U N I V E R S I T Y O F T A R T U
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE
Institute of Mathematical Statistics

Raivo Kolde

Co-expression queries across multiple experiments

Master's thesis (40 cp)

Supervisors: Sven Laur, D.Sc. (Tech.)
Jaak Vilo, Ph.D.
Jüri Lember, Ph.D.

TARTU 2008

Contents

Chapter 1

Introduction

Advances in technology have changed the face of molecular biology. Sequencing the genomes of yeast, mouse and human has opened perspectives and introduced a new paradigm for doing research. The high-throughput technologies allow measuring thousands of effects simultaneously and do not require very specific hypotheses, rather they are generated as the outcome of experiment. One of the most used and universal new technologies is the gene expression microarray. Given a sample of cells, it measures the activity or expression of all the genes in the sample. The procedure generates a myriad of data and a wealth of statistical methods have been created to analyze and visualize the end results. Standardization of the experimental procedures and accumulation of has opened up new opportunities. It is possible to use old data to answer new questions, since so many experiments have been done already. This line of research combines and explores the publicly available datasets and is called gene expression meta-analysis.

In some sense this thesis also deals with meta-analysis. The aspect we are interested in is co-expression of genes. Co-expression means that the genes respond to different stimuli in similar ways and therefore have similar expression profile. The "guilt by association" principle states that co-expressed genes display increased likelihood of functional relation. Our idea is that by using multiple microarray experiments in finding co-expressed genes we can get more relevant results. More biological conditions are covered and more data gives us increased power to distinguish between random and systematic similarity in expression. Several studies have have already confirmed the utility of the idea [HMJ+00, ZMC+04, WKB05, LHS+04, SSKK03]. Despite all those studies there are very few publicly available tools for performing similarity search over multiple datasets simultaneously.

Our goal was to build such tool. However, we had to solve several methodological questions about similarity search. In particular we had to develop statistically sound methods for data aggregation and for detecting the significance of the results. We decided to do the similarity search on every dataset in parallel and integrate the search results. This leaves us with a rank aggregation problem. The most important contribution of the thesis is a novel rank aggregation algorithm that is designed to work on noisy situations. We test the performance of the algorithm on both real and simulated data. Large part of the thesis is dedicated to solving practical questions that emerged, when using the algorithm. One problem is constraining the number of results. The aim is to show only results that are significantly better than random. However, this is not enough, the power to distinguish results from random case is growing very fast, when increasing the number of datasets and we end up with too many results. Therefore, we propose two schemes for constraining the number of results consistently over different settings. In addition, we propose the way to select relevant datasets for the similarity search, to lessen the noise. As one result of the thesis, we present the similarity search tool itself. There we have implemented most of the methodology that we have developed within this thesis.

Chapter 2

Introduction to gene expression microarrays

Living cell is a very complicated biochemical system. Although it has been studied extensively, the knowledge about its exact workings are sparse. The main building blocks of the cell are *proteins*. In addition to forming the structure of the cells, they have a wide variety of functions: catalysis, immune recognition, cell adhesion, signal transduction, sensory capabilities, transport, movement and cellular organization. Despite of all the diversity in size, shape and function, all the proteins are built the same way from the same building blocks. Chemically the proteins are strings of *amino acids* that are folded into different structures. Only twenty different amino acids are used in producing proteins. All the information about the amino acid sequences is coded into DNA. Thus, DNA is a blueprint for all living organisms.

The *DNA* is a long spiral shaped molecule, which consists of two complementary strands of nucleotides. There are 4 different nucleotides: adenine, guanine, thymine and cytosine. These nucleotides are usually denoted by roman letters A, G, T and C respectively. One strand of DNA can be represented then as a string encoded with these 4 letters. The order of nucleotides on the other strand is determined by the sequence in first one, it is said that the two strands are complementary. Nucleotides form pairs with the ones on other strand, where A is always paired up with T and C with G. It is known that each amino acid is coded by a triplet of nucleotides on DNA. The parts of DNA that code a protein are known as genes. Humans have about 20000 protein coding genes [CFK+07].

To make the proteins, the information from DNA is first transcribed into a smaller RNA molecule that is later used as a template for producing proteins. The RNA is a simpler version of DNA that has only one strand and one of the nucleotides thymine is substituted with uracil. Although the RNA is single stranded, it can still bind to a singe stranded complementary DNA molecule. This property is essential in many cell processes and experimental methods. The process of synthesizing a complementary RNA molecule based on DNA is called *transcription*. The RNA that is transcribed from a gene is called *messenger RNA* or mRNA. The mRNA molecules are transported out of the nucleus, where protein complexes called ribosomes attach to it and start translating the sequence of RNA into protein.

2.1 Gene expression and microarrays

The process of producing proteins from genes is called *gene expression*. Not all genes are expressed at once, it depends on the proteins needed currently in the cell. Therefore, knowing the genes expressed in the cell would give a valuable characterization about the state of the cell. By comparing the the expression in several conditions we could find the molecular differences between them. The most exact way to quantify gene expression would be measuring the number of different proteins in the cell. However, this is very complicated on large scale due to the large variety of protein structures and physical characteristics. Some solutions exist for this task, but right now they are exceedingly expensive.

However, it is much easier to measure the number of different mRNA molecules in a cell. With mRNA it is possible to use the fact that it binds to complementary DNA molecules. Therefore, we can capture basically any mRNA molecules with complementary single stranded DNAs and estimate their abundance in the cell. Even more importantly, we can capture different mRNA molecules in parallel. The technology for this is called *gene expression microarray* or *gene chip*.

A gene expression microarray can measure the expression of all genes in the genome simultaneously. For each gene a batch of millions complementary DNA sequences are synthesized, each batch is printed to a spot on a glass slide. as a result, each gene is measured by one or more spots on an array. Before using the array, the mRNA is extracted from the biological sample and labeled with a fluorescent marker. The solution containing labeled mRNA molecules is put on the glass slide, where all the spots start capturing corresponding molecules. After some time, pictures are taken of the array. The spots that captured more material should be brighter on the picture due to the radioactive labeling of mRNA molecules. An example of a picture can be seen on Figure 2.1a. The numerical values of gene expression are inferred from the brightness of the spots. The end result of a microarray study that is using several gene chips is a gene expression matrix, a schematic illustration of it can be found on Figure 2.1b.

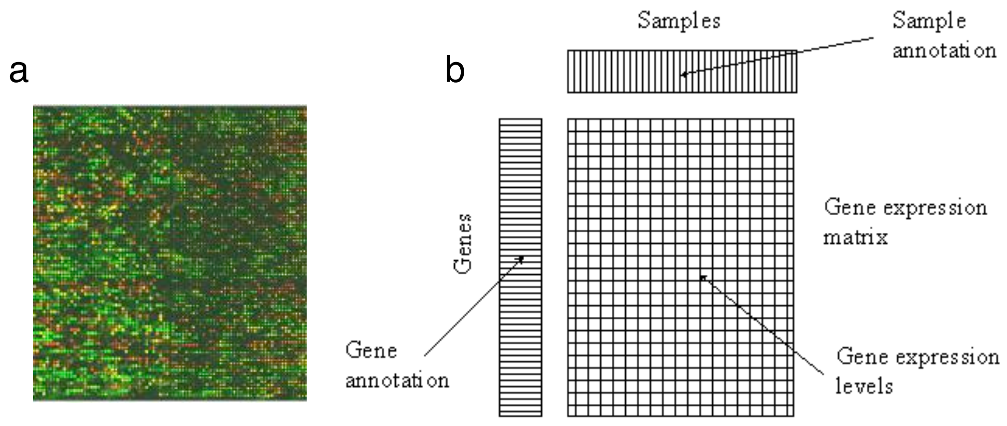


Figure 2.1: A picture of microarray together with a schematic representation of gene expression matrix.

The process of getting expression matrix out of a set of microarray pictures is not trivial. Depending on the array type and design, the workflow is different. In some cases, complicated image recognitions algorithms are needed. Often there is a need for extensive statistical corrections. The intensity of a spot depends on several factors besides the amount of specific mRNA on the solution. By transforming the expression values, most of the problems can be eliminated to the point that different arrays become comparable. Whole process of producing expression matrix from image files is called normalization. There are several different possibilities for normalization, but in this thesis we ignore the issue and assume we already have an expression matrix.

2.2 Gene expression analysis

The gene expression microarrays are used in wide variety of situations. The most simple experimental design compares the gene expression in two biological conditions. For example, biologists often compare normal and cancer tissue. In this analysis, the main goal is to find genes that are differently expressed between the conditions. These genes can be used later as diagnostic markers or functionally studied further. In the pipeline of a study, the microarrays are often used for generating hypotheses and finding candidate genes for further experiments.

For finding the genes that are differentially expressed between two conditions, people usually use simple statistical tests, such as t-test. The test is performed on each gene separately. The more complicated part is deciding the significance of the test results. Large number of tests can introduce easily false positives if common significance threholds are used. Therefore, the test results need to be corrected for multiple testing at first. In the early days of microarrays the many statisticians dealt with the multiple testing problems. Several techniques, such as false discovery rate (FDR) and empirical Bayes methods owe their popularity to the advent of microarrays. The experimental design can be more complicated, we may have more information about the samples. For example we can measure the expression on different patients with some disease. In these cases the usual linear models have proven to be very useful.

Another statistical methodology that has relived its renaissance thanks to microarrays, is cluster analysis. As the datasets are usually huge in normal standards, then almost only way to visualize the data is by clustering. Either we divide the data into smaller groups by k-means style clustering or draw a dendrogram of hierarchical clustering. A clustered heatmap of a microarray data has become the symbol of modern molecular biology.

Depending on the particular question many different unsupervised and supervised learning methods have been applied to the microarray data. For example, early studies have shown that it is possible to recognize subtypes of cancer based on the microarray profile [BRS$^+$01]. Many popular classification methods, such as support vector machienes, linear discriminants and neural networks, have been employed to solve on the task [FLNP00, BGL$^+$00, THNC02]. A lot of effort has put into merging the microarray data with other kinds of experimental data. For example, kernel based approaches and graphical modeling has been tried [TDO$^+$03, BTvOM07].

One of the great promises of microarray data is the ability to predict functional relations between genes and maybe even annotate functionally uncharacterized genes. The idea is to use co-expression for these tasks. If two genes behave in similar fashion then it might mean several things. First, the genes are regulated by the same mechanisms, the genes interact with each other or just participate in same processes. In summary, if two genes are co-expressed then it is more likely that they have some functional relations. This principle is called *Guilt by association* and its validity is proved several times [HMJ$^+$00, ZMC$^+$04, WKB05, LHS$^+$04, SSKK03]. This principle can be put into use in several ways. It can be used on its own to predict gene function by looking at the co-expressed genes. Another possibility is to combine it with other types of evidence to strengthen some hypotheses, Some examples of this can be found in [LDB$^+$06, KvBV$^+$02].

With the accumulation of microarray studies a new trend in research has emerged. It is possible to use old data to answer new questions. This approach is called *meta-analysis of gene expression*. For example, scientist have been trying to find some disease specific gene by merging multiple publicly available datasets together [RKSM$^+$07, SFKR04]. The usage of multiple datasets also gives more power to distinguish the signal from noise.

In this thesis we concentrate on the co-expression queries over multiple datasets. We are trying to build a tool that puts *Guilt by association* principle in work and borrows its power from the multitude of expression datasets used.

Chapter 3

Similarity search methods

In this chapter, we introduce and formalize the most important mathematical concepts and techniques that are used in in this thesis. In our tool, we want to find genes that are often co-expressed with a query gene. In addition, we want to identify the experiments where the genes were co-expressed.

In our setting, we perform the similarity search separately in every gene expression experiment and later combine the resulting similarity rankings into final co-expression ranking. This approach reduces the similarity search into the rank aggregation problem. The problem of combining rankings is old and has been studied in different contexts, like social choice theory and information retrieval [DKNS01, Saa99]. We give an overview of most important methods that are used to solve this problem. However, since none of the methods fits perfectly to our setting, we propose an algorithm of our own and test its applicability on simulated data.

3.1 Problem statement

Co-expression is a strong indicator of functional similarity between genes. Finding co-expressed genes on only one microarray study cannot uncover all the connections. By introducing more experiments and widening the spectrum of biological conditions, we have more power to find the co-expression links. Our goal is to build a query engine that finds similar expression profiles over many microarray experiments.

As a result, we have a situation, where the input data consists of several expression matrices and each of them corresponds to specific experiment. In the simplest case all the experiments are on the same microarray platform. That means, all the datasets contain the measurements for the same set of genes. Which makes any combination of experiments relatively easy. Using different microarray platforms introduces several technical difficulties. Different platforms measure the expression of different sets of genes. Then in merging the data, we have to introduce some missing values, since we do not have measurements for all the genes. Conversely, we can throw out all the genes that are not present in all the arrays, but this might mean a huge loss of data. Even more technical difficulties will arise, when we used data from different organisms. In addition to the problem of different subsets of genes, it is not trivial to if determine the identical genes between organisms.

In scope of this masters thesis, we develop methods only for the simplest case, where all he experiments are on the same platform. At the same time we keep in mind the more complicated settings, such as different platforms and organisms, so the methods established in here, are relatively easy to extend to the more complex query types.

Our goal is to find genes with similar expression to query gene. The most obvious solution is to put all the microarrays to the same matrix and perform the similarity search over this large matrix. However, this approach has several shortcomings. An experiment is a set of microarrays, answering specific biological question, prepared in one laboratory by the same people. Putting all the microarrays to the same matrix can introduce some unwanted variation. For example, very strong lab effect on a microarray study has been shown [IWS$^+$05]. in other words, even if results are consistent within one study, then merging two similar studies can introduce biologically meaningless artefacts. Also, the microarray normalization methods are going to start distorting the expression values, if the biological conditions are too different. Most importantly, it is quite difficult to generalize this single matrix approach to the more complicated settings. If the sets of genes are different between arrays, then we cannot just put the arrays together.

Therefore, we take a different approach and keep the experiments separated. We have to do the similarity search in every one of the experiments, to estimate the global similarity between two genes. Given the results of these queries, we have to find a way how to combine them into final list of co-expressed genes. The actual similarity scores from the queries depend on several factors. Most importantly, on the choice of co-expression measure. We do not want to restrict the choice of the metric to one alternative. Different hypotheses require different similarity measures. Another factor influencing the actual score values is the size and design of the experiment. Often some large datasets contain many redundant experiments, but smaller datasets measure very different conditions. So comparing numeric similarity scores from different experiments can be quite difficult task. To avoid pitfalls of direct aggregation of similarity scores, we use ordering information instead.

More formally, we sort the genes by similarity to the query gene in each dataset. For this, we may use any co-expression measure available, for example correlation, absolute correlation or Eucleidean distance. As a result, we end up with a different similarity order of genes for every experiment.

Consequently, we have to find a way to combine the single permutations of genes into a final order. We expect the method to reward genes that come out on the top of the lists more often than not. Another characteristic of our system is that queries in many datasets will produce spurious results. It might be that a gene is just not used in the specific biological conditions or its expression has no meaningful changes. In this case, sorting by similarity produces a random permutation of genes. Thus, the method for aggregating lists has to be able to cope with this noise. Also, it is important that the method can highlight the genes with significant co-expression to query gene. Significant in this context means that the gene is so close to query gene in so many datasets, that this is not very likely to happen if rankings were random.

3.2 Rank aggregation

We have reduced the problem of finding co-expressed genes to integrating similarity rankings. The problem is not new, usually it is called *rank aggregation*. The purpose of rank aggregation is to combine many different rank orderings on the same set of candidates to obtain a better ordering. Rank aggregation arises from several contexts. For example, it is studied thoroughly in theory of social choice and voting [Saa99]. In this context, the task is to find correct ranking of candidates if each voter ranks the candidates in order of preference. For instance, we might try to order athletes based on several competitions. The problem emerges also in context of information retrieval when combining search results from different databases or search engines.

Formally the rank aggregation task is defined as follows. We have $n \times m$ matrix of rankings R. Rows of R represent instances g_i that are being ranked, in our example genes, in case of voting, candidates. Let us denote the i-th row of R as R_i. Each column of R is a ranking vector produced by a data source or voter D_j. Let us denote the j-th column of R as R^j. The ranking vector is a permutation of numbers from 1 to n, the number of instances to be ranked. The rank aggregation can be formalized as a function of the matrix R

$$\bar{R} = \mathsf{RankAgg}(R), \tag{3.1}$$

where $\bar{R}$ is in some way optimal ranking vector.

To introduce some statistical methods, it is useful to normalize R to range $(0, 1)$. We can do it by simple division

$$R_{\text{normalized}} = \frac{1}{n} R, \tag{3.2}$$

where n is the number of rows in R and therefore, the maximal rank value on every column. From now on we consider rank matrices that are already normalized.

3.3 Rank aggregation methods

The first rank aggregation methods are from late 18th century. The first was *Borda count*, which for every instance g_i sums up values in the vector R_i and orders the candidates based on the sums [Bor81]. This is equivalent of taking an arithmetic mean of rankings for every g_i. One can suggest several similar ideas. For example, we can use median or different types of averages, like geometric or harmonic. Which one of them is the best depends largely on the setting and what we try to achieve.

Nevertheless, it is possible to define optimality of rank aggregation. *Condorcet criterion* looks at every pair of candidates and determines a winner between them, by counting which one was preferred by most voters. The overall winner is the one, who wins every individual matchup [dC85]. For example, $x > y$ if x has larger ranking than y in majority of datasets. Condorcet criterion is an optimality benchmark that different rank aggregation methods should satisfy. For example Borda count does not satisfy Condorcet criteria, which is often considered as a major shortcoming of it.

Another natural optimality criterion for rank aggregation methods was proposed by Kemeny [Kem59]. The *Kemeny optimal ordering* is a ranking that minimizes distances to all the input rankings. Formally the Kemeny optimal ordering is

$$\bar{R} = \arg\min_{\hat{R}} \sum_{j=1}^{m} d(R^j, \hat{R}), \tag{3.3}$$

where d is distance measure between two ranking vectors. By minimizing the distances to all input rankings, the Kemeny optimal ordering represents in a sense a best "compromise". The distance measure used for comparing the lists is *Kendall tau* [KEN38]. Which is defined as

$$K(\sigma, \rho) = |(i, j) : \sigma_i < \sigma_j \text{ and } \rho_i > \rho_j|, \tag{3.4}$$

where i and j are the objects to be ranked and σ and ρ are rankings. $K(\sigma, \rho)$ counts the pairs that have different rank order in σ and ρ. It has been shown that finding Kemeny optimal ordering with Kendall tau is NP-hard, where the size of instance is measured as the number of data sources [BTT89]. The NP hardness holds also with regards to the number of instances to be ordered [DKNS01]. In our case, where we might have over 100 data sources and over 20000 genes to order, the rank aggregation with this method would be computationally very expensive.

Another distance measure that has been used is *Spearman footrule* [DKNS01]. Spearman footrule is defined as follows:

$$F(\sigma, \rho) = \sum_{i=1}^{n} |\sigma_i - \rho_i|. \tag{3.5}$$

Using Spearman footrule for finding optimal ordering is computationally less expensive. It is also known [DG77], that Spearman footrule approximates Kendall tau within the factor of two:

$$K(\sigma,\rho) \leqslant F(\sigma,\rho) \leqslant 2K(\sigma,\rho). \tag{3.6}$$

So by using Spearman footrule in finding Kemeny optimal ordering, we get a good approximation for the Kendall tau version. In [DKNS01] it is shown that optimal ordering using Spearman footrule can in turn be approximated by ordering the instances by using median of their ranks, which is even less expensive computationally.

Another interesting approach is to use Markov chains in rank aggregation. It was also proposed in [DKNS01]. Those methods construct a Markov chain based on the ranking matrix. The states of the Markov chain are the instances we want to order. The transition probabilities between the states are calculated using the ranking matrix. The idea is to give higher probability for transitions from states that occur lower in the lists to states that occur higher.

For example the simplest algorithm for building the transition matrix is as follows: if the current state is an instance i, then the next state is chosen uniformly from the multiset of all instances that were ranked higher than (or equal to) i by some ranking R^j, that is, from the multiset $\cup_{j=1}^{m}\{l : R_l^j \leqslant R_i^j\}$. That means, the Markov chain moves from states that are usally ranked lower to the the states that are ranked higher. Therefore, we have higher probability to end up in the states that appear more in the top. Thus, for final ordering the stationary probabilities of the Markov chain are found and the states are ordered by them. In [DKNS01] several different options for constructing the Markov chains are proposed, each corresponding to different heuristics and goals.

Current developments on the field of rank aggregation rely mostly on the methods described above. From the voting theory point of view, important questions are related to "correctness" of the aggregated ranking [Saa99]. There are several "voting paradoxes", where different methods do not agree and it is not clear what is the correct solution.

In computer science the focus is more on modifing the methods, to become applyable in a variety of real world situations. One of the issues is of course speed of the algorithms, this question is addressed in [DKNS01]. The rank aggregation methods are commonly used in web meta-search, where search results are combined from different search engines. In this case, the number of all web pages is enormous and it is not reasonable to rank them all, so the algorithms can use only the top results from each engine. So the question becomes, how to combine incomplete rankings. Approaches to solve this problem can be found in [FKS03, Scu07].

3.4 Rank aggregation with outlier detection

In context of voting, all rankings are meaningful. The same applies for web meta-search. But in our setting this is not true. In case of most queries, we expect many datasets to produce arbitrary rankings. This fact shifts the emphasis from finding optimal rank aggregation method to finding method robust enough.

Therefore, we consider a different setup. Namely, we assume that the ranking vector R_i for gene g_i is a sample from mixture distribution. Data sources where is no co-expression between query gene and g_i produce rankings from uniform distribution. When there is co-expression then the rankings are from distribution skewed heavily towards 0. Given this model, the rank aggregation method should identify genes thats ranking vectors contain most observations from the skewed distribution.

If we have method that aggregates rankings as desired, then another question arises: how many of the top genes are really important? With the model we have described both the case we are interested in and the one we are not. With any method, we have to find a way to discriminate between relevant and random results. Further on, if the important results are identified, then it is interesting to see, which data sources were responsible for them. in other words, which type of experiments the co-expression occurred. The latter gives hints about the nature of the connection and reveals datasets that characterize this process most appropriately. This is what we call *rank aggregation with outlier detection.*

3.5 Rank aggregation with order statistics

On given assumptions a rank aggregation method was proposed in [SSKK03]. The article studied co-expression networks over several species. The authors performed the co-expression query on every organism one by one and combined the rankings from the queries into one result. The same method was used in [ALML06], where the authors use different types of evidence to associate the genes with diseases. The method was used to merge the results.

The idea is to use the joint cumulative distribution function of order statistics. Let us denote the ranking vector R_i with $r = (r_1, \ldots, r_m)$, the ranks are already normalized to range $(0, 1)$. Given the non-informative case, our null hypothesis H_0, then the vector is an *i.i.d* sample from uniform distribution $U(0, 1)$. If we order the values of r increasingly, we get a vector of order statistics $r = (r_{(1)}, \ldots, r_{(m)})$. We are interested, if the values are skewed towards 0, compared to uniform case. Thus, we can calculate the probability that each $R_{(j)}$ is so small or smaller if the sample would be from uniform distribution, which is the joint cumulative distribution function of the order statistics. The corresponding probability is

$$F(r_{(1)}, \ldots, r_{(m)}) = \mathbf{P}[(s_1, \ldots, s_m) : s_1 \leqslant r_{(1)}, \ldots, s_m \leqslant r_{(m)}],$$

with the constraint

$$s_1 \leqslant \ldots \leqslant s_m.$$

The integral for calculating the probability is

$$F(r_{(1)}, \ldots, r_{(m)}) = m! \int_0^{r_{(1)}} \int_{s_1}^{r_{(2)}} \ldots \int_{s_{m-1}}^{r_{(m)}} ds_1 ds_2 \ldots ds_m. \tag{3.7}$$

The $m!$ is the number of different rank vectors, that will give the same ordered rank vector. This score is quite natural way to measure the skewness towards 0. At first glance, it even appears to produce p-values, and so they claim also in [SSKK03]. In deciding the significance of the scores, they used standard multiple testing machinery. However, these probabilities are not p-values, since p-value is properly defined only on univariate statistics, but in this case the statistic is a vector of rankings. Also p-value should have uniform distribution under H_0. However with this score simulations show strong skewness towards 0. Therefore, using standard multiple testing corrections for deciding the significance of the score, might yield lot of false positives. So care should be taken when interpreting the scores. For example in [ALML06] the score was converted to p-values using simulations under null hypothesis.

The other problem with the score is, that it is computationally expensive to calculate. In original article the authors offered an algorithm for it, but its complexity is $O(m!)$. That was enough for their purposes but unfeasible for us, since we might have more than 100 rankings to aggregate at a time. The authors of [ALML06] used the same score, but found the algorithm impractical and came up with computationally less expensive alternative. If we start calculating the integral (3.7), then pretty soon we can notice how some components of the calculation start to reappear. The new algorithm uses the observation and employes a iterative scheme for calculating F. Let us denote $V_k = F(r_{(1)}, \ldots, r_{(k)})/k!$ and $V_0 = 1$. The V_k basically denotes the result of k-th integration in (3.7). We can use iterative update rule for V_k

$$V_k = \sum_{j=1}^{k} (-1)^{j-1} \frac{V_{k-1}}{j!} (r_{(m-k+1)})^j. \tag{3.8}$$

By the definition of V_k, we can find the end result $F(r_{(1)}, \ldots, r_{(m)}) = m!V_m$. This formula worked perfectly, when we merge moderate number of data sources. However, if the number of data sources grows larger than 40 our naive implementation of the algorithm becomes numerically unstable, due to the alternating terms. In the case of our tool the number of data sources can be much larger than 40. So we still need a solution to our problem.

3.6 BetaMEM method

As the previous method did not suit or needs we had to roll out our own solution. Our assumptions are exactly the same as in previous section. We assume that rankings come from a mixture distribution, where the first component is uniform and second component is skewed. If there is no co-expression, then all the rankings come from uniform component, this is our null hypothesis when deciding the significance of the vector.

Say we have the sample r from uniform distribution $U(0, 1)$. Let us order the values of r increasingly $r = (r_{(1)}, \ldots, r_{(m)})$. For every $r_{(i)}$ we can ask, how probable it is to see i or more rankings smaller than it, if the r is sample from $U(0, 1)$. In case of H_0 the number of elements smaller than $r_{(i)}$ has binomial distribution $B(m, r_{(i)})$. Thus for each $r_{(i)}$ we can calculate binomial p-value

$$\begin{aligned} p_i &= \mathbf{P}(j \text{ or more smaller rankings than } r_{(i)}) && (3.9) \\ &= \sum_{k=i}^{m} \binom{m}{k} r_{(i)}^{k} (1 - r_{(i)})^{m-k}. && (3.10) \end{aligned}$$

Each of those p-values measures deviation towards 0 for corresponding order statistic. Now we find which of the order statistics has the largest deviation from expected distribution and use the corresponding p-value

$$p(r) = \min\{p_1, \ldots, p_m\}, \tag{3.11}$$

as the final similarity score.

The actual calculations of p_i are done using beta distribution quantiles. There is a connection between beta and binomial distribution.

$$F_{beta}(p, a, m - a + 1) = \sum_{j=a}^{n} \binom{n}{l} p^j (1 - p)^{m-j}, \tag{3.12}$$

It means that we can use beta distribution quantiles instead of binomial distribution.

This fact has also a nice explanation. Under H_0 the order statistic $R_i^{(j)}$ is distributed according to $Beta(j, m - j + 1)$. As a result for every order statistic we find, how much the distribution is skewed towards 0 compared to its expectation. Thanks to the connection with beta distribution, we call our method *BetaMEM*.

The score also has an interesting byproduct. When finding the order statistic with minimal p-value, we actually find a separation between data sources that contribute to the final score and the ones that do not. The $r_{(i)}$ that yielded the smallest p-value can be considered as a significance cutoff for rankings. All the rankings that are smaller than the $r_{(i)}$ contributed to the final score, so they can be considered significant. This feature adds a new dimension to the search, not only do we see, that two genes are similar, but we can also guess the nature of similarity, by studying where the co-expression occurs.

The Figure 3.1 illustrates how the score works on a skewed ranking vector. The vector was obtained from mixture of uniform and $Beta(1, 50)$ distributions, 20 values are from the beta distribution that emits very small values. The first histogram shows the distribution of rankings and the clear peak near 0. The second plot shows how the p-value of the order statistics changes and reaches minima near 20. Which confirms our claim that the algorithm is able to identify the datasets where the genes are co-expressed. The third plot shows the actual values in the ranking vector ordered increasingly, the vertical line marks the the spot that yielded minimal p-value.

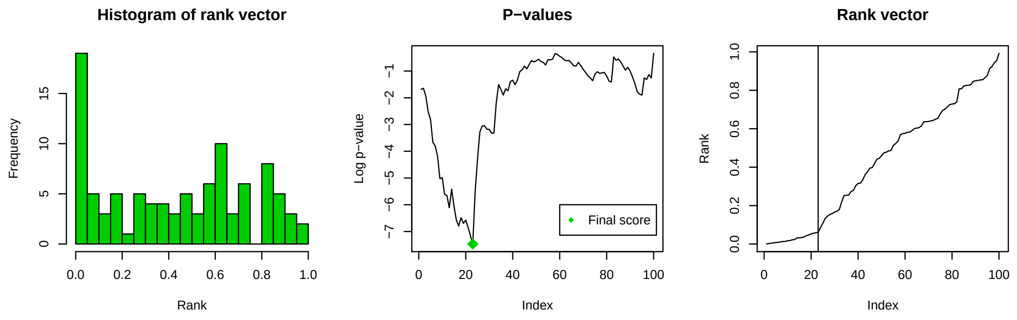


Figure 3.1: An example how BetaMEM method works. The first and third plot show the distribution of the skewed ranking vector. We planted 20 small values in the vector. Second plot shows the individual p-values on log scale calculated for order statistics, the green dot represents the final score. We can see the minima was reached near 20, exactly as it should.

In comparison to the joint cdf score described above, the BetaMEM method is certainly computationally much more feasible: it depends on m linearly. Also, it does not get numerically unstable even for large m values. We cannot say it is a p-value, however in the next chapter we see that we it is still correct use bonferroni multiple testing correction on it. So we have a natural way for deciding on the significance of the p-value.

3.7 Comparison of methods

Finally, we compared three methods on simulated data, to get some insight about the performance of the methods. The goal was to find out how well different methods recognize biased rank vectors. We selected three methods for comparison. From classical voting theory the Borda count, that is just the sum of vector elements. Then the method from [SSKK03], which uses joint cumulative distribution function of order statistics. Finally, our method BetaMEM, which compares order statistics with their expected distribution.

The first step was to estimate the same significance thresholds for all the methods, by simulating the distribution of scores under H_0. We generated a matrix of random rankings by taking 20 random permutations of vector $(1, \ldots, 22000)$ as columns. We chose the number 20, because at that length the algorithm (3.8) was still relatively stable; 22000 is approximately the number of genes on a human gene expression array. For all the rows in the matrix we applied all three scores. For each score, we took 0.01 quantile of the calculated values for a significance threshold.

We used the mixture distribution model, for generating biased rank vectors. The ranking vector contains two kinds of rankings: ones that show real co-expression in the given dataset and ones that do not. The first group has a distribution skewed strongly towards 0, second has uniform distribution.

We considered two scenarios. One with the biased elements very close to 0 other with biased elements having more relaxed distribution. Usually only few hundred top genes from co-expression query are considered to show real expression similarity. For this two scenarios, we use $Beta(1, 200)$ and $Beta(1, 50)$ as the biased mixture components. These distributions have peak at 0 and are quickly decreasing afterwards. If we have 22000 genes on a chip, then the first component produces rankings with an average of about 110, second with 440. So these two scenarios are really close to the situations we want to discover in the real life.

To simulate a wide variety of cases we varied the number of biased elements in the vectors from 1 to 10. For each case we generated 10000 observations. Our algorithm also tries to find the number of biased elements, so we recorded these numbers. Since we know the real numbers we can compare found numbers with the actual result. The results are shown in the Table 3.1.

$Beta(1,200)$	1	2	3	4	5	6	7	8	9	10
BetaMEM	0.94	0.71	0.2	0.01	0	0	0	0	0	0
Joint cdf	0.92	0.68	0.25	0.02	0	0	0	0	0	0
Borda Count	0.98	0.95	0.91	0.82	0.7	0.53	0.35	0.18	0.07	0.02
# sign elements	4.7	3.24	3.48	4.22	5.08	5.99	6.91	7.85	8.78	9.71
sd # sign elements	5.64	3.11	1.72	1.21	0.99	0.9	0.88	0.88	0.89	0.9

$Beta(1,50)$	1	2	3	4	5	6	7	8	9	10
BetaMEM	0.97	0.89	0.7	0.4	0.13	0.02	0	0	0	0
Joint cdf	0.96	0.87	0.66	0.38	0.14	0.02	0	0	0	0
Borda Count	0.98	0.96	0.91	0.83	0.71	0.54	0.38	0.22	0.09	0.02
# sign elements	6.98	5.27	4.96	5.16	5.74	6.46	7.26	8.09	8.96	9.84
sd # sign elements	6.46	4.96	3.71	2.74	2.19	1.87	1.73	1.61	1.56	1.48

Table 3.1: First table presents the more biased scenario and the second less biased. The numbers in the first three rows of the table show the proportion of misclassified observations by different methods. The fourth row indicates the average number of significant elements found by our algorithm, fifth row is its standard deviation. The numbers above columns represent how many small elements were added to the vector.

The results indicate, that both methods based on order statistics perform on similar level. The misclassification rates differ by only few percentage points and one is not clearly better than the other. Borda count, which is equivalent to taking average rank, performed much worse. The bad performance of Borda count can be explained by the fact, that it is not designed to handle systems with high level of noise.

It is interesting to see that the average number of significant elements predicted by our algorithm is in good correlation with the real number of planted elements. The correlation is good for cases that can be reliably classified as biased vectors. In the more relaxed case, the average of significant element estimation remained about right, but its standard deviation grew.

As summary we can say that our method works well. Its precision is on par with the competing method, but its much easier to compute. Additionally the number of significant datasets found agrees nicely with the actual number. So this is our method of choice in the subsequent study.

Chapter 4

Determining the number of co-expressed genes

So far we have defined only a method for ordering genes based on the similarity. Given such order it is natural to ask where to draw the line that identifies the results that show real co-expression. In previous chapter we employed the strategy, where we simulated the score distribution under null hypothesis and fixed the significance threshold based on it. This approach is universal, but not very elegant, since we have to do the simulations with different lengths of rank vector. In this chapter we propose a simpler solution. The BetaMEM method was based on a minimum of a batch of p-values. The minimizing step means that the final score is not a p-value anymore, but actually we can still use some of the multiple testing machinery to find the significance threshold. We introduce the Bonferroni correction and verify its applicability to our setting experimentally.

4.1 Bonferroni correction

Suppose we test several statistical hypothesis simultaneously. Even if the null hypothesis holds on every case, the probability to obtain a very small p-value increases, the more tests we make. Thus, we cannot use α as the significance threshold, if we want to control the probability of false positives on that level. The *multiple testing corrections* aim to adjust the

the significance thresholds, in order to hold the probability of false positives under desired level.

Let $T_1, \dots, T_m$ be a set of hypotheses that we want to test and α the desired level of significance. We want to find a criteria for passing T_i such that in summary

$$\mathbf{P}[\text{some } T_i \text{ passes } |H_0] \leqslant \alpha. \tag{4.1}$$

Using union bound we can write

$$\begin{aligned}\mathbf{P}[\text{some } T_i \text{ passes } |H_0] &= \mathbf{P}[\cup_{i=1}^{m}\{T_i \text{ passes } |H_0\}] \\ &\leqslant \sum_{i=1}^{m} \mathbf{P}[T_i \text{ passes } |H_0], \end{aligned} \tag{4.2}$$

If for every $1 \leqslant i \leqslant m$,

$$P(T_i \text{ passes } |H_0) \leqslant \frac{\alpha}{n}$$

then from (4.2) we can see that (4.1) holds. The latter forms the basis of *Bonferroni correction*. Namely, if we do m tests simultaneously and want to achieve significance level α, then we must decrease the significance threshold to $\frac{\alpha}{n}$ for individual tests. This result holds even if the tests are not independent. As union bound is really crude result for most cases, the Bonferroni correction is considered to be very conservative.

The Bonferroni correction is directly applicable if we want to find significance thresholds for BetaMEM scores. Indeed, the BetaMEM score is calculated as a minimum of m p-values. These p-values are strongly correlated, but the later does not matter for Bonferroni correction. Consequently, we can take significance threshold as $\frac{\alpha}{m}$ for individual score. On every query we calculate the BetaMEM score for each of the n genes. Thus, for a query the Bonferroni threshold is

$$t_\alpha = \frac{\alpha}{mn}.$$

4.1.1 Experimental validation of Bonferroni threshold

Clearly if all assumptions used to define BetaMEM score indeed hold in biological data, then there should be no problem in applying Bonferroni threshold for BetaMEM score. However, the latter is not immediately evident, as we made assumption that if two genes are not related, then the rankings are *i.i.d* sample from uniform distribution. In real life, the data is likely to contain intricate dependancies. Even if our search gene is not co-expressed with any other genes in any dataset, then the rankings from similarity searches can be still correlated, due to relationships between other genes. The latter can cause unexpected behaviour of the BetaMEM scores. To study that, we performed a simulation study.

The idea was to study the behaviour of BetaMEM score, when we know that there is no connection between query gene and other genes. At the same time it is critical that we preserve the cross correlated structure of the datasets. To achieve that situation we randomized only the expression of the query gene, all the other profiles were left intact. The BetaMEM scores obtained this way correspond to our real null hypothesis. Our goal is to show Bonferroni threshold works on this case. If we could show it, then we have obtained a easy procedure for assigning significance to the results.

The exact experiment setup was as follows. We randomized expression of one gene in each dataset, by drawing new values from standard Gaussian distribution. Using this gene for the query, we applied our tool. Since we are interested also the dependance on m, the query is done on $20, 30, \ldots, 140$ randomly selected datasets. The same procedure was repeated 100 times. Next, we took the smallest BetaMEM score from each query, and found out how many of them exceeded the Bonferroni thresholds. The results are presented in Table 4.1.

For both 0.1 and 0.05 threshold the number of queries exceeding the threshold is well within the expected limits, with average number of false positives nearly 2 times smaller than expected. Based on these results, we can clearly use Bonferroni threshold for deciding the significance of the query.

# of datasets	30	40	50	60	70	80	90	100	110	120	130	140	mean
Bonferroni 0.1	0.08	0.09	0.01	0.07	0.06	0.06	0.05	0.07	0.05	0.04	0.06	0.06	0.0583
Bonferroni 0.05	0.04	0.03	0.01	0.01	0.04	0.04	0.00	0.04	0.04	0.01	0.03	0.03	0.026

Table 4.1: The table shows the percentage of randomized MEM queries that exceed the bonferroni threshold. The percentages are based on 100 randomizations.

4.1.2 The number of significant results

Several questions arise given the Bonferroni thresholds and real data. For example, how many significant results one query produces? This number certainly depends on the number of data sources used. By adding datasets, we increase the chance, that more relevant datasets are used in the query. We also increase the sample size for our test, which in turn allows to detect more subtle effects. Thus, the number of significantly co-expressed genes is likely grow. It is important to find out, how many neighbours on average we can expect if we use Bonferroni correction and how strongly it depends on the number of experiments used in the query.

We performed a experiment to study these effects. We selected randomly 500 genes, with each of them we performed 5 different queries: with 5, 10, 20, 40 and 100 datasets, selected randomly. From each of those queries we extracted the number of genes, that had significant scores according to Bonferroni correction on significance level 0.05.

The number of significantly similar genes depends a lot on the number of datasets used for query. On Figure 4.1 the distributions in the case of 5, 40 and 100 data sources are presented. We can see that by increasing the number of data sources, the number of similar genes grows pretty fast. With 5 datasets, most of the queries did not return any significant results, but with 40 and 100, the number of results was mostly in thousands.

Table 4.2 characterizes the rapid increase in the number of results even more comprehensively.

The increase in the number of results is really fast. So fast, that the

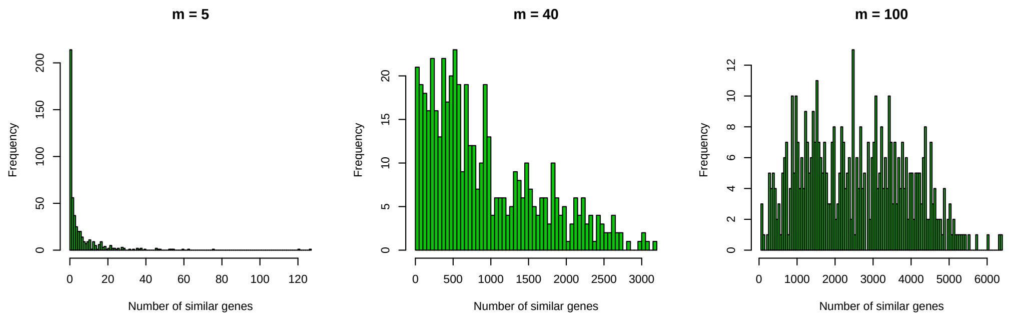


Figure 4.1: Distribution of the number of significant results, depending on the number of datasets used. The number of neighbours grows very rapidly, when increasing the number of experiments

# of datasets	5	10	20	40	100
Similar genes on average	6.478	45.940	263.936	915.906	2540.600

Table 4.2: The table shows how the average number of similar genes depends on the number of datasets used in the query.

results become incomprehensible even with moderate number of datasets. Just to look through thousands of genes is a very tedious work. Even more importantly, what does it exactly mean that we have thousand co-expressed genes? This is almost 5% of the entire genome. All those genes cannot have very strong functional similarities to the query gene. Thus, by increasing the number of datasets, the power of the test grows so fast that the relevance of "statistical significance" diminishes.

Chapter 5

Advanced co-expression thresholds

Our study of Bonferroni thresholds showed that using only statistical significance for determining the number of co-expressed genes does not work. The power of the test grows so fast when increasing the number of experiments that the number of results gets out of hand very quickly. Thus, the statistical significance plays a role only when we consider a query on very few experiments. On larger queries, any reasonable number of top results is almost certainly statistically significant. So question, how many results to show to a user, became the question how many results a user can digest.

The fact that co-expression is significant between two genes does not tell us too much about the actual connection between them. In most cases it indicates that two genes participate in the same processes on very general level, for example participate in development, metabolism or immune response, but do not have direct connections. These kinds of relationships are not what we are after. We want to find connections that can predict real interaction between the genes. Thus, in addition to determining the difference from random case, we have to characterize the strength of co-expression.

We have always an option to show only the k best results on every query, but that does not make too much sense biologically. The number of relevant neighbours for a gene can vary a lot. For genes that work in large complexes, we expect many strongly co-expressed neighbours, at the same time, some regulatory genes should have only few close neighbours.

Of course, the BetaMEM scores do not only show significance, but also measure the strength of similarity. The problem is that beyond significance, we do not have any interpretation for the score values. Thus, it is hard to place biologically justified thresholds for the scores. The problem becomes even worse if we consider queries with differing number of datasets. For example, consider the case we want to make the query on datasets corresponding to specific tissue type or biological manipulation. Then the score depends on both: the strength of co-expression and also on the power of the test. Hence, we have to find a consistent way to interpret the scores that we can define the desired strength of co-expression beforehand and then translate it to actual score thresholds. The critical part is to take into account also the dependance between p-values and the number of datasets used in query.

5.1 Percentile approach

One possibility for thresholding the number of genes, is to study the overall distribution of BetaMEM p-values. We can perform the query with all the genes and record all the similarity scores. This way we obtain a global distribution of similarity scores on real data. The natural way to place similarity thresholds to the data, would be using the percentiles of the global distribution. These thresholds would correspond to testing hypothesis if the co-expression between these two genes is stronger than between two randomly selected genes. The choice of the percentile to use would correspond to the desired confidence level. For instance, if we take the 1% percentile of a global BetaMEM score distributions for a threshold, then we test the given hypothesis on significance level 0.01. Basically, this thresholding scheme shows how exceptional is the result given all possible queries. Even more intuitive way to express the percentile, is by fixing the number of expected results. Since we have about 22000 genes on an array, then on average 220 genes should exceed the 1% percentile threshold on each query. In context of our web tool tuning the number of expected results is a natural way of adjusting stringency of queries.

Of course, the scores still depend on the number of datasets used. So to use this method we have to calculate the thresholds for every possible number of datasets. These thresholds do not depend only on m, but also the corresponding datasets. A priori m random datasets will produce thresholds similar to the Bonferroni results. So is it enough to calculate the thresholds on every m or do we have to calculate the thresholds on every single combination of datasets. If the latter is true, then the task would be close to impossible.

5.1.1 Calculation of percentile thresholds

To study the applicability of the percentile thresholds, we examined them on real data. The data consisted of 140 microarray datasets downloaded from NCBI GEO public microaarray database. All experiments were performed using Affymetrix HG U133 platform.

To study the threshold dependance on number of experiments m we took random subsets of experiments with sizes 10, 20, 30, ..., 140. To see the variability introduced by selection of experiments, we had 3 different subsets for each value of m. We calculated the thresholds for each combination of experiments. Since calculating the whole score distribution means about 22000 queries, we decided to approximate it by using results of 2000 queries per subset of experiments. The percentiles were obtained by first storing the results, sorting them and taking the value on appropriate position. Found percentile thresholds are depicted on Figure 5.1. The resulting thresholds on the figure correspond to 4, 16, 64, 256 expected results for a query.

By increasing the number of experiments m the thresholds are really getting smaller, as expected. However, the replicate experiments on the same m value are quite close together. To summarize, the variation introduced by using different subset of experiments and approximating the score distribution is much smaller than the variability due m. The latter means that we can calculate the thresholds by taking into account only the number of datasets used for query.

However the most interesting result is the clear linear dependance between the thresholds and number of experiments m in the log scale. This dependance means we can calculate the thresholds without having to do ex-

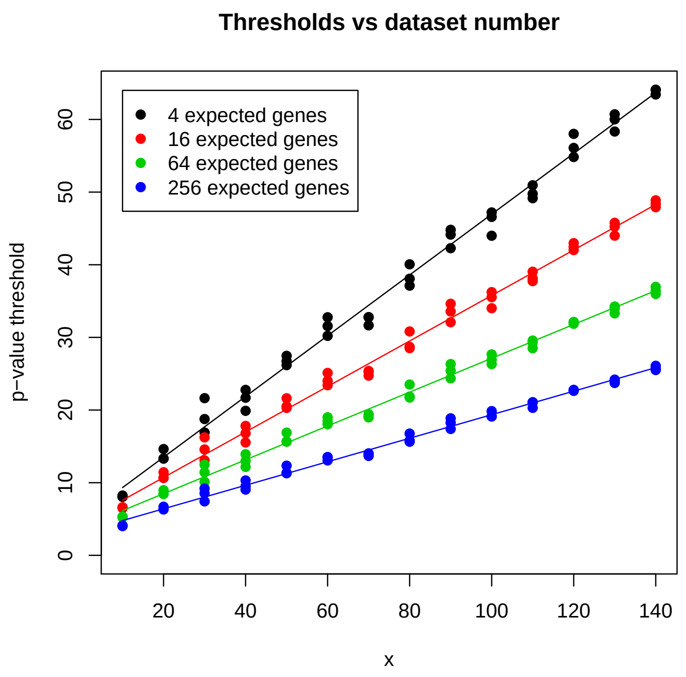


Figure 5.1: Linear dependance between number of datasets used and p-value thresholds. The thresholds are given on negative $\log_{10}$ scale.

tensive queries. We just calculate the thresholds on few m values and find the linear regression line through them. For intermediate values we can use the values predicted by the regression line.

5.1.2 Testing the thresholds

Now that we know how to compute reasonable thresholds we can take the next natural step in applying them to the real data. Although the thresholds specify the expected number of genes per query, we do not know how the actual distribution of results looks like.

In order to get some insights on this issue, we analyzed the same data we used in threshold computation. Namely we studied, how many genes are above the similarity threshold. An example in the case of using all experiments can be seen on Figure 5.2. The distribution is very similar if smaller subsets of data is used.

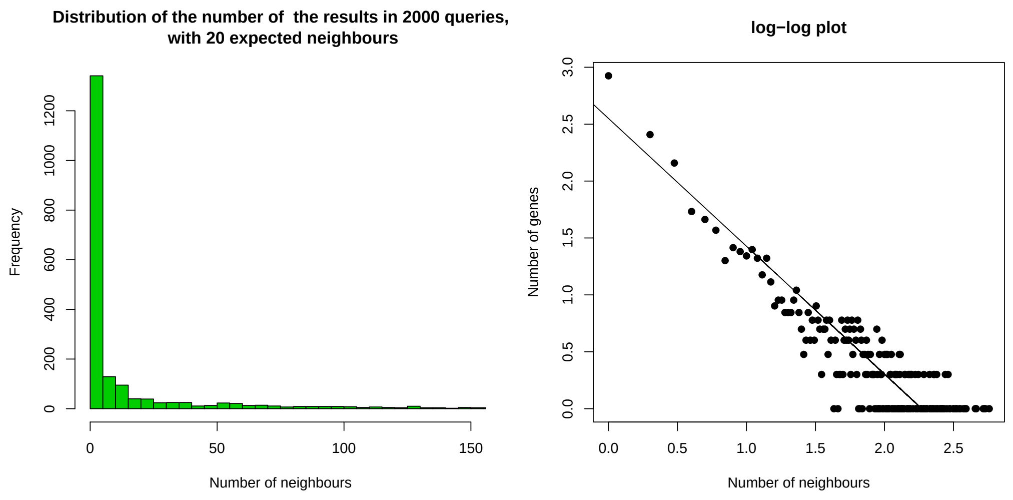


Figure 5.2: Histogram describing the distribution of number of results of a query. This example is based on 2000 queries with expected return of 20 genes, 76 values were left off the picture because they were bigger than 150. The maximal value was 574. On second plot we can see the relation between the number of neighbours and the number of genes having so many of them, in the $\log_{10}$ scale. This graph shows that the number of neighbours follows approximately the power law.

The bottom line is that the results are really asymmetrically distributed. In fact, the second graph reveals that the distribution follows approximately the power law. Power law is a polynomial relationship between two variables, usually in the form

$$f(x) = ax^k, \tag{5.1}$$

where a and k are constants. The examples of power law can be found everywhere, for example famous 80 - 20 rule (80% of the effects come from 20% of the causes) is based on it. It has been found, that the distribution of the number of connections of nodes in all kinds of networks, tend to follow power law.

After taking logarithm from both sides of (5.1), we get a linear relationship $\log(f(x)) = k \log x + \log a$. In brief, the data follows the power law if the log-log plot is close to linear. In our case, a line fits the data very well. Linear regression on the logarithmed data described 79% of the variation (R^2). It has been shown that this is the case also with gene expression networks [SSKK03, LHS+04]. So our results fit in really nicely.

5.2 Rank rescaling methods

In this section we propose another algorithm for constraining the number of results and making BetaMEM scores independent of the number of datasets used in the query. This approach takes us a step back to the rank matrix. We modify the rankings before calculating the BetaMEM p-values.

The BetaMEM method is a statistical test that measures how much the ranking distribution is skewed towards 0 compared to uniform. The study with Bonferroni thresholds showed that the test power grew rapidly, when increasing the number of experiments used in query, *i.e* increasing the sample size. We want to control the number of results in a query, by specifying the strength of the co-expression beforehands. A natural idea is to control the power of the test, by adjusting the sample size.

As before, we assume that the meaningful ranking vectors are samples from a mixture distribution, with uniform and skewed component. The more skewed the mixture the stronger the co-expression. Depending on the strength of similarity, different sample size is needed to distinguish it reliably from uniform distribution. Thus, the strength of co-expression can be classified, by the size of sample we need for detecting it.

In practice, the user will specify the desired strength of co-expression, by giving the sample size. The ranking vectors are calculated as usual, but before calculating the BetaMEM scores, the ranking vectors are transformed into a new sample with the size given by user. The p-values are calculated based on the new samples. At the end, Bonferroni thresholds can be applied to the scores to detect significant similarities.

Let us have a sorted ranking vector $r = (r_1, \ldots, r_m)$. We assumed that this ranking vector represents a sample from an underlying mixture distribution. Our goal is to rescale the sample without losing information about the distribution. It is useful to represent the sorted ranking vector as an increasing function of the element position. We can normalize the positions also into the range of $(0, 1)$ and denote $x_i = \frac{i}{m}$. Figure 5.3 shows two interpolations of a ranking vector.

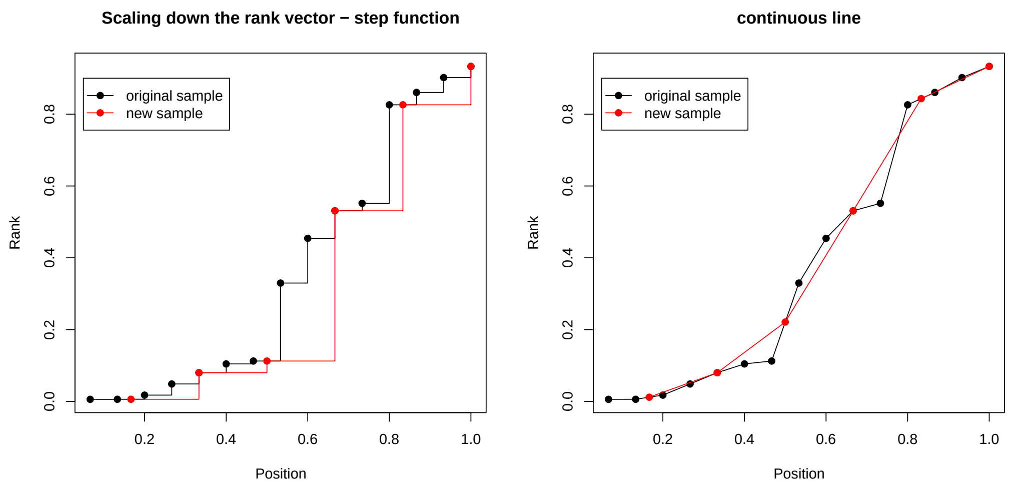


Figure 5.3: Different ways to represent and resample the ranking vector. On first plot the vector is represented as a step function and on the second as a linear interpolation of the points. The original vector has 15 points, the red line represents the resampled vector that contains 6 points.

The first is a stepwise function, where all the values in range (x_i, x_{i+1}) are equal to r_i. The second plot represents the linear interpolation, where all the points from ranking vector are connected with a straight line. Each line S_i between the dots x_i and x_{i+1} can be written down as

$$S_i(x) = r_i + \frac{r_{i+1} - r_i}{x_{i+1} - x_i}(x - x_i). \tag{5.2}$$

The rescaling can be done using those functions. Suppose the desired sample size is k. We can lay a grid with k points between 0 and 1, for example take values $\{1/k, \dots, k/k\}$. We obtain the new ranking vector, if we calculate the values of our interpolated functions on this grid.

Technically the method suits for both scaling down the ranking vectors and also scaling up, if the original sample size is smaller than desired. The question is, if scaling up is advisable. Does the smaller sample describe the distribution well enough that the BetaMEM scores on rescaled larger sample will be similar to those obtained from real distribution. Or the upscaling amplifies all the random effects and produces artificially better scores than the actual underlying distribution would. Of course downscaling may have its problems. The main question is, how much information we lose by rescaling the vector. How much does the order of results change, if we throw out most of the observations.

We have to study the two possible problems outlined above. If the method can cope with both of them, then we have an intuitive algorithm that achieves two goals at once: we can adjust the co-expression thresholds by the strength of similarity and make the score independent from m.

5.2.1 Rescaling influence to BetaMEM scores

We assumed that the ranking vector comes from a mixture distribution. By rescaling the vector we try to preserve the most important characteristics of the distribution, so the BetaMEM scores are comparable to those obtained, by sampling the vector straight from the mixture distribution. To study BetaMEM score behaviour in case of rescaled vectors, we fixed the mixture distribution, generated samples with different sizes from the distribution, rescaled them and applied BetaMEM.

The mixture distribution was defined as follows: 20% values were from $Beta(1, 200)$ distribution, which is really close to 0 and 80% are from uniform distribution. The rank vectors are sampled with sizes 10, 20, 40 and 80. With each size we sample 1000 vectors. The vectors are rescaled to size 40 and BetaMEM scores are calculated based on the rescaled samples. The results, both with step-function rescaling and simple spline are on Figure 5.4.

As we can see on the figure, then downscaling the samples does not do any harm in terms of BetaMEM score distribution. The distribution of scores in the downscaled samples seem to be identical to the no rescaling case. However upscaling can cause problems. The things are especially bad using step-

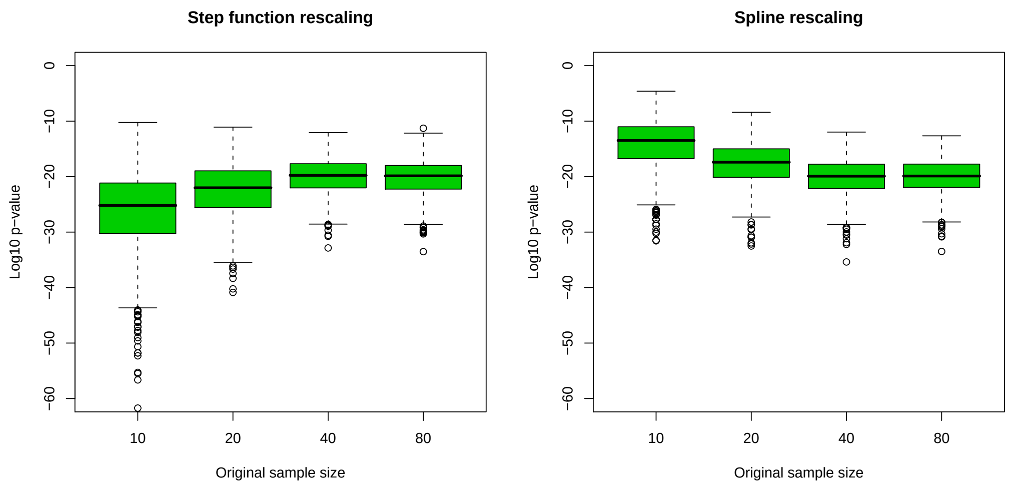


Figure 5.4: BetaMEM scores on the samples with different sizes from same distribution, rescaled to sample size 40.

function, it tends to create artificially smaller BetaMEM scores. However, the spline approach works fine both ways. On upscaling the scores are a bit smaller than they are supposed to. Thus, the upscaling is rather conservative, the co-expression that we can detect has to be stronger, than the one we can detect from the larger sample. This means we do not amplify the random effects of smaller samples and introduce false positives.

On downscaling another effect comes into play that is not visible from Figure 5.4. The BetaMEM scores from downscaled samples have a smaller variation than scores from vectors with same size that have been sampled straight from the mixture. To reason for this is very simple. Say we know the real distribution generating the rankings. Then we can find a most representative sample from that distribution, with the size k and calculate the BetaMEM score based on that. There is nothing random in this process, so there is no variance in resulting score. Now by increasing the sample size we get better approximation to the real distribution generating the scores. Therefore, the variance of the score decreases. Illustration of this can be found on Figure 5.5.

In summary we can say that in terms of BetaMEM scores the rescaling

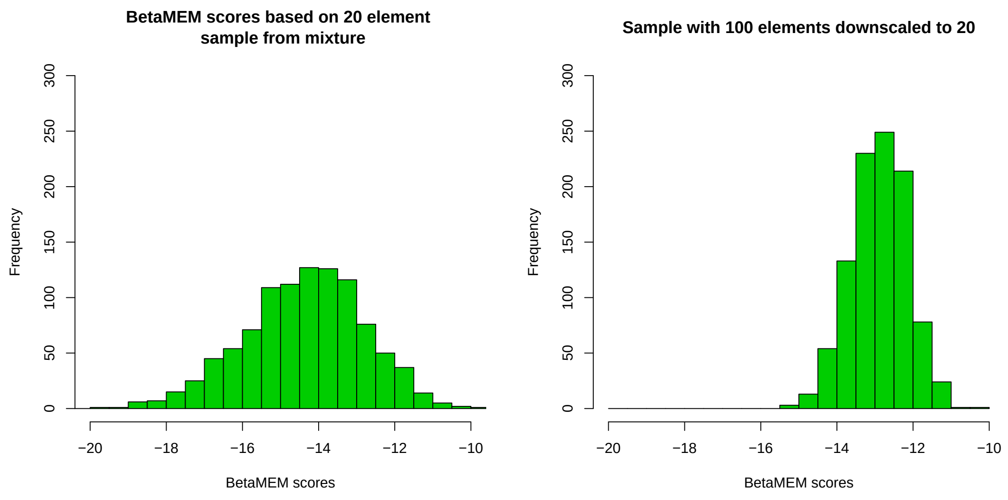


Figure 5.5: BetaMEM score stabilization. The first plot displays the BetaMEM score distribution if a sample with 20 elements was taken straight from the distribution. Second shows the scores on the same distribution, but the original sample size was 100 and it was downscaled to 20. The scores are on log_{10} scale.

of ranks is reasonable, especially when using the spline based method. On downscaling, the distribution of BetaMEM scores stabilizes. On upscaling, the distribution of the scores does not become unstable and create artificially false positives.

5.2.2 Stability of the list on rescaling

Another question is, how much information we lose when rescaling the rankings? If after rescaling the order of the genes changes completely, then the rationality of this method is questionable. To explore this issue we applied the rescaling to query results on real data to see the magnitude of changes in the order.

To get some intuition, we performed queries with 10 and 100 random datasets. Then we rescaled both the rank matrices of both queries to sample size 40, recalculated BetaMEM scores and reordered genes based on them.

Scatterplots that depict the dependance between old and new rankings can be seen on Figure 5.6.

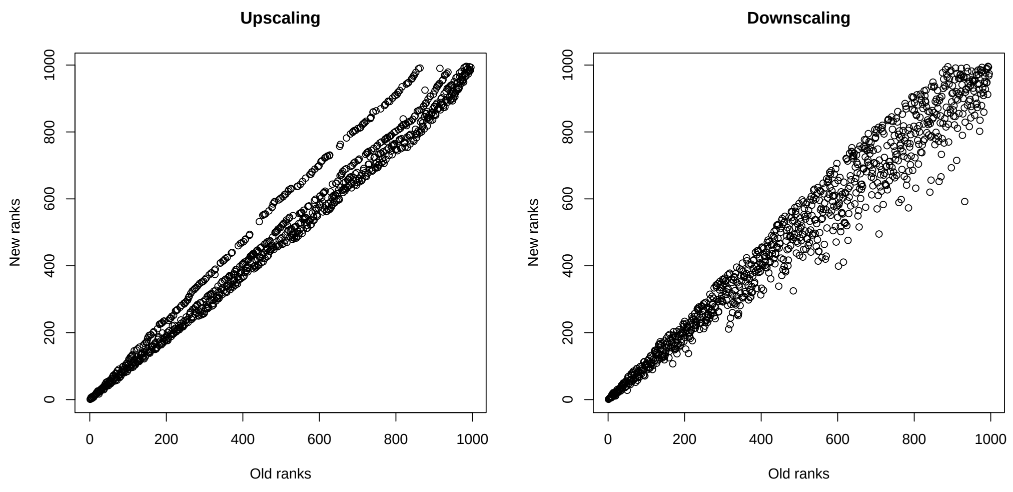


Figure 5.6: Changes in the order after rescaling. Upscaling is done from 10 to 40 grid points, downscaling from 100 to 40.

We can see, that the rescaling does not stir up the rankings completely. There is a strong correlation between the old and new rankings. It is interesting to note that the differences between rankings seem to grow linearily, when increasing the ranking. This suggest, that the magnitude of changes in the order of results is proportional to the rank number. Thus, we can quantify the changes introduced by rescaling, by studying the proportional differences in rankings. Let σ and ρ be the orderings before and after rescaling. Then for every gene i the rescaling error is

$$e_i = \frac{|\rho_i - \sigma_i|}{\sigma_i}. \tag{5.3}$$

For example in the cases presented on Figure 5.6 the average errors are 0.12 and 0.06 correspondingly. This means, on upscaling the average difference between the old and new ranking is about 12% of the old ranking value.

We performed the same analysis on a number of queries, to get more comprehensive picture about the errors introduced by rescaling. We carried out

500 queries with 10, 50 and 100 datasets. Rescaled the ranking matrix of the top 1000 genes, reordered the genes and calculated the average differences. The results are in Table 5.1.

	10	25	50	75	100
10	0	0.122	0.117	0.139	0.141
50	0.170	0.088	0	0.035	0.011
100	0.131	0.076	0.045	0.031	0

Table 5.1: Average errors introduced by rescaling. Rows represent the number of datasets used and columns the target sample size.

We can see, that the order does not change too severely when rescaling. Average difference in position is in every case less than 20% and usually much smaller. Also, it appears that the stability of a list increases, if we have larger original sample. Probably the differences between results are more pronounced in case of larger sample and the downscaling does not affect the relationships this much.

5.2.3 Choosing the target sample size

We claimed that we can specify the strength of co-expression by choosing sample size. In our web tool, we plan to let the user choose the target sample size in order to adjust the number of displayed results. Of course the sample size itself, without knowing anything about the test, is rather unintuitive. Therefore, we simulated samples from different scenarios and tested how big sample is needed to detect the bias towards 0. The results of the simulations should give intuition on selecting the target sample size on rescaling.

In theoretical treatment, we have assumed the mixture distribution of rankings, where rankings come from either uniform distribution or one that is skewed strongly towards 0. There are two parameters in the mixture that influence the BetaMEM score. One is the proportion of elements from the skewed distribution and the other is the actual skewness of the component. If we want a comprehensive picture about the target sample size, we have to vary both of the parameters.

We performed a following experiment. As the skewed distribution we used, the beta distribution with the first shape parameter being equal to 1. This left us decreasing distribution, with density function

$$f(x) = \frac{1}{B}(1-x)^{\beta-1}, \tag{5.4}$$

where B is a normalizing constant and β the only parameter. The distribution is defined in range $(0, 1)$ and its expected value is $1/(\beta + 1)$.

We fixed a set of values for both parameters of the mixture distribution. In choosing β values we kept in mind that we have roughly 20000 genes on a chip. The chosen β values correspond to the average rankings of $10, 20, 30, \ldots, 300$. The proportions of skewed component were chosen to be $10, 15, 20, \ldots, 100$. With each pair of parameter values, we tried to find smallest size of the sample in which we can reliably detect the non randomness of the data, by using BetaMEM and Bonferroni threshold on significance level 0.05.

To do that, we generated samples with 1000 rank vectors with given parameters. We considered the sample as reliably detected, if BetaMEM classified at least 950 rank vectors as significant. The results of the test can be seen on Figure 5.7.

The results on Figure 5.7 can be used as a reference when user wants to choose the target sample size in rescaling. Namely, we can visualize the desired strength of co-expression and then look up from the Figure, how to choose the target sample size. For example if we expect to have at least one third of the rankings small, then we can rescale the rank vectors to size 20.

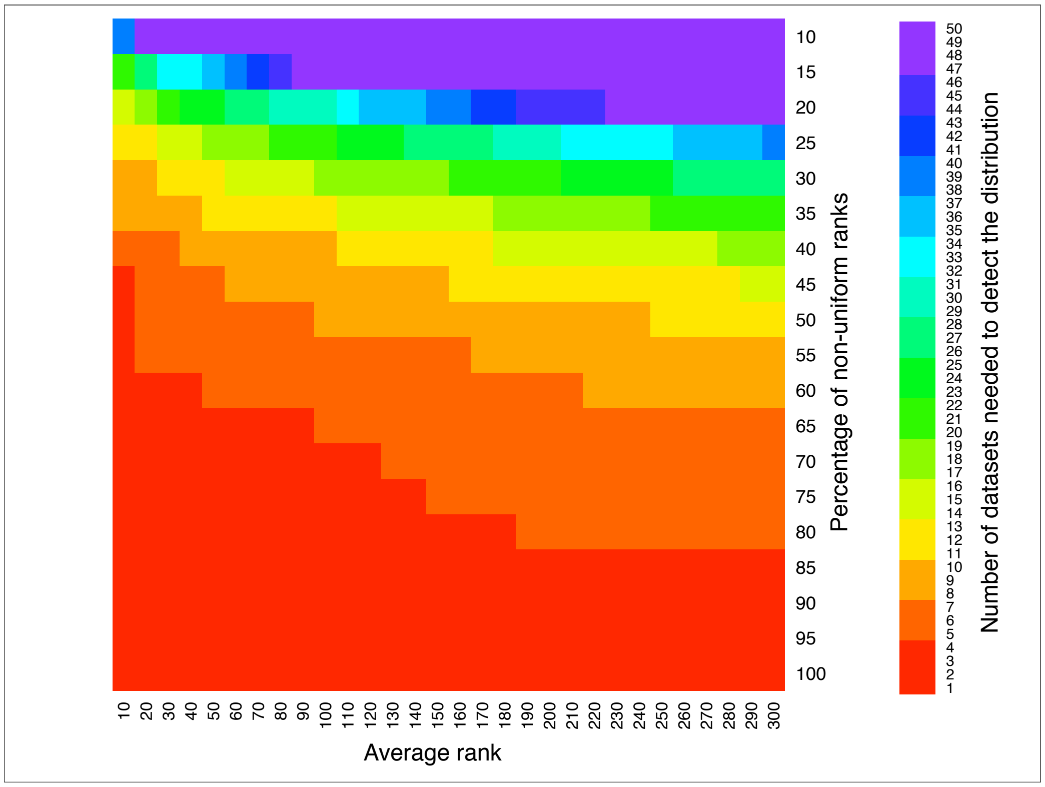


Figure 5.7: Study on the size of sample that is needed to detect reliably a rank vector from mixture distribution.

Chapter 6

Dataset selection

So far we have developed methods for ordering the genes based on co-expression with the query gene. We have also studied ways to determine the number of relevant results. In this chapter, we study methods for selecting the datasets. The selection of experiments that are used in the co-expression search, is probably the single most important factor determining the outcome of a query. We can improve the results significantly, if we apply proper dataset selection.

more precisely, we explore how to quantify the importance of a gene in a certain experiments. The latter allows us to improve the query by using only datasets where a gene plays important role. This way we hope to find neighbours that are similar to the query gene in most essential functional aspects. We estimate the importance by variation of the gene in a experiment.

6.1 Dataset selection based on standard deviation

Microarrays are not very good at quantifying the absolute expression of a gene, instead they give accurate picture about the dynamics of changes. That is the reason, why correlation is a popular measure for co-expression. But most of the genes are not changing their expression all the time. In a typical experiment, the conditions are chosen to be quite similar, so not too many processes change between them. Thus, most of the genes are constantly either turned on or off and the changes in their expression can be attributed to random noise. On the other hand, similarity search gives less random results if the query gene has big expression changes in the experiment that reflect real biological rearrangements.

We can find an indirect confirmation of this claim, if we study the expression of genes that have many neighbours and those that have only few. As an illuminating example, we took two genes that had hundreds of neighbours, when using the percentile thresholds UBE1C and HNRPR and genes that had none RABL5 and KCNIP2. Their global expression distribution over all datasets can be seen on Figure 6.1.

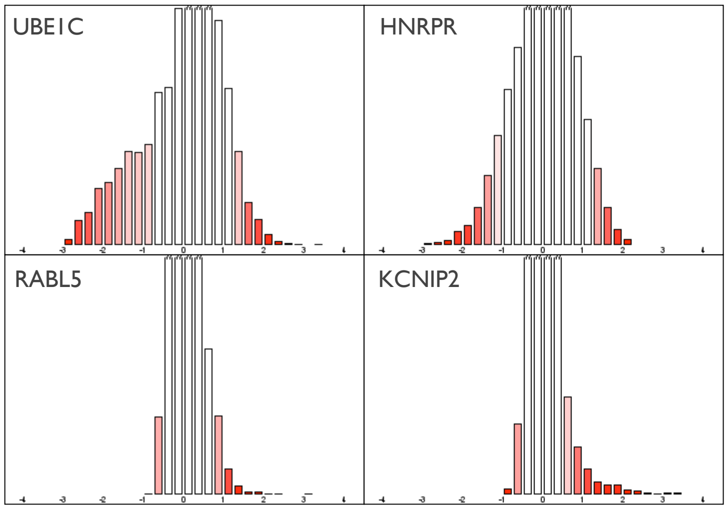


Figure 6.1: Histograms of global gene expression distribution, for genes with large number of neighbours are on the top row and no neighbours are on the bottom row. The expression values are gathered from all datasets.

We can see immediately that the genes with many neighbours have a much wider distribution. Consequently, they have large changes in many datasets and similarity search in those experiments gives probably meaningful results. As a result, BetaMEM test has more power to discover the co-expression.

Certainly we did not prowe that the large variation guarantees that we get meaningful results, but might be a good indication. Nevertheless, we can use it to select datasets for query. Namely, we are going to assume the model as follows for characterizing the influence of variation on search results. We assume that there is a threshold for standard deviation of a gene expression profile that can differentiate between the random fluctuations and meaningful changes. If the standard deviation of a gene exceeds this threshold, then its size does not matter too much. We can just consider that it is showing a biologically meaningful effect. If we can find such threshold, then we have a method for experiment selection. We can use just the datasets, where the query gene has larger standard deviation than our threshold.

6.1.1 Study of gene expression variability

To verify the hypothesis postulated about the connection between gene expression variability and number of query results, we conducted a series of experiments on gene expression variability. The main aim was to explore and characterize the dependance between query gene variation and the number of results.

We calculated the standard deviations for every gene in every dataset. For comparison we ran queries for 2000 random genes and found the number of similar genes for each of them, by using percentile threshold that returns on average 20 genes per query (see section 5.1.1). Now we had both the data about the number of MEM results and also the standard deviations of those genes in all datasets. Our model assumed that we can set a standard deviation threshold in a way that the number of datasets, where the genes variation exceeds that threshold, is correlated with the number of MEM results. We used standard deviation threshold 0.45 for testing out our model. The threshold has proven to be reasonable in our prior experience.

For each of the 2000 genes, we have the number of results and the number of datasets where its standard deviation exceeds 0.45. The correlation between the two variables is 0.3, which is not too strong, but still indicates an association between the two. To get a more precise picture about the connection, we built a simple linear model between the variables:

$$n_i = \alpha + \beta d_i + e_i, \tag{6.1}$$

where α and β are the model parameters, n_i shows the number of neighbours for gene i, d_i is the number of datasets where its standard deviation exceeds 0.45 and e_i is the error term. The fitted model showed that the factor d was very significant for estimating the number of neighbours, its significance p-value was close to 0. However, the model describes only 9% of the variation, which is quite low. That means the standard deviation certainly is a factor in determining the number of neighbours, but it does not explain too much.

We showed that even with a rather arbitrary standard deviation threshold we can see the correlation between number of neighbours and variation. To use the fact in practice, we should find a most optimal threshold that distinguishes biological noise from meaningful expression. We can use the same linear model to optimize the standard deviation cutoff, by finding one that maximizes the significance of linear model.

We optimized the threshold, by trying a set of values. We fixed a grid of possible thresholds: 0.10, 0.11, 0.12, ..., 1.00, and fitted the linear model in each case. The range of grid was chosen based on previous experience. Figure 6.2 displays the relationship between the standard deviation threshold and a t-statistic that measures the significance of the threshold in predicting the number of neighbours.

Figure 6.2 clearly indicates that the standard deviation threshold that maximizes likelihood of the model was 0.33. The unimodal shape of function seen on Figure 6.2 matches pretty well the intuition about the thresholds. Namely, if the threshold is too low, we include noisy datasets to the search, if it is too high, then we throw out potentially useful information.

In case of our best standard deviation threshold 0.33, the linear model describes 12% of the variation in the data. The number is still quite small. To see why, let us turn to the scatter plot of the data that is shown on the Figure 6.3. We can see that most of the queries have very few results, regardless of the number of datasets with large standard deviations.This is consistent with our earlier observation that the distribution of the number of neighbours follows approximately the power law. Now if we concentrate our attention to the genes that have many neighbours, we can see clear dependance between the number of results and highly variable datasets. Thus, we can conclude that high variability of gene expression is necessary, but not

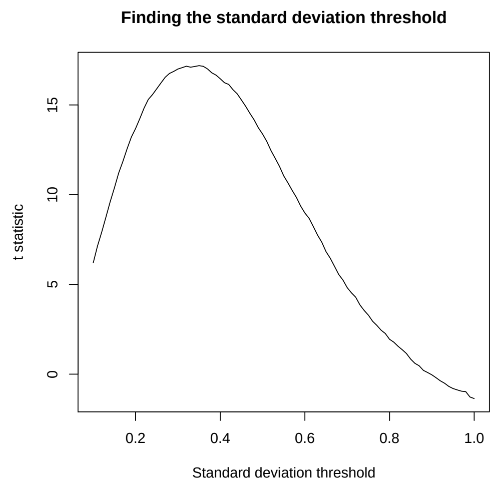


Figure 6.2: Optimization of the standard deviation threshold. On the y-axis is the t statistic showing the significance of the number of highly variable datasets in estimating the number of results. On x-axis is the standard deviation threshold.

sufficient, condition for having high volume of neighbours. This means, we do not lose too much information by throwing out the datasets where the variation of query gene is low.

The plot on right hand side on Figure 6.3 is there just to reassure that with our threshold the number of highly variable datasets is reasonably

6.1.2 Experiments with standard deviation threshold

Another interesting aspect to study is how this selection method based on query gene variation works in practice. Therefore, we ran the queries on selected datasets. To evaluate the goodness of results, we decided to look at the corresponding BetaMEM scores. As a comparison we used query results with random selection of datasets. If the scores with selected datasets are consistently better than with random experiments, then it shows that we have really reduced the noise and obtained more relevant results.

We took 20000 genes and for each gene identified the datasets, where its

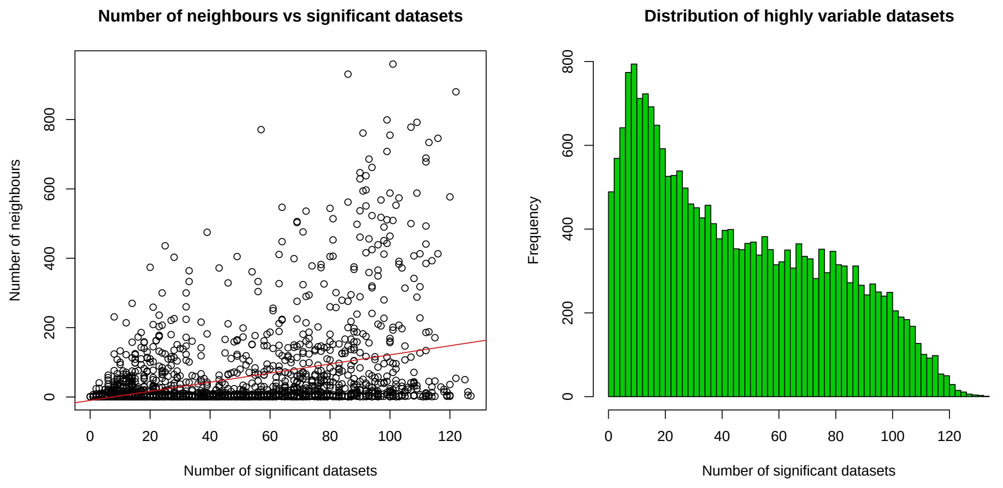


Figure 6.3: The left pane shows the dependance between the number of results and number of datasets, where the standard deviation of expression profile exceeded 0.33. The right pane describes the distribution of the number of highly variable datasets over genes, given the optimized threshold 0.33

standard deviation exceeds 0.33. These datasets were used in the queries with the corresponding genes. Our goal was to compare the similarity scores from these queries with the scores with random data. We want to see if the scores on selected datasets are consistently lower than on random datasets. Let us recall the percentile based thresholds from section 5.1.1. The thresholds were just percentiles of the global score distribution that characterize well the behaviour of the smaller scores. To compare the distributions we can study the percentiles of BetaMEM scores on selected and random datasets. Previously we varied the number of datasets m and generated 2000 query results for each one. Now the situation is more complicated, since m depends on the reference gene. To solve the problem, we grouped the queries by the number of experiments and calculated the percentiles in each group separately. The new percentiles together with old ones are shown on Figure 6.4.

Figure 6.4 clearly indicates that BetaMEM scores on selected datasets are significantly better than the ones on randomly chosen data. The lines on

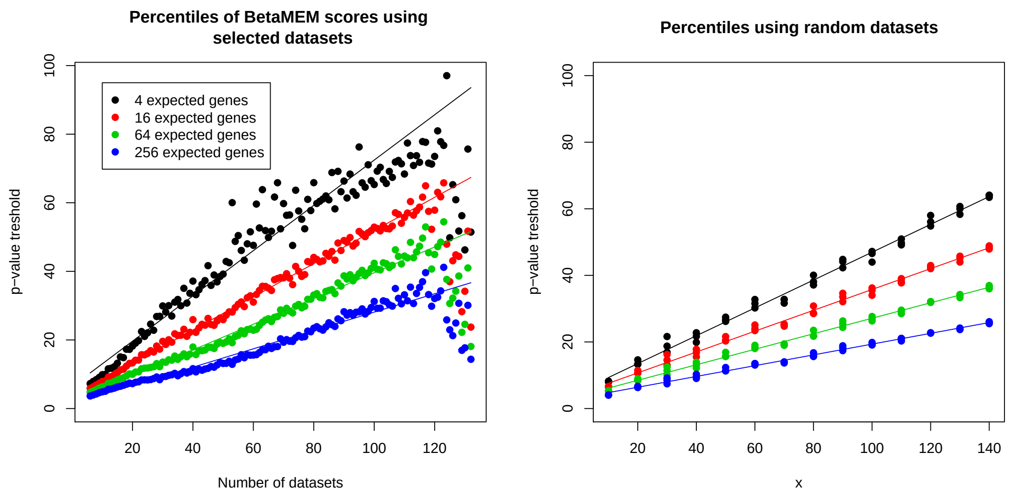


Figure 6.4: Comparison of the BetaMEM score distribution in the case of random and selected datasets. The p-values are on -$\log_{10}$ scale

the picture describe the BetaMEM thresholds which should give out certain number of genes on average. A closer look at the two sets of thresholds reveals that the ones based on selected data are significantly better. We can say very roughly that using the thresholds derived from randomly chosen data will give four times more results on selected datasets than expected. The results show clearly the utility of dataset selection.

Let us use the percentiles as thresholds on the actual queries, to see if the standard deviation based dataset selection had any influence on the distribution of results. The percentile values calculated on each m value were little noisy (see Figure 6.4 left pane), due to small number of queries each of those is based on. Therefore, we approximated the thresholds with linear regression through all the individual percentiles. We applied them on all the previous queries. Figure 6.5 contains a comparison of the number of results with dataset selection and without it.

Although the distribution of the results seems to still follow the power law, it is bit more balanced, than in case of all datasets. Thus, there are less queries that do not give any results. This is of course good news in the context of our web tool. But it also shows that by careful noise reduction, we

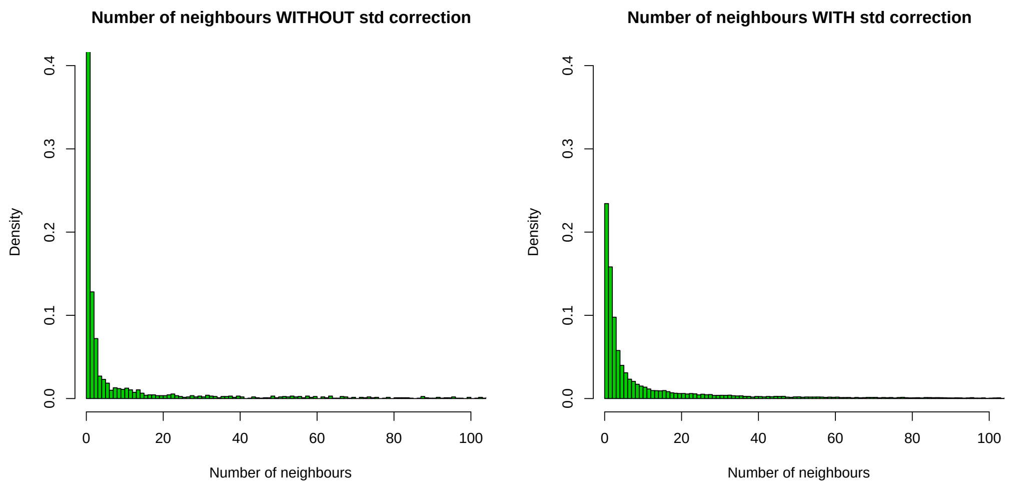


Figure 6.5: The distribution of results for 20 expected neighbours using all datasets vs using only significant datasets.

can reveal the connections between more genes and thus get more complete results.

Chapter 7

Practical outcome

The main goal of this thesis was to develop methods for expression similarity search over multiple datasets. Hand in hand with the evolution of the methods we have developed also the tool that implements them. Therefore, the most important practical result of our work is the actual tool itself. In this chapter we display the tool and discuss the features. Most of the simple features that were proposed in the thesis are already implemented, some of the more advanced methods are yet to be added.

We have also applied the tool on several settings. Here we show some results to confirm that our tool really finds biologically meaningful results. First, we present outcome of a WebMEM query with a known marker of breast cancer FoxA1. We find several other breast cancer genes that are similar to it. To show the applicability on a larger scale, we calculate expression similarities between genes from a metabolic pathway. The result allows us to identify a subgroup of genes from the pathway that are very tightly co-expressed.

7.1 The tool

There are not so many tools around that do gene expression similarity queries over several datasets. The main reason is technical and conceptual complexity. The query is computationally very expensive, since the corresponding data sets are big. Smaller datasets are only few, larger can be up to 40-50 megabytes. As a result, analyzing one dataset at a time is not a problem but if we have a hundred of them, then even reading in the data can be quite slow. As a second step, we have to measure the co-expression and sort the results that also takes time. According to our algorithm we have to calculate the distances and sort in every dataset separately and afterwards aggregate the results. This means a lot of computations that can be hard to do within reasonable time.

In our case, the computationally expensive part, let us call it `CoreMEM`, was programmed in C++ by Meelis Kull. `CoreMEM` works as a standalone command line tool. The main methods from this thesis are implemented in this part. The `CoreMEM` start the query from dataset selection, it is possible to select the datasets based on standard deviation of query genes. The next step is the actual calculation of all the relevant distances that can be done with many distance measures. The last part is aggregating the rankings, where also BetaMEM is implemented.

All this is done very fast, a full query with 140 datasets takes time about 10 seconds, if less datasets is used, the speed is even higher. `CoreMEM` is especially useful if one does a large scale study, where results of many queries are needed. For example, many experiments in this thesis would have been impossible without that tool.

Although the `CoreMEM` is very useful for large scale studies and for power usage, it is too complicated for ordinary biologist. In particular note that the interpretation of results can be difficult for persons without sufficient statistical background. To alleviate these shortcomings, we have built a web tool on top of the `CoreMEM`. The web interface was programmed by Priit Adler and it makes using our tool rather convenient. We call it WebMEM. The Figure 7.1 represents the user interface of WebMEM.

An important parts of WebMEM are also the data collections. We have gathered several groups of expression datasets from various sources. For example for yeast, mouse and human we have a data collection that contains all experiments that are publicly available on GEO database [BTW$^+$07]. In addition we have data collection, where each dataset represents different cancer and the data is compiled from different public sources [Con].

The normal usage of WebMEM is as follows. First, to do a query with default values, user has to insert only a gene name and select the data collection. The microarray can measure same genes with several different probes. If that is the case for query gene, the program offers the choice, which probe to use.

If the user wants he or she can customize the query further through the advanced options, see Figure 7.1. The first batch of options lets user choose the co-expression measure, rank aggregation method and output type. Although we have always used correlation distance in this thesis, there are other meaningful co-expression metrics that can be used in for queries. Therefore, we have a wealth of choices for the metric. In addition to correlation we have absolute correlation, eucleidic, minkowski, manhattan and many other distances.

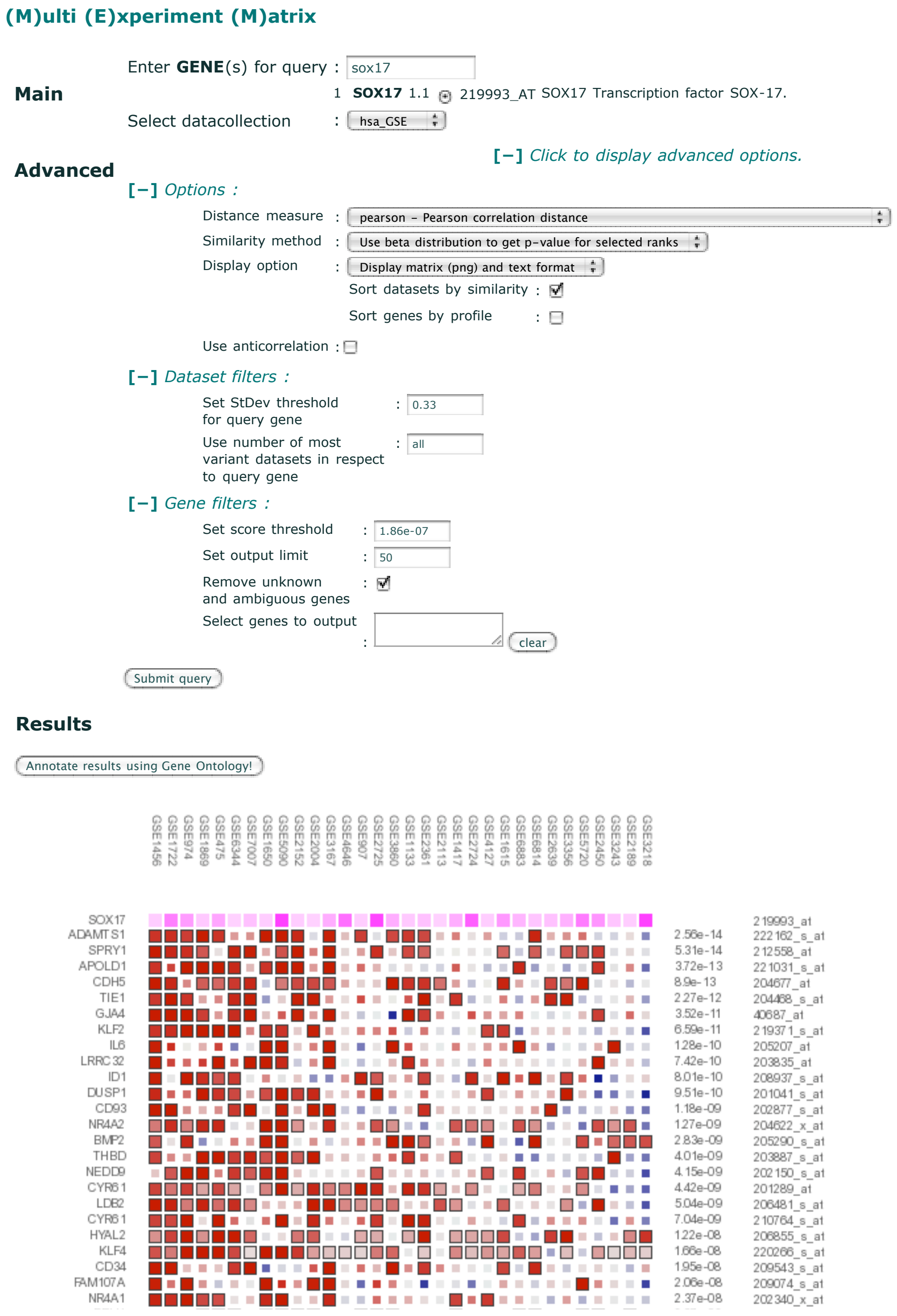
The default choice for rank aggregation method is BetaMEM, but there are other choices, such as mean and geometric mean of k smallest rankings. The output type can be chosen between textual and graphical output. A textual output just lists the genes and their similarity scores, but graphical shows also the rank matrix and indicates, where the co-expression was significant. The example of graphic representation is on the bottom of Figure 7.1.

The next set of options deals with dataset selection. the user can use for example a standard deviation threshold for that. Finally, we can select how much output we want. We can control it either by the similarity score or give a hard threshold on number of results.

The results themselves are represented as a table, where on the rows are found genes and on columns datasets. The table itself represents the rank matrix, based on what the similarity scores are calculated. Red squares represent the small rankings and blue the big. Large squares are the rankings that were used in calculating the similarity score. As an example consider BetaMEM score, then the ranks with large squares are the ones that were classified as significant by BetaMEM. The purple squares on the top row represent the standard deviation, the brighter the color the smaller the standard deviation. This representation gives user information about how the query gene behaves in the selected datasets and in which datasets the the co-expressions occurred.

The link “Annotate results using Gene Ontology!” on top of the graphical output gives a functional characterization about the resulting gene list. It leads to g:Profiler [RKP$^+$07], a tool developed in our group. The g:Profiler tool finds functional categories that have large overlaps with the list.

One thing that current implementation of WebMEM lacks is an intelligent dataset selection form. Very often the biologists are interested only in connections between genes on very specific conditions, such as on some disease or tissue type. It would be very good if we could select datasets based on the biological conditions they correspond to. One problem is certainly the poor annotation of microarray experiments, it is hard to divide the experiments into biologically similar groups automatically. In our workgroup Aleksandr Tkatchenko is applying some text mining methods on the problem.



(M)ulti (E)xperiment (M)atrix

Main

Enter **GENE**(s) for query : sox17

1 **SOX17** 1.1 219993_AT SOX17 Transcription factor SOX-17.

Select datacollection : hsa_GSE

[−] Click to display advanced options.

Advanced

[−] Options :

Distance measure : pearson − Pearson correlation distance

Similarity method : Use beta distribution to get p−value for selected ranks

Display option : Display matrix (png) and text format

Sort datasets by similarity : ✓

Sort genes by profile : ☐

Use anticorrelation : ☐

[−] Dataset filters :

Set StDev threshold for query gene : 0.33

Use number of most variant datasets in respect to query gene : all

[−] Gene filters :

Set score threshold : 1.86e-07

Set output limit : 50

Remove unknown and ambiguous genes : ✓

Select genes to output :

Results

SOX17		219993_at
ADAMTS1	2.56e-14	222162_s_at
SPRY1	5.31e-14	212558_at
APOLD1	3.72e-13	221031_s_at
CDH5	8.9e-13	204677_at
TIE1	2.27e-12	204468_s_at
GJA4	3.52e-11	40687_at
KLF2	6.59e-11	219371_s_at
IL6	1.28e-10	205207_at
LRRC32	7.42e-10	203835_at
ID1	8.01e-10	208937_s_at
DUSP1	9.51e-10	201041_s_at
CD93	1.18e-09	202877_s_at
NR4A2	1.27e-09	204622_x_at
BMP2	2.83e-09	205290_s_at
THBD	4.01e-09	203887_s_at
NEDD9	4.15e-09	202150_s_at
CYR61	4.42e-09	201289_at
LDB2	5.04e-09	206481_s_at
CYR61	7.04e-09	210764_s_at
HYAL2	1.22e-08	206855_s_at
KLF4	1.66e-08	220266_s_at
CD34	1.95e-08	209543_s_at
FAM107A	2.06e-08	209074_s_at
NR4A1	2.37e-08	202340_x_at

Figure 7.1: The WebMEM.

7.2 Biological results

The tool can be used in wide variety of situations. Depending on the biological question, different aspects of the results are interesting. A single query can already give a lot of information. Often, the query can be only one step on a longer pipeline of actions. For example, when searching for regulatory elements on genome, we can use our tool to produce groups of tightly co-regulated genes. We may also opt for large scale queries, where we are trying to build huge gene expression networks, by finding the edges by queries of CoreMEM. Here we just give a few examples about MEM usage in biological context.

As the first example, we demonstrate the query with FoxA1 that is a widely known marker gene of breast cancer [WBW+07]. Corresponding outcome of the query is shown on the Figure 7.2. To reduce the noise the query was done using only datasets where FoxA1 had large variation. The red boxes mark the genes that are also often related to breast cancer. We can see that the all the genes in our results were close to query gene in first six datasets. It turned out that all the six experiments studied breast cancer. In summary, a query with a marker of breast cancer identified several other genes that are related to the disease and also brought out the experiments that were related to the cancer. Therefore, biologically our query gave meaningful results.

Of course a single query on a single gene does not prove anything, it only gives an intuition what kind of results we can expect. The second example shows the large scale usage of the tool and tries to give more comprehensive argument about the usability of it. We have mentioned several times that given a functionally related group of genes, we can associate new members to the group by searching genes with similar expression. This idea is based on assumption that functionally related genes are also co-expressed. Although this is a reasonable guess, it must be verified using practical data. To study this hypothesis, we calculated BetaMEM scores between genes known to work together and looked if we can find strong co-expression.

More specifically, we took the metabolic and signaling pathways from Reactome database [VDS+07]. Then we calculated the BetaMEM scores between all the genes on each pathway using the 140 human datasets. To

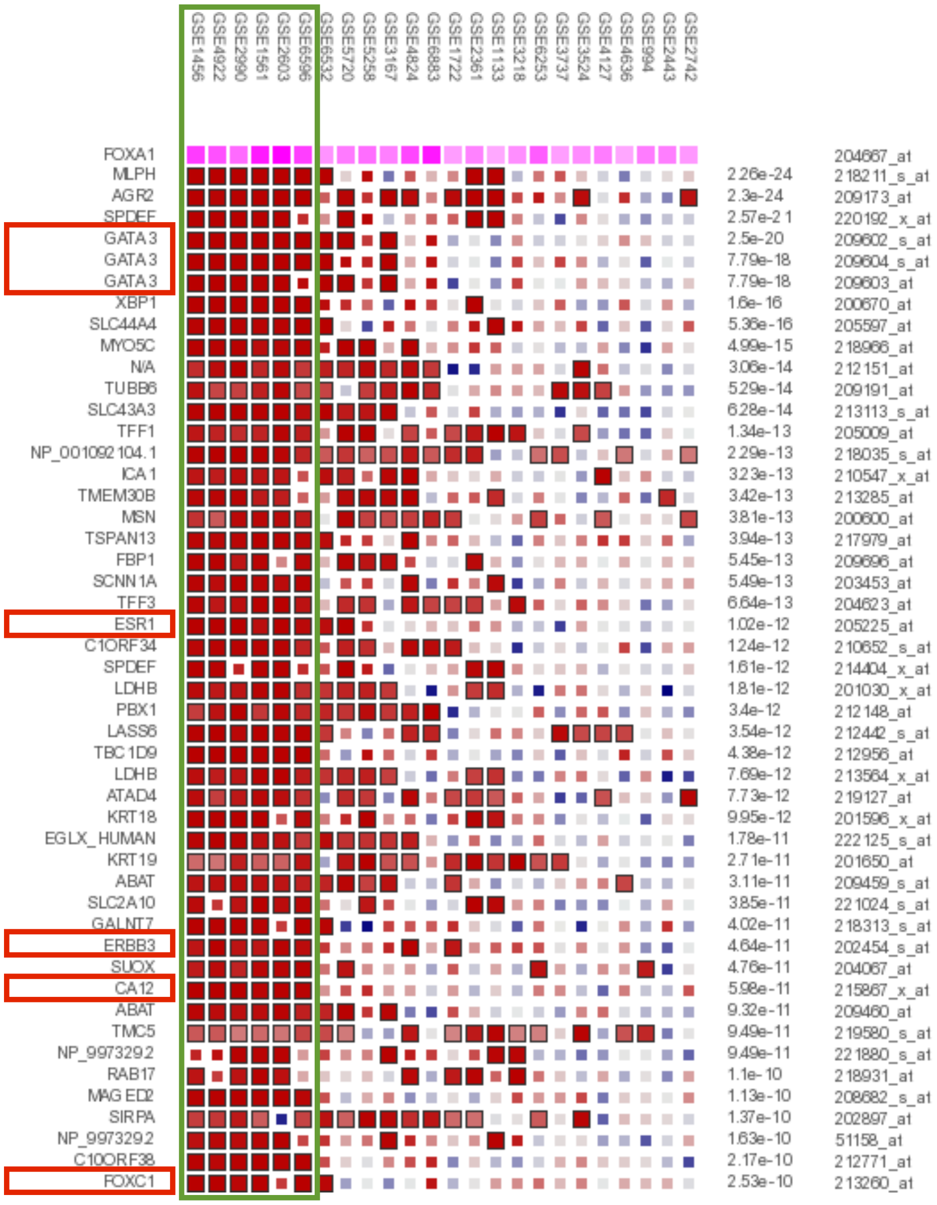


Figure 7.2: The WebMEM query result with a gene FoxA1. The red boxes mark other known breast cancer related genes. The green box indicates the datasets related to breast cancer.

visualize the results, we drew a clustered heatmap that has the same genes both on rows and columns. The color of the boxes on the heatmap represents the $\log_{10}$ of BetaMEM scores between the corresponding genes. The clustering should group together the genes that are similar with the same set of genes. Therefore, the clusters tightly of co-expressed genes should appear as darker boxes on the diagonal.

Figure 7.3 shows the results on Transcription pathway. The method clearly works, as we can clearly identify one large group of genes that are co-expressed with each other. Down the diagonal we can see other smaller groups of genes that are also co-expressed. Several other pathways had even larger groups of similar genes. The latter proves that some biological pathways, or at least parts of them, are really tightly co-expressed. Also, we are able to find the co-expression with with our tool.

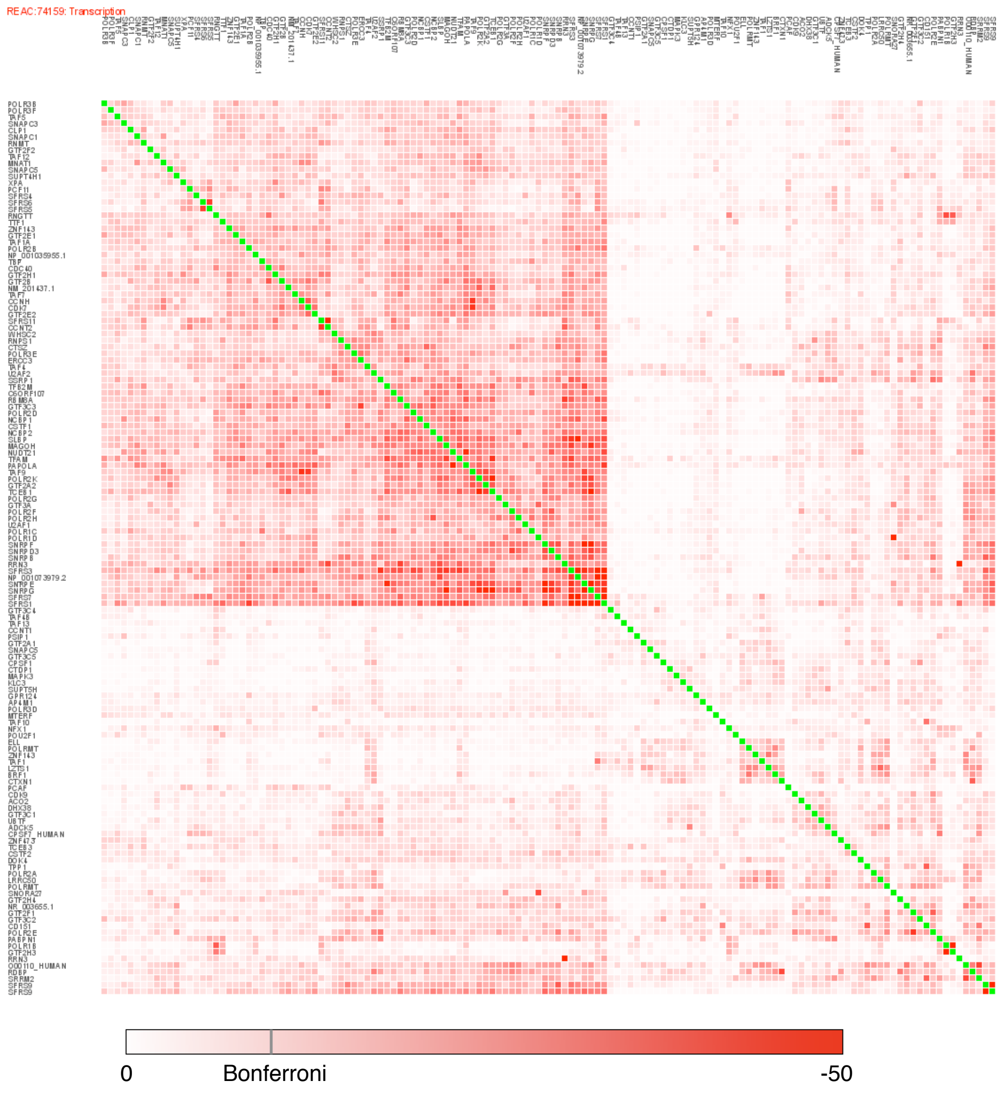


Figure 7.3: The BetaMEM distances between the genes on the Transcription pathway. we can clearly identify a group of genes that are very similarly expressed

Chapter 8

Discussion

Although we have solved most of the problems with rank aggregation, there are several inherent limitations. We can only do the similarity search using a single specific microarray platform at a time. Even though on some platforms there are hundreds experiments available, incorporating many platforms should give more comprehensive picture about the different tissues, diseases and biological conditions. In this chapter, we discuss possible ways to extend existing methods that we would be able to perform the query over several microarray platforms.

As a second problem, we note that sometimes we are not only interested only in co-expression with one gene, but rather with a set of genes. For instance, we may want to associate new genes with already known functionally coherent group of genes. Therefore, we discuss how to extend our methology so that we could measure distance between a gene and a group of genes.

These methods utilize the concepts introduced in the previous chapters, such as rescaling the rankings and BetaMEM score. This demonstrates the strength and universality of our early design decisions that make extending the capabilities of the tool relatively simple. This is important, since these advanced queries are not available in any existing tool that we know of. Thus, implementing those will be our top priority in the future.

8.1 Combining data from different platforms

The advances in technology mean that more accurate sequence data with better annotation becomes available all the time. That means it is possible to design more precise microarrays. The large companies update their chips from time to time, we refer to a chip as one platform. The difference between platform is mainly in the sequences of probes that capture the mRNA. Some of them are removed, some added and many changed. Thus, the collection of genes is changed and we cannot be entirely sure that the changed probes capture exactly the same mRNAs. All the reasons we mentioned make comparing data on different platforms not trivial.

One additional problem is, that expression of one gene is usually measured with several probes on a chip and the measurements are sometimes quite different. On one platform we may just use any of those probes and we do not have a problem. If we are going to combine the data, then it becomes a worry, because we have to find some kind of mapping between genes and the probes that measure them.

In spite of all those problems it is important that we could extend the MEM query over several platforms. The latter would dramatically increase the number of datasets biologists could use in co-expression studies. Although, results of chapter 4 indicate that we do not need to use more data to increase the power of the test, we will cover larger variety of biological conditions this way. As a result, we can perform queries that cover only a certain type of experiments. For instance, we can search for co-expressions manifested only in certain tissues or cancer types.

To extend MEM query to multiple platforms, we have to address all the problems outlined before. As a first step toward solution we must combine the expression profiles measuring the same gene in every experiment. There is no correct solution to this task, but we can try some simple options. For example, we can average their expression, select the ones that have largest absolute expression or select the ones with the largest variation. Which aggregation method is the best, is a question that cannot be determined without extensive tests and therefore requires separate study. In the end of this step, we have expression matrices with one expression profile per gene and the genes are comparable between platforms.

Next, we can proceed with the normal ranking queries. That is, given the query gene, we can calculate the rank matrices on each platform where the gene is present. We have to normalize the rankings to the range $(0, 1)$, before combining rank matrices Next, we combine rankings from all the platforms, where the gene is present. As a result we obtain a ranking vector for every gene. The length of the ranking vector might vary. To get comparable BetaMEM scores, we have to rescale the the rank vectors to a common length. The previous results in chapter 5 indicate that linear interpolation of empiric probability distribution should provide optimal result. After rescaling, we can calculate the BetaMEM scores and sort the genes according to them.

The most critical part of the scheme presented above is compressing the expression profiles, which requires separate study. In other respects the algorithm is fairly straightforward and its implementation is our first priority.

8.2 Distance between gene and list of genes

One of the most important applications of MEM is associating genes with some known biological processes. It is usually done by finding genes that are expression wise similar to the the genes involved in the process. We can be more confident in the association, if the gene is similar to several genes with similar function. In a sense, we are trying to find a distance between a gene and a set of genes. Of course, we can do the queries one by one and combine the results afterwards, but it is more reasonable and elegant to calculate the distance straight from the ranking matrices. In the follows we propose a methodology for computing distances from list to gene and vice versa, that can be applied on different occasions.

8.2.1 Estimating distance from gene to list

We start discussion on gene to gene list distance providing a beautiful example. Assume we have collected an extensive database of experiments and biologically relevant gene groups. For example, consider a database of metabolic and signaling pathways, protein complexes and so on. Again user enters a query gene, but instead of genes it lists the functional groups by the expression similarity to the query. That is, the tool finds possible functional associations for a gene and outlines relevant experiments. The tool can give valuable hints about the function, especially if the query gene is otherwise poorly characterized.

For such ranking we can still use the rank matrix produced by MEM, when defining the distance. Given a query gene, we use MEM to calculate the $n \times m$ ranking matrix R. Additionally, we have a group of functionally related genes $S = \{s_1, \ldots, s_k\}$. Let the R_S be a fragment of R that contains only rankings of S elements. All rankings in R_S can be considered as a sample from our mixture distribution with a small and a uniform component. Strictly speaking, ranks in the same columns are not independent but we can neglect this if the size of S is small compared to total number of genes. As a result, we can apply the BetaMEM method directly to this sample, to get the score.

Now note that corresponding score can vary a lot if we calculate distances for groups with different sizes. Therefore, we have to use rescaling methods to transform the ranking vectors into same size. Also we can control the strength of detectable events by carefully selected target ranking vector size on rescaling.

Finally, we can use the feature of BetaMEM that tells us which rankings contributed to the score. We can map these rankings back to the matrix R_S and identify the genes and experiments that contributed mostly to the similarity.

8.2.2 Estimating distance from list to gene

Again let us start from an interesting example: predicting new members to known pathways. That is, we have to find genes that behave similarly to the set of genes. Obviously, we can use the method we just described and perform the distance query for every gene and then use the results for reordering. This means, we have to do thousands of queries which is too expensive computationally. Therefore, we define distance from list to a gene.

There can be different reasons for such queries. First, this kind of query would allow us to naturally combine several measurements of the same gene. More precisely, if a gene is measured multiple times on an array, we can do the query with all the instances simultaneously to get more comprehensive results for the gene. We can also insert genes that are known to interact with each other and try to find more interaction partners.

We base our methods again to the rank matrices. Suppose, we have a set of query genes $S = \{s_1, \ldots, s_k\}$. We can calculate the rank matrices for every each S element. Now there are several ways how to combine individual rankings.

First, we can just merge the rank matrices column wise, so that for every gene the rank vector contains ranks from all the queries. If there is no connection between the genes, then the ranks can be considered independent. Again, we can calculate BetaMEM scores based on those values. Of course the sample size increases very fast if we increase the number of elements in S. As the scores grow quite rapidly with increase of sample size, the increase in sample size can make the scores uninformative. Therefore, we should control the sample size with rescaling of the ranking vector.

The second proposal is a sort of minimum distance with the gene group. We have the MEM rank matrices or every gene in S. Instead of just dumping the rankings into one vector for each gene, we group the rankings based on the dataset. Thus we have for every gene and every experiment k rankings. We obtain the final sample for a gene, if we take the minimal rank from each group. Basically, given a gene we select for each dataset the ranking that shows the largest similarity to one of the genes in S. Of course, with such treatment of ranks we lose the assumption about the uniform distribution of non informative rankings, since every final rank represents a minima over several ranks. As a result we should use more elaborate formula for p-value that counts in the minimum. Nevertheless, we can use other algorithms, such as average ranking or median ranking, for finding similarity score. We cannot estimate the significance of those scores, but we can at least order the results.

The last score is appealing, since the gene that it finds does not have to be similar to the whole group of input genes. It suffices if they are really similar to few members of the group. The last property is very important if we are trying to assign genes to some metabolic or signaling pathways. These pathways are not entirely homogeneous in expression, rather they contain several subsets of genes that behave similarly. Thus, the minimum based distance can give more relevant results in this context, since it does not require similarity to the whole group of genes.

Chapter 9

Conclusions

Gene expression microarray has become a common tool among biologists. As a result, the number of publicly available microarray experiments has recently exploded. As whole genome studies, these experiments contain a wealth of undiscovered information. The main goal of this thesis was to create a tool that allows to explore this information and discover new connections. In particular, we were interested in genes that behave in similar fashion in multiple gene expression experiments.

When creating the tool we encountered several methodological problems that we tried to solve within this thesis. The fist task was to come up with a scheme for combining information from different experiments. We chose a robust scheme, where we leave the individual studies intact, but merge the results of individual similarity queries. As a result, we reduced the task of finding similar genes into rank aggregation problem. Although there are many existing methods none is appropriate for this task. Therefore we had to design our own method for rank aggregation. To study and validate the method we conducted a series of experiments that showed the method performed satisfactorily in most situations.

Secondly, we addressed the question how to determine the significance of obtained results. Namely, we described method for choosing significance thresholds to the score values. Another interesting and practical issue was the dependence between the score and and the number of experiments used in the query. Our aim was to make the score independent of it, so the volume of results would be consistent regardless of the selection of datasets. We proposed two solutions to achieve the goal. The first one is based on the overall distribution of similarity scores. The second method exploits the fact that our similarity score is a statistical test and we can control its sensitivity by changing sample size.

While exploring the data, we noticed that the goodness of results is correlated with the variance of query gene in the experiments used. We studied the hypothesis and there was a strong link between these two quantities. That gave us an idea how to reduce noise in the query by using only datasets, where the gene has large variance. As a practical result we were able to also quantify the needed variance.

One of the practical results of the thesis was the tool. We gave an overview about its features and displayed also some examples about its usage. In discussion we layed a path for further development of the tool. We introduced several new query types that try to answer to biologically relevant questions.

Ko-ekspressiooni päringud üle mitmete eksperimentide

Magistritöö (40AP)

Raivo Kolde

Kokkuvõte

Geeni ekspressiooni mikrokiibid on muutunud bioloogidele igapäevaseks töövahendiks. Seetõttu on viimasel ajal kasvanud plahvatuslikult avalikult kättesaadavate eksperimentide arv. Ülegenoomi uuringutena sisaldavad need andmed palju veel avastamata informatsiooni. Meie töö peamiseks eesmärgiks oli luua tööriist, mis võimaldab kaevandada neid andmestikke ja avastada uusi seoseid. Täpsemalt olime me huvitatud geenidest, mis käituvad väga sarnaselt paljudes geeni ekspressiooni eksperimentides.

Tööriista loomisel kerkis esile mitmeid metodoloogilisi küsimusi, mille lahendamisele ongi antud magistritöö pühendatud. Esmalt tuli välja mõelda, kuidas üldse ühendada informatsiooni erinevatest uuringutest. Meie otsustasime suhteliselt robustse skeemi kasuks, kus me jätame eksperimendid eraldiseisvateks üksusteks, kuid ühendame ainult sarnaste geenide otsingu tulemused. Nii taandus meie sarnaste geenide otsing järjestuste ühendamise ülesandele. Me uurisime põhjalikult olemasolevaid meetodeid selle ülesande lahendamiseks, kuid ükski neist ei olnud meie jaoks sobiv. Seetõttu, me defineerisime uue meetodi järjestuste ühendamiseks. Me testisime antud meetodit nii simuleeritud kui ka reaalsetel andmetel ja jäime tulemusega rahule.

Järgmisena tekkis küsimus, kuidas määrata saadud tulemuste olulisus ja kuhu tõmmata selle piir. Inimene jõuab läbi vaadata vaid paarkümmend kõige sarnasemat geeni, seega pole mõtet näidata tuhandeid tulemusi. Samas on selge, et erinevate päringute puhul on ka tulemuste olulisus erinev, seega näidata iga kord sama arvu vastuseid on ka vale. Sellele probleemile pakkusime välja kaks lahendust. Esimene uuris üleüldist sarnasusskooride jaotust ja määras piirid vastavalt sellele. Teine kasutas ideed, et meie sarnasuskoori saab vaadelda ka statistilise testina, mille võimsust me saame ohjata valimi mahu muutmisega. Mõlemal juhul me uurisime vastavate lahenduste käitumist katsetega nii reaalsetel kui simuleeritud andmetel. Mõlemal lahendusel on omad voorused ning kumba kasutada, sõltub konkreetsest rakendusest.

Andmetega mängides jäi meile silma, et tulemuste headus on otseses seoses päringu geeni hajuvusega kasutatavates andmestikes. Me uurisime seda hüpoteesi lähemalt ja leidsime tõesti selge seose. See andis meile idee päringus esineva müra vähendamiseks: me kasutame ainult andmestikke, milles päringu geeni hajuvus ületab teatud piiri. Me suutsime anda ka hinnangu vastava piiri väärtusele.

Et meie töö peamiseks tulemuseks on ikkagi tööriist, siis andsime ülevaate ka sellest. Lisaks pakkusime näiteid kasutamisest. Diskussiooni osas visandasime plaani tööriista edasise arengu jaoks. Kirjeldatud algoritmid tõid ka hästi esile esialgses disainifaasis tehtud valikute kasulikkuse.

Bibliography

[ALML06] S Aerts, D Lambrechts, S Maity, and P Van Loo. Gene prioritization through genomic data fusion. *Nat Biotechnol*, Jan 2006.

[BGL$^+$00] M P Brown, W N Grundy, D Lin, N Cristianini, C W Sugnet, T S Furey, M Ares, and D Haussler. Knowledge-based analysis of microarray gene expression data by using support vector machines. *Proc. Natl. Acad. Sci. U.S.A.*, 97(1):262–7, Jan 2000.

[Bor81] J.C. Borda. Mémoire sur les élections au scrutin. *Histoire de l'Académie Royale des Sciences*, pages 42–51, 1781.

[BRS$^+$01] A Bhattacharjee, W G Richards, J Staunton, C Li, S Monti, P Vasa, C Ladd, J Beheshti, R Bueno, M Gillette, M Loda, G Weber, E J Mark, E S Lander, W Wong, B E Johnson, T R Golub, D J Sugarbaker, and M Meyerson. Classification of human lung carcinomas by mrna expression profiling reveals distinct adenocarcinoma subclasses. *Proc. Natl. Acad. Sci. U.S.A.*, 98(24):13790–5, Nov 2001.

[BTT89] J. Bartholdi, CA Tovey, and MA Trick. Voting schemes for which it can be difficult to tell who won the election. *Social Choice and Welfare*, 6(2):157–165, 1989.

[BTvOM07] Tijl De Bie, Léon-Charles Tranchevent, Liesbeth M M van Oeffelen, and Yves Moreau. Kernel-based data fusion for gene prioritization. *Bioinformatics*, 23(13):i125–32, Jul 2007.

[BTW$^+$07] Tanya Barrett, Dennis B Troup, Stephen E Wilhite, Pierre Ledoux, Dmitry Rudnev, Carlos Evangelista, Irene F Kim, Alexandra Soboleva, Maxim Tomashevsky, and Ron Edgar. Ncbi geo: mining tens of millions of expression profiles–database and tools update. *Nucleic Acids Res.*, 35(Database issue):D760–5, Jan 2007.

[CFK$^+$07] Michele Clamp, Ben Fry, Mike Kamal, Xiaohui Xie, James Cuff, Michael F. Lin, Manolis Kellis, Kerstin Lindblad-Toh, and Eric S. Lander. Distinguishing protein-coding and non-coding genes in the human genome. *Proceedings of the National Academy of Sciences*, page 0709013104, 2007.

[Con] International Genomics Consortium. Expression project for oncology. http://www.intgen.org/expo.cfm.

[dC85] M. de Condorcet. Essai sur l'application de l'analysea la probabilite des decisions renduesa la pluralite des voix. *Imprimerie Royale, Paris*, 1785.

[DG77] P. Diaconis and R. Graham. Spearman's footrule as a measure of disarray. *Journal of the Royal Statistical Society, Series B*, 39(2):262–268, 1977.

[DKNS01] C Dwork, R Kumar, M Naor, and D Sivakumar. Rank aggregation revisited. *Proceedings of WWW10*, Jan 2001.

[FKS03] Fagin, R Kumar, and D Sivakumar. Comparing top k lists. *SIAM J. Discrete mathematics*, page 27, Feb 2003.

[FLNP00] N. Friedman, M. Linial, I. Nachman, and D. Pe'er. Using Bayesian Networks to Analyze Expression Data. *Journal of Computational Biology*, 7(3-4):601–620, 2000.

[HMJ$^+$00] T R Hughes, M J Marton, A R Jones, C J Roberts, R Stoughton, C D Armour, H A Bennett, E Coffey, H Dai, Y D He, M J Kidd, A M King, M R Meyer, D Slade, P Y Lum, S B Stepaniants, D D

Shoemaker, D Gachotte, K Chakraburtty, J Simon, M Bard, and S H Friend. Functional discovery via a compendium of expression profiles. *Cell*, 102(1):109–26, Aug 2000.

[IWS$^+$05] Rafael A Irizarry, Daniel Warren, Forrest Spencer, Irene F Kim, Shyam Biswal, Bryan C Frank, Edward Gabrielson, Joe G N Garcia, Joel Geoghegan, Gregory Germino, Constance Griffin, Sara C Hilmer, Eric Hoffman, Anne E Jedlicka, Ernest Kawasaki, Francisco Martínez-Murillo, Laura Morsberger, Hannah Lee, David Petersen, John Quackenbush, Alan Scott, Michael Wilson, Yanqin Yang, Shui Qing Ye, and Wayne Yu. Multiple-laboratory comparison of microarray platforms. *Nat Meth*, 2(5):345–350, May 2005.

[Kem59] J. Kemeny. Mathematics without numbers. *Daedalus*, 88(575-591):8j, 1959.

[KEN38] M.G. KENDALL. A new measure of rank correlation. *Biometrika*, 30(1-2):81–93, 1938.

[KvBV$^+$02] Patrick Kemmeren, Nynke L van Berkum, Jaak Vilo, Theo Bijma, Rogier Donders, Alvis Brazma, and Frank C P Holstege. Protein interaction verification and functional annotation by integrated analysis of genome-scale data. *Mol. Cell*, 9(5):1133–43, Jun 2002.

[LDB$^+$06] Karen Lemmens, Thomas Dhollander, Tijl De Bie, Pieter Monsieurs, Kristof Engelen, Bart Smets, Joris Winderickx, Bart De Moor, and Kathleen Marchal. Inferring transcriptional modules from chip-chip, motif and microarray data. *Genome Biol.*, 7(5):R37, Jan 2006.

[LHS$^+$04] Homin K Lee, Amy K Hsu, Jon Sajdak, Jie Qin, and Paul Pavlidis. Coexpression analysis of human genes across many microarray data sets. *Genome Res.*, 14(6):1085–94, Jun 2004.

[RKP$^+$07] J Reimand, M Kull, H Peterson, J Hansen, and J Vilo. g:profiler–a web-based toolset for functional profiling of gene lists from large-scale experiments. *Nucleic Acids Res.*, May 2007.

[RKSM$^+$07] Daniel R Rhodes, Shanker Kalyana-Sundaram, Vasudeva Mahavisno, Radhika Varambally, Jianjun Yu, Benjamin B Briggs, Terrence R Barrette, Matthew J Anstet, Colleen Kincead-Beal, Prakash Kulkarni, Sooryanaryana Varambally, Debashis Ghosh, and Arul M Chinnaiyan. Oncomine 3.0: genes, pathways, and networks in a collection of 18,000 cancer gene expression profiles. *Neoplasia*, 9(2):166–80, Feb 2007.

[Saa99] D Saari. Explaining all three-alternative voting outcomes. *Journal of Economic Theory*, Jan 1999.

[Scu07] D Sculley. Rank aggregation for similar items. *eecs.tufts.edu*, Jan 2007.

[SFKR04] Eran Segal, Nir Friedman, Daphne Koller, and Aviv Regev. A module map showing conditional activity of expression modules in cancer. *Nat. Genet.*, 36(10):1090–8, Oct 2004.

[SSKK03] Joshua M Stuart, Eran Segal, Daphne Koller, and Stuart K Kim. A gene-coexpression network for global discovery of conserved genetic modules. *Science*, 302(5643):249–55, Oct 2003.

[TDO$^+$03] Olga G Troyanskaya, Kara Dolinski, Art B Owen, Russ B Altman, and David Botstein. A bayesian framework for combining heterogeneous data sources for gene function prediction (in saccharomyces cerevisiae). *Proc. Natl. Acad. Sci. U.S.A.*, 100(14):8348–53, Jun 2003.

[THNC02] Robert Tibshirani, Trevor Hastie, Balasubramanian Narasimhan, and Gilbert Chu. Diagnosis of multiple cancer types by shrunken centroids of gene expression. *Proc. Natl. Acad. Sci. U.S.A.*, 99(10):6567–72, May 2002.

[VDS$^+$07] Imre Vastrik, Peter D'Eustachio, Esther Schmidt, Geeta Joshi-Tope, Gopal Gopinath, David Croft, Bernard de Bono, Marc Gillespie, Bijay Jassal, Suzanna Lewis, Lisa Matthews, Guanming Wu, Ewan Birney, and Lincoln Stein. Reactome: a knowledge base of biologic pathways and processes. *Genome Biol.*, 8(3):R39, Jan 2007.

[WBW$^+$07] Ido Wolf, Shikha Bose, Elizabeth A Williamson, Carl W Miller, Beth Y Karlan, and H Phillip Koeffler. Foxa1: Growth inhibitor and a favorable prognostic factor in human breast cancer. *Int J Cancer*, 120(5):1013–22, Mar 2007.

[WKB05] Cecily J Wolfe, Isaac S Kohane, and Atul J Butte. Systematic survey reveals general applicability of "guilt-by-association" within gene coexpression networks. *BMC Bioinformatics*, 6:227, Jan 2005.

[ZMC$^+$04] Wen Zhang, Quaid D Morris, Richard Chang, Ofer Shai, Malina A Bakowski, Nicholas Mitsakakis, Naveed Mohammad, Mark D Robinson, Ralph Zirngibl, Eszter Somogyi, Nancy Laurin, Eftekhar Eftekharpour, Eric Sat, Jörg Grigull, Qun Pan, Wen-Tao Peng, Nevan Krogan, Jack Greenblatt, Michael Fehlings, Derek van der Kooy, Jane Aubin, Benoit G Bruneau, Janet Rossant, Benjamin J Blencowe, Brendan J Frey, and Timothy R Hughes. The functional landscape of mouse gene expression. *J. Biol.*, 3(5):21, Jan 2004.