughput data, appears most suitable for these data. However, since unexpected situations may occur with any new dataset, this is only a starting point. The results should be critically supervised by the user. Starting from this "initial guess", the user also needs to adapt the parameters iteratively in order to optimize filtering results. To this end, the wizard presents the suggested filters in a temporal sequence best suited for visual supervision of their outcome. It also provides guidance with respect to the parametrization by briefly (in few words) hinting at what to look at and by asking simple questions. Much like preferring mouse click over command line operation, users tend to achieve their goal (e.g. proper filtering) much faster by using a wizard than by reading the manual.

4.2.4 Programmers' preferences and ease of implementation

Providing such user friendly features (in particular wizards) costs considerable time. In contrast to commercial enterprises, implementing a user friendly system in an academic setting appears an ambitious goal. Packages coded by scientists for scientists (such as Bioconductor) tend to focus on command line interfaces and manuals (here "vignettes") instead of featuring mouse-clicks and wizards.

On the positive side, MATLAB provides the opportunity to code graphical objects very quickly. Figures are important for the user to visually supervise the analysis process. In M-CHiPS, normalization of each measurement can be evaluated by looking at a scatter plot of the measurement versus the control it is to be fitted to. As the human eye is able to detect patterns in fractions of seconds, artefacts such as saturation effects (data points concentrated into a line orthogonal to a particular axis as if "shifted by a snowplough") will not escape the user, even within the short timeframe the figures need to build up. The normalization performance is represented by a regression line or curve (depending on linear or log scale and on the normalization method). If the regression line leaves the center of the cloud of data points, the user will revisit the plot in order to decide on using a different normalization method, using a saturation filter, or discarding this particular measurement for poor quality.

In MATLAB, such figures can also be easily interacted with by mouse click, e.g. for selecting a cluster of genes by clicking a fence around it that is closed by hitting the middle mouse button. Furthermore, the M-CHiPS workspace holds variables for storing the status of such selections (see http://mchips.org/workspace.html, at the end of the paragraph headed "Genes"). Each newly coded figure displaying genes can be quickly endowed with gene tags by adding one command. Thus MATLAB provides the opportunity for user-friendliness and swift coding at the same time.

As already discussed, implementation time for many statistical algorithms can be saved by interfacing to R. Much like a user prefers certain methods, a programmer will save considerable time in programming environments she or he is used to. This applies not only to MATLAB and R. Personally, I am used to coding tasks requiring regular expressions in Perl, simply because in former times this was not possible in MATLAB. While nowadays comfortable handling of regular expressions is available both in MATLAB and R, I still prefer to use Perl for regular expressions. As discussed below (4.3), other tasks are best

implemented on database level, by using SQL. Thus, interfacing to other languages not only provides the opportunity to plug already implemented code. It also eases new implementations by meeting programmers' preferences as well as by providing the "ideal" environment for each particular task.
As one would expect, using MATLAB, R, C, Perl, Java, and SQL within one system also causes problems. Sloppy programming of interfaces (e.g. accessing memory not properly allocated) may result in fatal errors (segmentation faults). Object oriented programming may increase safety and avoid bugs within one language. Nowadays, this is possible for all of the above languages except SQL. But how to send objects and events back and forth through an interface? For M-CHiPS, it proved a successful strategy to keep any interface as simple as possible. The database interface transports only a limited number of data types, resulting in short mex files. Type casting can be done within either MATLAB or SQL. Also, the heterogeneity of factors and levels associated to different fields of research (as discussed in the next chapter) is handled already at database level instead of implementing a large and error prone middleware. Data are transported to and from Perl simply via Unix pipe. Wherever possible, already implemented and tested interfaces where used, such as Perl DBI (http://dbi.perl.org) for accessing the database from Perl. This module is also independent of the database management system (DBMS) used. Using query syntax common to all SQL dialects, in particular refraining from object-relational extensions which tend to be DBMS specific, allows to switch e.g. from PostgreSQL to Oracle without breaking code.
Thus, either using well-established or simple and clear-cut own implementations, problems caused by interfaces are minimal. However, for seamless interaction of the components, such a heterogeneous system requires many modules, libraries, and packages. Installation represents a considerable workload in and of itself, interfering with distributing the system. With the advent of server virtualization, however, all components can be distributed as a whole. Such a virtual machine can be regarded a "computer within a computer". It comprises the system including all necessary modules, libraries, and packages plus the operation system in a tested configuration. Further, it will run (with few exceptions) on any hardware and, thanks to extensive standardization, on (almost) any host operation system.

4.3 Fields of research

Differences related to different technical platforms used for data acquisition (such as microarrays or mass spectrometry), as well as user preferences of different machine learning methods result in the need to add and maintain a large set of algorithms, a task best coped with by interfacing to R and other languages. However, in addition to using different platforms to acquire their data, users will work on different types of samples stemming from highly diverse biological contexts. In the following I will put forward the thesis that computer-aided interpretation of data from different fields of research with one and the same software package is best coped with by interfacing to a database.

4.3.1 Data interpretation

As already mentioned (4.2), analysis of data stemming from new research is necessarily exploratory. In many cases, few hypothesis exist. Thus, the first task of data analysis is to generate hypotheses that can stand verification by subsequent statistical tests. To this end, M-CHiPS provides an unsupervised method that is most exploratory and easy to parametrize as a common view. Correspondence analysis (Fellenberg, 2001) is regularly used by most M-CHiPS users. Belonging to the subclass of planar projection (also called ordination) methods, it shows how discrete (or fuzzy) cluster borders are. Much like principal components analysis, objects will be neighbours in the projection plot, whose quantified signals behave similar. However, unlike principal component analysis, it is able to visualize more than one kind of objects at the same time, also displaying the correspondence (interrelation) between objects of different kind. At this stage, only two kinds of objects exist, genes and measurements.

A typical data matrix comprises three or more repeatedly performed measurements (as columns) for each biological condition, with two to at most a few hundred conditions. In stark contrast, the numbers of genes typically range from minimum hundreds to mostly several ten thousands, also resulting in large numbers of genes within each observed cluster. In order to interpret the data, it is necessary to characterize such a gene cluster in terms of one or more descriptive traits. Because a list of hundreds of genes along with their names and traits is too large for visual inspection, extracting descriptive traits has better be done by automatic means. A trait is the more descriptive for a chosen cluster, the more common it is to all (or at least a large share of) cluster members, and the fewer its occurrences outside the cluster (hence to discriminate the cluster from other clusters). Apart from simply comparing the overall frequency of each trait to its frequency in a particular user-selected cluster (Fellenberg, 2003), the traits can e.g. also be visualized by correspondence analysis. To this end, characteristic traits are filtered (from the vast majority of uninformative ones) and displayed as centroids in the middle of the cluster of genes they apply to (Busold, 2005).

For any statistics simple or sophisticated the instances of occurrence for any given trait must be countable by a computer. For genes, among other sources, the gene ontology (GO) initiative (http://geneontology.org) provides a controlled vocabulary of terms annotating gene products in a computer-readable format. The terms are subcategorized from general to special traits by “is a” and “part of” relations that form the edges of a directed acyclic graph. The same set of terms (ordered by the same graph) is applicable to the genes of every organism under study, a priceless bonus for data integration.

4.3.2 Differences between yeast, plants, and human cancer

The value of any trait set universally applicable to all fields of research can be best appreciated by regarding traits for which this is not the case. While any gene either shows phospholipid-translocating ATPase activity (GO:0004012) or not, growth conditions may be characterized by temperature, additive concentrations, and other variables of continuous range. Also, a biological sample can be processed by a wide variety of different wet-lab protocols prior to data acquisition and preprocessing. Each step may influence the observed patterns by contributing specific systematic errors. Thus, representing these steps within the sample annotations can serve to track down artefacts. Even more complex than the description of wet-lab protocols (but also more interesting) is the description of the samples. Their composition ranges from only one cell type and heterogeneous mixtures to complete organs and organisms. Organisms, genotypes, phenotypes, disease stages, clinical data, or culture conditions may need to be accounted for. Biological contexts under study by means of high-throughput quantification are highly diverse. Traits annotating culture conditions for yeast are not applicable to plants growing in a green house. Soil type and circadian light rythms are irrelevant to cancer research. There are initiatives to describe all fields of research into one ontology (MAGE-OM; Brazma, 2003). However, the complexity of the controlled

vocabulary is overwhelming. Most terms are not applicable to a particular field of research, while terms important to describe novel aspects of a new biological context are often missing. Often it is not known a priori if a particular trait is relevant for the biological context under study or not. Omitting it bears the risk of overlooking something important.
As for the genes, lists of hundreds of samples along with their specific traits are too large for visual inspection. Thus, the global players driving the observed expression patterns of the samples need to be extracted from a large number of irrelevant traits computationally. M-CHiPS organizes these traits in a database tailored for computational analysis (data warehouse; Fellenberg, 2002). The database structure is flexible enough to provide for each field of research a tailor-made arbitrarily structured trait set comprising both enumeration type variables and those of continuous range. It also accounts for rapid growths of each trait set with new kinds of experiments. These very different trait sets (currently 13, see http://mchips.de#annos) are presented to the analysis algorithms in a unified way such that all fields of research can be operated by the same algorithms. As already mentioned, the heterogeneity of the sets is already encapsulated on database level (granted US patent US7650343) such that no middleware is required and the database interface can be kept small and simple.
As for the genes and samples, or for genes and samples and gene traits, correspondence analysis can visualize correspondence between genes and samples and sample traits at the same time. Like the gene traits, sample traits are represented as centroids of the samples they apply to. Prior to visualization, however, any continuous value ranges must be discretized into bins of highest possible correlation to the expression data. Furthermore, the filtering approach (selecting relevant traits) is a different one. However, the main principle difference to interpreting genes by means of gene traits is that sample traits only become necessary for computer-aided interpretation of large numbers of samples. For two or three conditions under study, an analyst can keep track of all differences without using a computer. In this case, one correspondence analysis plot is able to capture the total variance (information content, inertia) between all conditions. The typical use case for sample traits is more a dataset comprising hundreds of samples, e.g. stemming from cancer biopsies. Here, a single plot can only account for the predominant variations, at the risk overlooking minor (but possibly interesting) changes. Since pathological classification is not always reliable and (more importantly) because unexpected groupings could be concealed by imposing known classes, the variance cannot be reduced in this way. However, the total variance can be systematically dissected into pieces sequentially visualized by separate correspondence analysis plots. Thus, no important detail can escape the analyst's attention (Fellenberg, 2006).

5. Systems biology

Systematic interpretation, as described above, means to systematically screen the entire information content of a data matrix step by step for exploratory research (as opposed to verifying a hypothesis or just trying out things). It should be carefully discriminated from data interpretation at systems level, which stands for a different approach. Here, the focus lies on the interplay of genes. A gene may be regulated by one or more other genes or by its own abundance (auto regulation), or it may regulate one or more other genes. For some genes, all three events may even take place at the same time. A gene regulatory network, albeit complex, is by no means chaotic. Several so-called network motives (e.g. feed-forward

loops) could be identified to occur significantly more often in biological networks than at random (Alon, 2007). Network fluxes are rigorously controlled, resources carefully spent only where needed (e.g. as an investment into swift reaction times). Much like traffic lights, key switches tightly coordinate the temporal order of important events, e.g. for cell division.
Reverse engineering gene regulatory networks stands for reconstructing such networks from data. Each dependency (e.g. gene A represses gene B) must be estimated from the traces it leaves in data (signals for B decrease when signals for A increase). Therefore, data obtained from controlled system perturbations (removing one or more particular genes at a time, e.g. by so-called "RNA Inference") are most valuable for reverse engineering, followed by time course data. Temporally unrelated biological conditions (e. g. cancer biopsies, each representing an end point of possibly different courses of disease progression) are less informative, but reverse engineering is still possible (Basso, 2005). Traditionally, reverse engineering is carried out with few selected genes, only. One cannot expect to elucidate the dependencies of some ten thousand genes on the basis of only dozens of observations (measurements). Once again, data integration is vital. Since large datasets are rare, it pays off to merge several smaller datasets in order to delineate robust networks for considerable numbers of genes. This is even possible across different technical platforms by adapting the differing scales (Culhane, 2003).
However, many datasets that would be interesting to merge have been recorded for different species, posing the additional problem to affiliate genes across species. Orthology relations (affiliating genes of one species to another) can be one-to-one, one-to-many, or even many-to-many. Furthermore, evolution events so-called sub- or neofunctionalization may assign new functions to certain genes. This results in that some genes of the same (or a very similar) sequence carry out "different jobs" in different species and should thus not be affiliated for merging datasets. In March 2010, we published a method capable of merging datasets across species on the basis of the expression data alone. The algorithm is tailor-made for reverse engineering of gene regulatory networks, converging on the optimal number of nodes for network inference. It could be shown that the networks inferred from cross-species merges are superior to the ones obtained from the single datasets alone in terms of both sensitivity, specificity, accuracy, and the number of comprised network motifs. Not being restricted to two datasets, it offers the opportunity to merge arbitrary numbers of datasets in order to reliably infer large common gene regulatory networks across species (Moghaddas Gholami, 2010).

6. Conclusion

Unlike only a few decades ago, nowadays biology is a quantitative science. With the advent of systems biology, it is now at the verge of formalizing properties of living systems, modelling systems behaviour, and reliable predictions. Exploiting the already large number of high-throughput biological datasets will considerably contribute to this end. Interpretation of high-throughput biological data is facilitated by integration of heterogeneous data. Differences result from different technical platforms for data acquisition, highly diverse fields of research, and different species. A multitude of algorithms is needed for integration, causing the additional problems of different user preferences (of methods) and programmer preferences (of languages). This poses the question if a larger software package can be developed in a user-friendly manner in an academic setting at all, and if MATLAB is the right programming language for this task in particular.

The M-CHiPS project provides prove of principle that a larger software system can be developed and maintained in an academic setting. It is user-friendly, received grant money for development of a commercially distributable prototype and was awarded a price from a business plan award. It comprises many novel approaches. Its database structure has been patented (granted US patent). Analysis algorithms operating the database, implemented and constantly amended over the years by a small team of scientists, have been thoroughly tested by more than 80 users (also scientists). Being a server-based solution (SAAS), it frees the user from installation, update, database administration, or any maintenance. M-CHiPS is predominantly coded in MATLAB.
Its MATLAB code is compiled, running without a MATLAB license for any number of users. R, Bioconductor, Perl, C (database interface as mex files), SQL, as well as all the required Linux libraries are installed such that they seamlessly work together. The installation (which would otherwise represent considerable work) is being available as a whole in form of virtual machines. For flexible allocation of computing resources, web server, file server and database server are separate. It is e.g. possible to run three calculation servers on different machines together with the same database server, or also to run all services on one and the same machine. As PostgreSQL is quickly installed and because different versions of backend and client are uncritical as long as the difference is not too large, it is a good compromise to put database server as well as file server "bare metal" while using the calculation server and the database server as virtual machines.
Experienced administrators will have no difficulties also setting up an Apache web server and installing the packages needed for database acess of the Perl/CGI scripts. However, in our experience performance decrease by virtualization is negligible, both for web and calculation services. In stark contrast to e.g. sequence analysis and protein identification in mass spectrometry, the interactive process of high-throughput quantification hardly provides any perceivable delay, anyway. Therefore, installing web and in particular calculation servers directly on a machine is certainly not worth the effort. Packed into virtual machines, the advantage of having at hand various programming languages for swiftly amending the package does not interfere with its distribution. Combining the wealth of high-throughput biological data statistics prevalently available in R with systems biology tools in MATLAB as well as Perl, Java, SQL and other languages, it satisfies the needs of users and programmers alike and can thus serve as a communications platform both for sharing data and algorithms.

7. Acknowledgement

I am indepted to Martin Vingron, Tim Beißbarth, Christian Busold, Dieter Finkenzeller, Liang Chunguang, Meju Kaipparettu, Radhika Upadhyayula, Amit Agarwal, Mahesh Visvanathan, Corinna Schmitt, and Amin Mohaddas Gholami for contributing with code, further to Martin Vingron, Benedikt Brors, Jürgen Dippon, and Stefan Winter for contributing with ideas.
I wish to thank Jörg Hoheisel, Nicole Hauser, Judith Boer, Marcel Scheideler, Susanne Grahlmann, Frank Diehl, Verena Aign, Andrea Bauer, Helene Tournu, Arno Meijer, Luis Lombardia, Manuel Beccera, Andy Hayes, Albert Neutzner, Nikolaus Schlaich, Tamara Korica, Boris Beckmann, Melanie Bey, Claudia Hartmann, Diana Stjepandic, Kerstin Hasenpusch-Theil, Marcus Frohme, Britta Koch, Marc Hild, Fabian Kruse, Marc Valls, Julia Klopp, Olaf Witt, Kerstin Böhm, Christian Weinel, José Pérez-Ortín, Matthias Nees, Arnaud

Lagorce, Melanie Bier, Sonja Bastuck, Gaelle Dubois, Yana Syagailo, Jan Wiemer, Wladislaw Kusnezow, Patrick Kuery, Daniel Wenzler, Andrea Busch, Sarah Marsal Barril, Rainer Schuler, Benjamin Altenhein, Angela Becker, Li Zheng, Stefanie Brems, Rocio Alba, Bo Xu, Duc van Luu, Iana Siagailo, Martina Brachold, Klaudia Kuranda, Enrico Ragni, Viola Denninger, Martin Brenndoerfer, Stephane Boeuf, Karoliina Pelttari, Eric Steck, Michaela Burkhardt, Michaela Schanne, Prachiti Narvekar, Sandra Bloethner, Anja Boos, Sven Christian, Yi-Ping Lin, Christoph Schröder, Linda Ottoboni, Frank Holtrup, Mahmoud Youns, Rafael Queiroz, Rosa Hernandez-Barbado, Daniela Albrecht, Neeme Tönisson, Raphael Martinez, Christopher Lößner, Florian Haas, Li Lu, Sheng-Jia Zhang, Jorge Soza-Ried, Corinna Wagner, Fiona Pachl, and Zhixiang Wu for contributing with data and testing.
I am thankful to the NGFN program of the German Federal Ministry of Education and Research as well as to the Department of Plant Physiology of the Ruhr University in Bochum for funding. Support for the development of the software platform M-CHiPS was provided by the Helmholtz Association.

8. References

Alon, U. (2007). Network motifs: theory and experimental. *Nature Reviews Genetics,* Vol.8, No.6, (June 2007), pp. 450-461, ISSN 1471-0056

Basso, K.; Margolin, A. A.; Stolovitzky, G.; Klein, U.; Dalla-Favera, R.; & Califano, A. (2005). Reverse engineering of regulatory networks in human B cells. *Nature Genetics,* Vol.37, (March 2005), pp. 382-390, ISSN 1061-4036

Beißbarth, T.; Fellenberg, K.; Brors, B.; Arribas-Prat, R.; Boer, J. M.; Hauser, N. C.; Scheideler, M.; Hoheisel, J. D.; Schuetz, G.; Poustka, A.; & Vingron, M. (2000). Processing and quality control of DNA array hybridization data. *Bioinformatics,* Vol.16, No.11, (June 2000), pp. 1014-1022, ISSN 1367-4803

Bolstad, B. M.; Irizarry, R. A.; Anstrand, M.; & Speed, T. P. (2003). A comparison of normalization methods for high density oligonucleotide array data based on variance and bias. *Bioinformatics,* Vol.19, No.2, (January 2003), pp. 185-193, ISSN 1367-4803

Brazma, A.; Parkinson, H.; Sarkans, U.; Shojatalab, M; Vilo, J.; Abeygunawardena, N.; Holloway, E.; Kapushesky, M.; Kemmeren, P.; Garcia Lara, G.; Oezcimen, A.; Rocca-Serra, P.; & Sansone, S. (2003). ArrayExpress—a public repository for microarray gene expression data at the EBI. *Nucleic Acids Research,* Vol.31, No.1, (January 2003), pp. 68-71, ISSN 0305-1048

Busold, C. H.; Winter, S.; Hauser, N.; Bauer, A.; Dippon, J.; Hoheisel, J. D.; & Fellenberg, K. (2005). Integration of GO annotations in Correspondence Analysis: facilitating the interpretation of microarray data.. *Bioinformatics,* Vol.21, No.10, (March 2005), pp. 2424-2429, ISSN 1367-4803

Cleveland, W. S. (1979). Robust Locally Weighted Regression and Smoothing Scatterplots. *Journal of the American Statistical Association,* Vol.74, No.368, (December 1979), pp. 829-836, ISSN 0003-2638

Culhane, A. C.; Perrière, G.; & Higgins, D. G. (2003). Cross-platform comparison and visualisation of gene expression data using co-inertia analysis. *BMC Bioinformatics,* Vol.4, (November 2003), pp. 59ff, ISSN 1471-2105

Eisen, M. B.; Spellman, P. T.; Brown, P. O.; & Botstein, D. (1998). Cluster analysis and display of genome-wide expression. *Proceedings of the National Academy of Sciences,* Vol.95, No.25, (December 1998), pp. 14863-14868, ISSN 0027-8424

Fellenberg, K.; Hauser, N. C.; Brors, B.; Neutzner, A.; Hoheisel, J. D.; & Vingron, M. (2001). Correspondence Analysis Applied to Microarray Data. *Proceedings of the National Academy of Sciences,* Vol.98, No.19, (September 2001), pp. 10781-10786, ISSN 0027-8424

Fellenberg, K.; Hauser, N. C.; Brors, B.; Hoheisel, J. D.; & Vingron, M. (2002). Microarray data warehouse allowing for inclusion of experiment annotations in statistical analysis. *Bioinformatics,* Vol.18, No.3, (March 2002), pp. 423-433, ISSN 1367-4803

Fellenberg, K. ; Vingron, M. ; Hauser, N. C. ; & Hoheisel, J. D. (2003). Correspondence analysis with microarray data, In: *Perspectives in Gene Expression,* K. Appasani, (Ed.), 307-343, Eaton Publishing, ISBN 978-1881299165, Westboro, MA

Fellenberg, K.; Busold, C. H.; Witt, O.; Bauer, A.; Beckmann, B.; Hauser, N. C.; Frohme, M; Winter, S.; Dippon, J.; & Hoheisel, J. D. (2006). Systematic interpretation of microarray data using experiment annotations. *BMC Genomics,* Vol.7, (December 2006), pp. 319ff, ISSN 1471-2164

Gonzalez, R.; Masquelier, B.; Fleury, H.; Lacroix, B.; Troesch, A.; Vernet, G. & Telles, J. N. (2004). Detection of Human Immunodeficiency Virus Type 1 Antiretroviral Resistance Mutations by High-Density DNA Probe Arrays. *Journal of Clinical Microbiology,* Vol.42, No.7, (July 2004), pp. 2907-2912, ISSN 0095-1137

Huber, W.; von Heydebreck, A.; Sültmann, H.; Poustka, A.; & Vingron, M. (2002). Variance stabilization applied to microarray data calibration and to the quantification of differential expression. *Bioinformatics,* Vol.18, Suppl. 1, (March 2002), pp. S96-104, ISSN 1367-4803

Moghaddas Gholami, A. & Fellenberg, K. (2010). Cross-species common regulatory network inference without requirement for prior gene affiliation. *Bioinformatics,* Vol.26, No. 8, (March 2010), pp. 1082-1090, ISSN 1367-4803

Schanné, M.; Bodem, J. ; Gerhold-Ay, A.; Jacob, A.; Fellenberg, K.; Kräusslich, H.G.; & Hoheisel, J. D. (2008). Genotypic resistance testing in HIV by arrayed primer extension. *Analytical and Bioanalytical Chemistry,* Vol.391, No.5, (July 2008), pp. 1661-1669, ISSN 1618-2642

Smyth, G. K. (2005). limma: Linear Models for Microarray Data, In: *Bioinformatics and Computationtional Biology Solutions Using R and Bioconductor,* R. Gentleman, V. J. Carey, W. Huber, R. A. Irizarry, & S. Dudoit, (Eds.), 397-420, Springer, ISBN ISBN 978-0-387-25146-2, New York

Tusher, V. G.; Tibshirani, R., & Chu, G. (2001). Significance analysis of microarrays applied to the ionizing radiation. *Proceedings of the National Academy of Sciences,* Vol.98, No.9, (April 2001), pp. 5116-5121, ISSN 0027-8424

Zhang, B.; VerBerkmoes, N. C.; Langston, M. A.; Uberbacher, E.; Hettich, R. L., Samatova, N.F. (2006). Detecting differential and correlated protein expression in label-free shotgun. *Journal of Proteome Research,* Vol.5, No.11, (November 2006), pp. 2909-2918, ISSN 1535-3893

Applications of MATLAB in Science and Engineering

Edited by Prof. Tadeusz Michalowski

ISBN 978-953-307-708-6

Hard cover, 510 pages

Publisher InTech

Published online 09, September, 2011

Published in print edition September, 2011

The book consists of 24 chapters illustrating a wide range of areas where MATLAB tools are applied. These areas include mathematics, physics, chemistry and chemical engineering, mechanical engineering, biological (molecular biology) and medical sciences, communication and control systems, digital signal, image and video processing, system modeling and simulation. Many interesting problems have been included throughout the book, and its contents will be beneficial for students and professionals in wide areas of interest.