GENKOMB project report

Jakub Orzechowski Westholm, Adam Ameur

12th September 2003

Abstract

The aim of the GENKOMB project was to find and analyse transcription factor binding sites in the human genome, by correlating expression data for a set of genes with the nucleotide sequences in their upstream regions. This document is a technical description of tools that have been developed and results that have been obtained in the GENKOMB project. Apart from detecting transcription factor binding sites, this project has resulted in programs for matching weight matrices to DNA sequences and multiple alignment of short nucleotide sequences.

SICS Technical Report T2003:14
ISSN 1100-3154
ISRN:SICS-T–2003/14-SE

Contents

1 Introduction

The idea behind all this work is to detect transcription factor binding sites (motifs) in the human genome, by correlating expression data for a set of genes with the nucleotide sequece in the promoter regions of those genes. For this, the REDUCE method [1] was used. It fits a linear model to the data, in which expression levels are explained by the presence of different motifs in the upstream regions. The most significant motifs are selected and the fitting parameters indicate the function of the significant motifs: activating or repressing.
The REDUCE method has previously been used on yeast. In this project we tried to use it on human data, provided by AstraZeneca. It consists of the following: expression data (from cortex stem cells and the cardio-vascular system), sequence data (from the human genome) and a mapping (between sequence and expression data). The sequence data is public, whereas expression data and the mapping between expression and sequence data are not.
This document describes all work related to the GENKOMB project, starting with a brief explanation of some concepts from biology. Then, the pre-processing and REDUCE method are described. The topic of the following section is the analysis and further applications of results obtained from the REDUCE method. Then, the main results are summarized, and the last section points to some possible extensions of the project.
There are two appendecies to this document with some supplementary information. Appendix A describes an entropy based approach for matching DNA sequences with weight matrices, and a further application of multiple alignment of short DNA sequences. Appendix B shows results of some tests that we have performed regarding the quality of transcription starts.

2 Some biological concepts

This section is a short description of some biogical preliminaries, mainly concerning the (human) genome and its represenation. The purpose is just to explain some concepts used in this report, not to provide an introduction to molecular biology or genomics. The human genome is represented as a sequence of about $3 * 10^9$ nucleotides, divided into 46 chromosomes. Each chromosome consists of DNA, which is a double stranded molecule. It is possible to represent the genome by only one sequence since the strands are complmentatary, so that information about one strand is sufficient to determine the other. There are four different nucleotides: *Adenine*, *Cytosine*, *Guanine* and *Thymine*, represented by the letters A, C, G and T. The DNA is not symmetrical, so if one looks at a DNA strand the two ends are different. One end is called the 5' and the other 3'. Therefore the genome

can be thought of as a string over the alphabet $\{A, C, G, T\}$ typically going from 5' to 3'.. Sometimes it is convinient to talk about degenerate sequences, where a postion may for example be either C or G. For this so called *IUPAC codes* are used. Besides A, C, G and T there are codes for all combinations of bases. The full table (taken from [20]) is given below:

IUPAC code	nucleotides
A	A
C	C
G	G
T	T
R	G or A
Y	C or T
K	G or T
M	A or C
S	C or G
W	A or T
B	C or G or T
D	A or G or T
H	A or C or T
V	A or C or G
N	A or C or G or T

Table 1: The IUPAC codes.

When the genome was sequenced, it was divided into several smaller parts called *contigs*. Each such contig was sequenced individually. When the genomic sequence is stored in databases such as [16] it is stored contig by contig. The sequenced contigs have then been ordered and combined to give the full genome. This ordering of the contigs is called *golden path*. The golden path is used to position features, such as genes, in the genome. It is beyond the scope of this report to discuss the concept of genes in depth. But simply put, a gene is a region in the DNA that is being copied (transcribed) to mRNA. The mRNA then serves as a blueprint for the proteins being produced in the cell. There are approximately 30000 genes throughout the human genome, with known positions relative to the golden path. Each contig also has a postion (start and end) relative the goden path. Thus knowing the golden path postion of a contig and of a feature X, one can compute the postion of the feature relative the contig. Since a gene can be on either DNA strand, it can be in the same direction as the golden path, or in the oposite direction. The region just before the transcription start of a gene (relative to the gene) is called the *upstream region*.

3 Pre-processing

Besides implementeing the REDUCE method and analyzing the results, a large part of this project has been about pre-processing the data. That is, to format the data, insert it to databases and extract the important parts. This section is a step by step description of how the original data was transformed into a form suitable as input to the REDUCE program. The pre-processing requires the following data as input:

- **Sequence data** The file 'contigs', with contigs from the human genome. The 'contigs' file is almost on ensembl format [16], but it also contains the fields 'CH', 'CP' and 'LC' which are not ensembl standard. Moreover, the entries in the file are not properly delimited.
- **Expression data** Expression levels in human cells for a set of probes, in the files 'cortex' and 'cv'. The expression files contain expression levels for the Affymetrix HG-U133A and HG-U133B probes [12] in two different cell samples (cortex stem cells and cardio-vascular) undergoing different treatments. For each probe there are measurements from 81 experiments for the cortex stem cells, and about 600 experiments for the cardio-vascular cells.
- **Mapping data** The file 'coords', a mapping from probes to positions in the sequence data. The map entires in the file are delimited by 'XX'.
- **Motifs** A set of putative transcription factors.
- **Filtered genes** A subset of all probes.

The sequence, expression and mapping data are provided by AstraZeneca, whereas we have to create our own data for motifs and filtered genes. In the pre-processing described below we assume that we already have a set of motifs and some filtered genes. Later, in subsections 3.4 and 3.5 we show how the motifs and probes are found.
The implementation of the pre-processing was mainly done in Perl, using the Bioperl package [8].

3.1 Storing data in databases

Some of the data provided by AstraZeneca is hard to handle because of its size unless it is stored in databases. Our way of solving such problems is to store data as fasta [18] or ensembl [16] files, and then use built-in Bioperl functions to

index them (with the program index.pm). The indexing makes it possible to extract information from the databases quickly. Starting with the files 'contigs' with sequence data for the human genome, 'coords' with a mapping from Affymetrix probes to the genome and one of the files 'cortex' or 'cv' with expression levels, the following databases are set up.

- **contigs.embl** The database contigs.embl contains the same information as the 'contigs' file, but on ensembl format. We modify the 'contigs' file by adding the tag '//' after every entry, and storing information in the fields 'CH' (chromosome), 'CP' (contig path) and 'LC' (?) in the field 'DE' (description). The program make_embl.c builds the database.

- **mapping.fasta** The 'coords' file contains a mapping from the probes on the Affymetrix u133 chip to the human genome. In the 'coords' file, a map is represented as an entry with a probe ID (ID), chromosome (CH), gene path (CP) and a list of contigs (CT). The gene path contains start and stop position relative to the golden path for the gene that the probe was sampled from. The contig list is all contigs that the gene stretches over. There are 44928 probes on the u133 chip, but only 15248 of them have maps. We store all maps in a fasta file, mapping.fasta, with the following header: '>ID CP CH CT'. When extracting information from the database, ID is the key and (CP,CH,CT) the value. The mapping database is created by the program build_id_mapping.pm.

- **cortex.fasta or cv.fasta** These databases consist of expression levels taken from the files 'cortex' and 'cv'. An entry in one of the databases has the following fasta header: '>NR.ID E', where NR is the experiment number (0-80 in 'cortex' and 0-600 in 'cv') and ID is the probe ID. E is created from the original expression levels, by first taking the logarithm of the original expression level and then subtracting the mean of the expressions for the probe in all experiments. When this is done, the sign of the expression for a probe in an experiment will tell something about the how the experiment affects (represses or increases) the expression of the probe. The database makes it possible to extract the expression level for a probe in some experiment quickly. Only entries for which the probe IDs are present in the mapping are stored in the expression databases. The program setup_expression.sh sets up the expression database.

- **contig_mapping.fasta** A contig ID is a string that consists of several smaller parts, separated by '.', for example 'AC026374.19.1.75385'. Sometimes we only have the prefix of a contig when we in fact need the whole string. Therefore it is convenient to have a mapping from prefixes to all contig IDs

that start with that prefix. The file contig_mapping.fasta, created by the program contig_mapping.pm, contains such a mapping.

3.2 Building an upstream sequence database

When the databases in section 3.1 have been properly set up, another database with upstream nucleotide sequences for all genes is constructed. A new upstream database can be created for every possible length and position of the upstream region.

We find the upstream regions by going through all entries in the file mapping.fasta in the following way (in the program upstream.pm). For every probe, we go through the list of contigs (that is, all contigs in the CT field) and select the contig that contains the gene start, since that contig is the one that contains the upstream region. In the next step, the upstream sequence is extracted from the contig. There are two different cases when this is done, depending on the direction of both the gene and the contig relative to golden path. The output is written to a fasta file where an entry is on the form: '> GSTART-CH-GSTOP' followed by a sequence of nucleotides. GSTART and GSTOP is the start and stop of a gene, CH is the chromosome where the gene is and the sequence of nucleotides is the upstream sequence from 5' to 3'. Since many probes may be sampled from the same gene, this database contains only 9150 entries, which is much less than the number of probes in the mapping. For a few probes, the contigs contining the gene start is missing.

The script upstream.sh sets up an upstream database, and is called with three arguments: UP, DOWN and REPEAT. UP is the start position of the upstream sequence and DOWN is the stop position of the upstream sequence, relative to the transcription starts. For example, if UP is 1000 and DOWN is 100, then the upstream sequences in the database will be of length 1100, ranging from 1000 bases upstream of the transcription start to 100 bases downstream. If the optional flag REPEAT is set to '_r', then the upstream database is masked for repeats with the program RepeatMasker [2]. The RepeatMasker program replaces nucleotides contained in DNA repeat sequences with the letter 'N'. The upstream database is stored in the file 'upstream_u_UPd_DOWN.fasta'. If repeat masking is performed, then the masked database is stored in the file 'upstream_u_UPd_DOWN_r.fasta'.

3.3 Databases for filtered genes

As mentioned above, we assume that we have a set of filtered genes. The filtered genes are the ones that later will be used in the REDUCE model. It is efficient to store information about the upstream sequences and expression levels for the

filtered genes in separate databases. For example, if we want to run the REDUCE program many times on the same filtered genes but with different putative motifs, then we will only have to create the upstream and expression databases for the filtered genes once. Below is a description of the databases.

- **Filtered upstream database** When creating the filtered upstream database, the parameters UP, DOWN and the file FILTER with filtered genes are given. The filtered upstream database is created from the corresponding upstream sequence database (with correct values on UP and DOWN) by only selecting entries from filtered genes. The program upstream_filter.pm does this. The filtered upstream database is stored in 'FILTER_uUPd_DOWN.fasta', or in 'FILTER_uUPd_DOWN_r.fasta' if the repeat mask option was selected.

- **Filtered expression database** The filtered expression database is created by the program extract_expressions.pm, from the original expression database together with the file FILTER containing the filtered genes. The result is a directory with the name FILTER, in which one file is created for each of the experiments. The files are called 'experiment_NR', where NR is ranging from 0-80 for cortex and 0-600 for cv. Each file contains expression levels only for the filtered genes, and for a given experiment.

3.4 Motifs

The REDUCE method requires a set of putative motifs, and we have two ways of building such sets. The motifs are either taken from a public database or generated as fixed DNA sequences.

Transcfac

The database TransFac [5] contains information about known transcription factor binding sites. A problem with TransFac is that the database of transcription factors is far from complete and sometimes there is a lot of uncertainty about the motifs. Therefore only the motifs from TransFac that have additional information in the form of weight matrices (see Appendix A for a definition). Such motifs are more more reliable than the others, since they are based on a larger number of observations. TransFac contains about 350 weight matrices.

Fixed motifs

Another approach is to look at all fixed motifs of a certain length, that is to only look at sequences of A,C,G and T [1]. An advatantage of this approach is that we are always guaranteed to find the best motif if we search through all possible ones. The problem with this method however is that the number of possible motifs is too big for the REDUCE program. In the current setup, matching about 10000 motifs to 10000 upstream sequences takes about a day. Therefore we somehow have to restrict the set of possible motifs. One way of doing this is to only consider motifs between a minumum and maximum length. Currently we look at motifs of length 4 to 7. This would give

$$4^4 + 4^5 + 4^6 + 4^7 = 21720 \text{ motifs}$$

Moreover, if two sequences are each others reverse complement, we only use one of them (the first one in lexical order). This reduces the number of motifs by almost a factor of 2, down to 10920.

3.5 Filtered genes

The set of filtered genes is just given as a file with probe IDs. In most cases, we let the set of filtered genes be all probes in the expression file cortex.fasta or cv.fasta. That is, the REDUCE program is most often run on all probes that have a present call in at least one of the experiments. However, it is also possible to run the REDUCE program on a smaller set of genes, which may be interesting when searching for motifs that explain a specific group of correlated genes.

3.6 Matching transcription factors

The REDUCE method requires a set of possible motifs together with a set of filtered genes as input. The motifs are matched against upstream sequeces of the genes to compute how often each motif occurs in the upstream sequence of a gene. More formally, the result of the matching is a matrix $Hits$, where $Hits(i, j)$ is the number of occurences of motif i in the upstream sequence of gene j. This matrix, along with the expression levels for each gene, is the input to the REDUCE program.
For now, we assume that we already have a set of putative motifs and a set of genes. The motifs are given as weight matrices, and the genes are just a list of probe IDs. The motifs are matched against the upstream sequences using the program 'wm_match', which computes the number of matches of a weight matrix

[1] These fixed sequences are also represented as weight matrices, containing only 0's and 1's.

(and its reverse complement) to a nucleotide sequence. Details of how the weight matrix matching is implemented is given in a Appendix A.
The matching of all possible motifs against the upstream sequences is handled by the script match.sh. This script requires a database with all upstream sequneces of the filtered genes. Given those sequences, it runs the 'wm_match' program for some given set of possible motifs. The results of 'wm_match' is then formatted and merged (by the programs translate_tf.pm and merge_results.pm) into a file on the following form:

```
> GENE 1
  MOTIF 1: [SEQUENCE,  NR OF MATCHES TO GENE 1]
  ...
  MOTIF M: [SEQUENCE,  NR OF MATCHES TO GENE 1]
...
> GENE N
  MOTIF 1: [SEQUENCE,  NR OF MATCHES TO GENE N]
  ...
  MOTIF M: [SEQUENCE,  NR OF MATCHES TO GENE N]
```

For each probe only the motifs with at least one match are listed in the file above. That is, all zero elements of the $Hits$ matrix are removed.

3.7 Putting everything together

The last part of the pre-processing is to create the final input to the REDUCE program from a match file and the filtered expression database. The main reason for doing this is that we want to replace probe IDs and motif names by unique numbers. From the match file and the file with filtered genes , we create four new files.

- **probe_map.fasta** A file with a mapping from probe IDs to numbers (created by make_probe_map.pm).
- **reduce_data.map** A file with a mapping from numbers to motifs (created by make_input.pm).
- **reduce_data.hit** The same information as in the match file, but probe and motif IDs have been replaced by numbers (created by make_input.pm).
- **experiment_NR.exp** One file for each experiment. Contains the expression levels for all filtered probes in the experiment (created by make_exp.pm).

3.8 Running the pre-processing

The entire pre-processing is executed by the following command:

```
>./pre_process.sh [Up] [Down] [Motif_dir] [Exp_name] [Filter_file]
```

Here Up is the length of the upstream sequence and Down is the length of the downstream sequence. Motif_dir is a directory containing files with putative transcription factors. Exp_name is the experiment name, either 'cortex' or 'cv'. Filter_file is the name of the filter file (in the filter_db directory). If no Filter_file is given, then a default filter called 'cortex' or 'cv' is used. The default filter file contains all probes in either the 'cortex.fasta' or 'cv.fasta' expression file. In all results in this paper, we use the default filter file 'cortex', containing 9150 probes.

4 REDUCE

A central part of the GENKOMB project is an iterative method called REDUCE, for finding motifs in upstream regions by correlating sequence data with expression. This section gives an introduction to the REDUCE method, and a description of our implementation. It shall be noted that the REDUCE method can only be applied to gene expression levels in a single sample. See [1] for a more detailed description of the method.

4.1 The model

The idea behind REDUCE is the simplified assumption that the expression level of a gene is a linear function of all motifs in its upstream region. This assumption gives a model

$$A_g = C + \sum_{\mu \in M} F_\mu N_{\mu g}$$

where A_g is the expression level of gene g, M is the set of significant motifs and $N_{\mu g}$ is the number of occurences of motif μ in gene g's upstream region. C is a base level expression, the same for all genes. Finally, F_μ is the increase/decrease in transcription caused by the presence of motif μ. The expression levels A are normalized and centered around zero, so that

$$A_g = \delta a_g / (\mid G \mid < \delta a^2 > \mid)^{1/2}$$

where a_g are expression levels for a certain sample produced in the pre-processing. $< X >= (1/ \mid G \mid) \sum_{g \in G} X_g$ is the average quantity of X over all genes in G and $\delta a_g = a_g - < a >$ is the deviation of a_g from the mean. Thus, $< \delta a^2 >= var(a)$

is the variance of a. The same rescaling operations are applied to each vector N_μ (the number of occerences of a certain motif μ in the upstream sequences of the genes):

$$N_{\mu g} = \delta n_{\mu g}/(\mid G \mid < \delta n_\mu^2 >|)^{1/2}$$

Here $n_{\mu g}$ is the original number of occurrences of motif μ in the upstream sequence of gene g found in the pre-processing, $\delta n_{\mu g} = n_{\mu g} - < n_{\mu g} >$, and the mean and variance are defined the same way as above.
It is possible to define an error measure $\chi^2(M)$ for every set of motifs M, that tells how good the motifs in M fit the model. If A_g is the expression as defined above and $E_g(M)$ is the expression that the model gives for a gene g, then

$$\chi^2(M) = \sum_{g \in G} (A_g - E_g(M))^2$$

The model and this error estimate makes it possible to construct an iterative method for finding significant motifs.

4.2 The algorithm

REDUCE is an iterative method, similar to principal component analysis. It requires a set of motifs M_{all} as indata, together with the number of occurences of each motif in the genes upstream regions.
The set M will become the set of significant motifs, initially $M = \varnothing$. In every iteration, we go through all motifs in M_{all}. If $M_{all} = \{\mu_1, \mu_2, .., \mu_m\}$, then all models $M \cup \{\mu_1\}, M \cup \{\mu_2\}, ..., M \cup \{\mu_m\}$ are tried. The model $M \cup \{\mu_i\}$ that gives the lowest χ^2 value is selected, and $M = M \cup \{\mu_i\}$, $M_{all} = M_{all} - \{\mu_i\}$. Thus, one new motif is added in each iteration. The analogy with principal component analysis holds, since adding a motif to a model that already contains an identical motif doesn't give any reduction of χ^2. Adding a similar motif can at most give a small reduction.
The iterations continue until no motif gives a significant reduction of the χ^2 value. The result is a list of motifs, as shown in Table 2:

consensus sequence	*motif*	$\Delta\chi^2$	F	*genes*	*hits*
GTCGA	μ_A	0.0083	0.0200	1822	2534
WWTWMTR	μ_B	0.0064	-0.0238	5304	24533
NNNNCCGGAARYNN	μ_C	0.0047	0.0187	891	1050
CTCGTT	μ_D	0.0031	0.0151	991	1104

Table 2: Example of REDUCE output

In this example, four significant motifs are found and $\Delta\chi^2$ is the reduction of χ^2 when a motif is added to the model. By looking at the sign of F, we see that μ_B is the only motif that represses expression, the other ones have an activating effect. The column $genes$ shows the number of genes that contain each motif in their upstream sequence, whereas $hits$ is the total number of matches for each motif. The $consensus\ sequence$ is just a IUPAC symbol representation of each motif.

4.3 Executing the program

Our implementation of the reduce program is written in C, and it consists of several modules. The easiest way of running the program is to execute the script reduce.sh by the following command:

```
>./reduce.sh [Expression_file] [Map_file] [Hit_file] [Mode]
```

The required files must be on the following format.

- **Expression_file** A file with expression levels for the probes.

  ```
  probe(PROBE_ID1,EXPRESSION,CALL)
  ...
  probe(PROBE_IDN,EXPRESSION,CALL)
  ```

  CALL is one of the letters 'A' (absent), 'P' (present) or 'M' (marginal).

- **Map_file** A file with a mapping from numbers to motifs.

  ```
  nr probes:N
  nr motifs:M
  motif(MOTIF_ID1,NAME,CONSENSUS,DESC,NR_PROBES,NR_HITS)
  ...
  motif(MOTIF_IDM,NAME,CONSENSUS,DESC,NR_PROBES,NR_HITS)
  ```

  NAME is a unique name for the motif, CONSENSUS is the IUPAC consensus sequence, DESC is a name that can give a biological information of the motif, NR_PROBES is the number of probes that have the motif in its upstream dequence, and NR_HITS is the total number of occurences.

- **Hit_file** A file with the number of occurences of each motif in the upstream sequence of every gene.

```
probe(PROBE_ID1)
motif(MOTIF_ID1,nr_matches(PROBE_ID1,MOTIF_ID1))
...
motif(MOTIF_IDM,nr_matches(PROBE_ID1,MOTIF_IDM))
...
probe(PROBE_IDN)
motif(MOTIF_ID1,nr_matches(PROBE_IDN,MOTIF_ID1))
...
motif(MOTIF_IDM,nr_matches(PROBE_IDN,MOTIF_IDM))
```

nr_matches(i,j) is the number of occurences of j in i's upstream sequnce.

- **Mode** An optional flag that may be set to one of the strings '-motifs', '-hits', '-search', '-start'. The program is run in different modes depending on the flag. If no flag is given, then the program is run in default mode. The main difference between the modes is how the output is presented.

4.4 Output

As described in section 4.3, our implementation of REDUCE may be run in different modes, resulting in different kinds of output. However, the direct output of the program is only interesting when the program is run in default mode, without any flag. Therefore, we only describe the default output from reduce in this section. Later, in sections 5.1, and 5.2 we will see why running the program in other modes can be useful.
In the default mode, output from the program is given as a table, written to stdout: In this table, there are two F values. F_{single} is the best value when fitting a motif

$consensus\ sequence$	$motif$	$\Delta\chi^2$	F_{single}	F_{multi}	$genes$	$hits$
GTCGA	μ_A	0.0083	0.0200	0.0172	1822	2534
WWTWMTR	μ_B	0.0064	-0.0238	-0.0259	5304	24533
NNNNCCGGAARYNN	μ_C	0.0047	0.0187	0.0204	891	1050
CTCGTT	μ_D	0.0031	0.0151	0.0097	991	1104

Table 3: The result of running REDUCE in default mode.

to the model, and F_{multi} are other values of F, obtained by simultaneously fitting all significant motifs. If the values of F_{single} and F_{multi} are different, this may suggest that the significant motifs are not independent.
It may be hard to extract all meaningful information from the output tables that we get by running REDUCE on several samples by a manual inspection. Therefore,

we have developed a method for visualizing the output which is shown in Fig. 5.2. Hopefully, this makes it easier to identify motifs that may have a repressing or activating effect in specific samples. As we will see in the next section, there are also other possible applications.

5 Applications

There are many ways to further investigate the results obtained from the REDUCE method. In this section the following applications are discussed:

- To cluster samples on motifs found by REDUCE.
- To find new motifs using REDUCE.
- To cluster the genes on the motifs found by REDUCE.

In all applications we use expression data from 71 cortex cell samples undergoing various treatments, as shown in Fig. 5.1. The original cortex data file contains 81 samples; the file with only 71 samples is created by merging some of the duplicates.

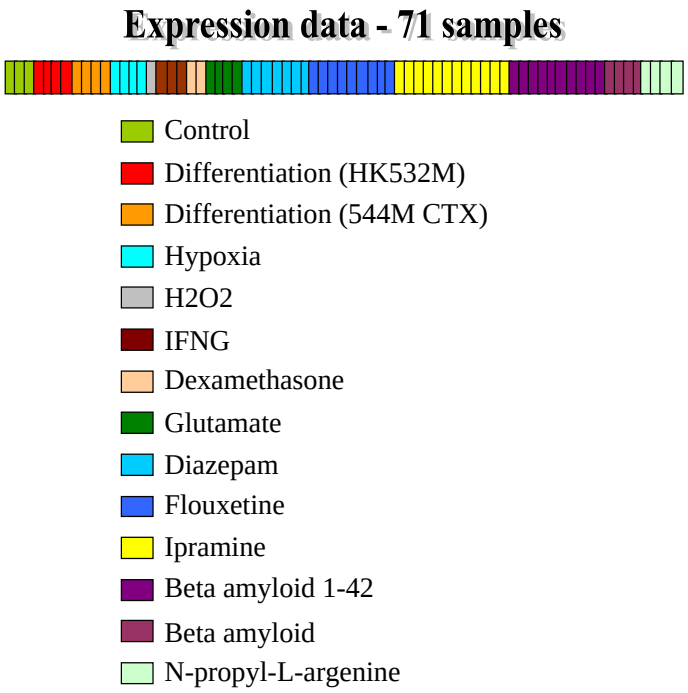

Figure 5.1: 71 expression data samples from cortex stem cells.

5.1 Clustering samples on motifs and on expression

One way to use the motifs found by the REDUCE method is to cluster the samples on them, to detect groups of samples that are governed by the same transcription factors. The starting point is to run REDUCE on all samples, so that significant motifs are detected in each sample. We have used all matrices from Transfac as the set of putative motifs (see section 3.4), and the 71 cortex samples as expression data. The idea is to associate a vector of significant motifs to each sample. The elements of the vectors are either the F-value of the motif for that sample (if REDUCE found that motif in the sample) or 0 (if not detected by REDUCE).
Then the program arrange [11] is used, which performs hierarcical clustering on both samples and significant motifs and then rearranges all columns and rows so that similar samples and motifs are close to each other. The advantage of the arrange program is that it makes it easy to get an overview of the results, and perhaps identify biologically meaningful motifs. The script exp_cluster_setup.sh performs clustering with the arrange program, and the resulting files are viewed with the program TreeView [10]. The result is shown in Fig. 5.2.

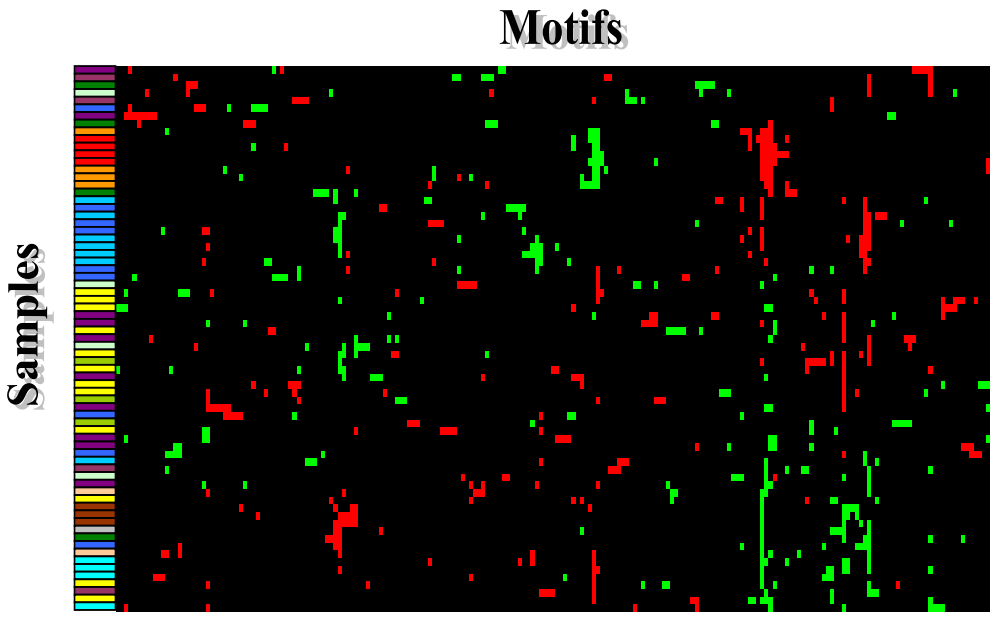

Figure 5.2: The 71 cortex stem cell experiments clustered on significant motifs. A green dot indicates that a motif has a repressing effect in a sample, whereas red dots indicate activation. Samples are colour coded based on their treatmens as in Fig. 5.1. The figure shows several motifs that discriminate between different treatments.

The clustering of samples on motifs was also compared to a clustering of the samples on expression. To obatain this, the Superparamagnetic Clustering algorithm [6] was used[2]. The shell script cluster.sh sets up the necessary files (using the perl program pre_clustering.pm), creates a new directory for the output and then runs the SW program. The parameters used in the clustering were

- Lowest temperature = 0.0
- Highest temperature = 0.5
- Temerature steps = 0.005
- SW cycles = 2000

Then the program stable_clusters.pm was used to extract the clusters that were stable over the longest period of time from the outdata of SW. The stable clusters were examined by eye and compared to what was known about the samples. Fig. 5.3 shows that the clustering on expression is rather similar to the one on motifs, but not as informative.

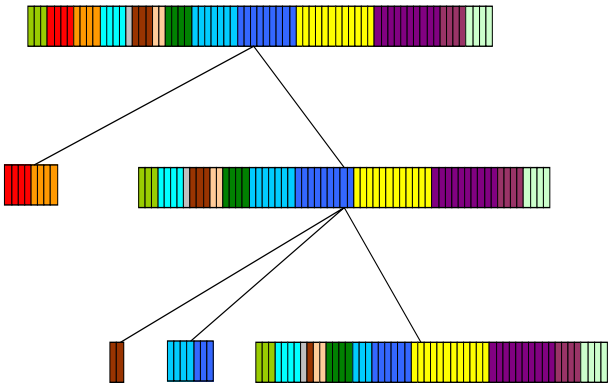

Figure 5.3: The 71 cortex stem cell experiments clustered on expression data. The figure shows how stable clusters are decomposed into sub clusters. Samples are colour coded based on their treatmens as in Fig. 5.1.

Some conclusions from clustering the samples:

- Samples with similar treatments cluster together on motifs (as seen in Fig. 5.2). This suggests that it is possible to detect motifs that discriminate between grops of samples, and such motifs can be identified in a heatmap such as the one in Fig. 5.2.

[2]The reason for this is that the arrange program cannot handle the large number (9150) of dimensions in the expression data.

- The REDUCE method is not too sensitive to small changes in the expression profiles. If it was, it would be hard to find motifs discriminating between different treaments.
- Samples with similar treatments cluster together on expression (as seen in Fig. 5.3) but the clustering is not as good as the clustering on motifs. The reason for this is probably that the clustering on expression involves too many dimensions. In this context REDUCE can be seen as a form of principal component analysis: We detect about 100 dimensions (the motifs) which give a similar clustering as the original approximately 10000 dimensions (the expressions of the genes).

5.2 Unconditional search for new motifs

The REDUCE method can also be used as a backbone when performing an unconditional search for new motifs. The idea is to run the method on all samples and on all fixed motifs of length 4 to 7 (see section 3.4), to obtain a set of significant motifs. We then assume that significant motifs that are similar in fact are instances of the same TF binding site. Therefore, we combine such similar motifs into weight matrices. The weight matrices can then be further improved by a local search technique. Hopefully, some of the resulting weight matrices represent TF binding sites that are present in the human genome. A more detailed description of the methods involved is given below.

Finding the motifs

Since we are interested in finding similar motifs, i.e fixed motifs that are predicted by REDUCE to have the same functionallity, REDUCE is run in a special mode called start guess mode (see section 4.3), in which the model is never updated. Consider the case where there are two motifs which are similar to each other (that is, they mostly occur in the same upstream sequences) and both are significant. The original REDUCE method would find one of these motifs and add it to the model. This would probably make the second motif unsignificant, and we wouldn't find it. When REDUCE is run using the start guess mode motifs are never added to the model, which means that the program tries to fit every motif to an empty model. This way two motifs that are both significant but very similar can both be found.

Clustering the motifs

When REDUCE is run in start guess mode the output is printed on the usual form, as lists of motifs. These lists are transformed into vectors of F-values, so that

there is one vector for each motif that is significant in at least one sample. The programs compare.pm and remove_zeros.pm take care of these steps.
The next step is to cluster the fixed motifs on their F-values so that similar motifs can be detected. As in section 5.1 this is done with the arrange algorithm [11], and the results are visualized by TreeView [10]. An example of what the results can look like is given in Fig. 5.4. The picture shows that there are several regions with similar motifs, and when looking closer we can examine the sequences of the fixed motifs in such clusters. As Fig. 5.5 shows, the motifs which are clustered together on the F-values often have similar sequences. This is quite interesting, since it is hard to explain such a phenomena unless the sequneces are in fact different instances of binding sites for the same transcription factor. Moreover, it should somehow be possible to combine such sets of sequences into weight matrices.
We have automatically selected some clusters of similar motifs to continue to work with. Selecting significant sets of motifs from the hierachical clustering requires some heuristics. Here, all clusters satisfying the following conditions were extacted:

- The cluster contains at least 10 motifs.
- The cluster has correlation above a certain threshold, in this case 0.4.
- The cluster is not a subset of any other cluster we have chosen.

Note that there are no requirements concerning the sequences of the motifs in a cluster. Using the heuristic above, we obtained a total of 11 clusters.

Constructing weight matrices from significant motifs

In the next step we want to create weight matrices that represent the 11 clusters of similar motifs. Combining the sets of fixed motifs into weight matrices is a non-trivial problem. There are many possible ways to tackle this problem, and the most obvious would be to use a multiple alignment algorithm such as clustalW [9] on all sequences. This approach was tried but didn't work very well. One reason for this is that the sequences are so short, and the overlaps between them even shorter. The multiple alignment method used (clustalW) is not suited to handling such short sequences. Reverse complements also caused problems. Another problem was that some sequences differed a lot from most other sequences in the cluster and perhaps shouldn't be aligned at all. To overcome these problems we developed the program 'wm_align', a method based on weight matrix entropies, described in Appendix A.

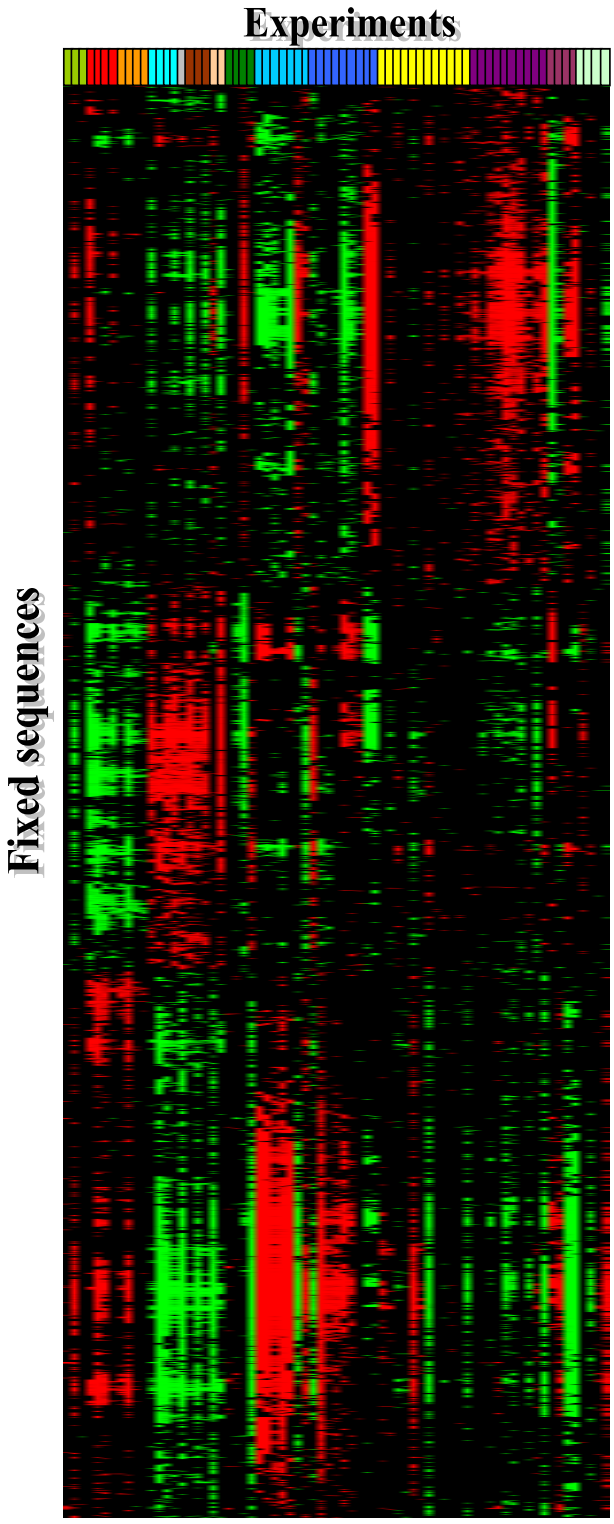

Figure 5.4: The effect of all fixed sequences of length 4 to 7 in all 71 samples from cortex stem cells.

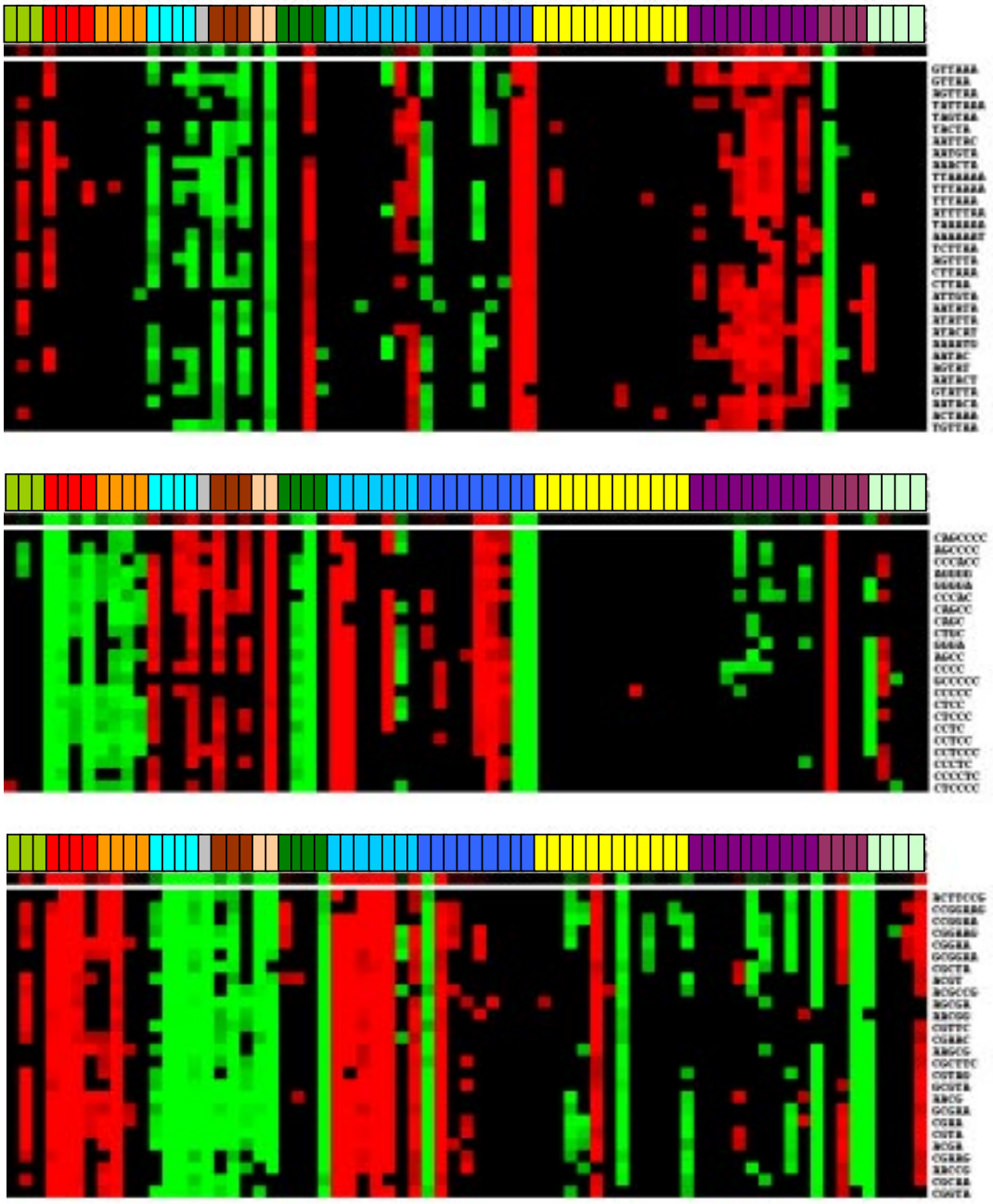

Figure 5.5: Examples of motifs with similar sequences and similar F-values. Each row represents a motif and each column a sample. Red dots represent postitive F-values, green dots negative F-values, and black dots that the motif was not significant in the sample. The motifs have been clustered on their F-values with hierachical clustering, and then rearranged with a optimal node nodering algorithm [11] so that motifs with similar F-values end up next to each other.

Improving the weight matrices using local search

The significance of the matrices obatined this way can be tested using REDUCE. One way of improving the weight matrices further is to use a local search approach, trying to maiximize the total significance of a weight matrix over all samples. Using local search methods, one must define the search space, a neighbourhood relation on the search space, and a value function for evaluating the elements in the search space. For this application, they are defined as follows:

- The search space is the set of all weight matrices (see Appendix A for a definition) of a fixed length, which typically is between 10 and 15 base pairs. Also, all entries in the weight matrix are probabilities, in discrete steps of 10%. That is, all entries are one of the numbers $0, 0.1, 0.2, 0.3, ..., 1.0$.

- The neighbourhood relation defines which weight matrices are considered close to each other in the search space. Two weight matrices A and B are neighbours if they have exactly the same entries on all rows except for one. On the row where they differ, the matrix A has 0.1 higher probability for one base and matrix B has 0.1 higher probability for another base. The probabilities for the two other bases are the same for both matrices. This means that for each matrix all of its neighbours can be obtained by applying the operation of subtracting 0.1 from one matrix postion and adding 0.1 to a position on the same row in every possible way. Thus, for a weight matrix with L rows there are $3 \cdot 4 \cdot L = 12L$ neighbours.

- The value function assigns a score to each weight matrix. It is simply defined as the sum of the $\Delta\chi^2$ (significance) values of the matrix, over all samples. Thus, we want find weight matrices that maximize the value function. Note that we need to run REDUCE on all samples to compute the value of a weight matrix.

With the search space, the nieghbourhood relation and the value function defined, it it possible to apply local search methods to improve the significance of the matrix. This is done using a best improvement heuristic: We use the matrix obtained by aligning the motifs as start guess. Let this matrix be M, which is the current matrix in the search, and evaluate it using the cost function. Next, all matrices in the neighbourhood of M are evaluated. If any of those matrices has a higher value than M, we let M be the matrix with the highest value, and examine its neighbouthood. This procedure continues until no matrix in the neighbourhood has a higher value than M. Then we have reached a local minimum where the search stops, and the current matrix M is the result of the search.
This method doesn't guarantee that the solution we find is optimal, but it is an easy method to improve the significance of a weight matrix, by a series of small

local changes. The programs search.sh, best_motif.sh and neigh.pm are involved in the local search.

5.3 Clustering genes on motifs

The third application is to the significant motifs found by REDUCE to cluster genes. The programs gene_cluster_setup.sh and merge_hit_files do this clustering. The aim of this is to detect groups of genes that are governed by the same transcription factors. This approach will only work if we first find significant motifs that are biologically meaningful. The results of clustering genes on motifs remain to be analysed.

6 Results

We have run the REDUCE method on the 71 samples from cortex stem cells presented in Fig. 5.1. In this section we develop a label permutation test to estimate the significance of our results, at first when running on all TransFac motifs and later when also adding the ones found by unconditional search.

6.1 Label permutation test

A label permutation test was carried out to validate the results of REDUCE. The idea was to run the algorithm both on the actual data and on randomized label sets, and then compare the fit of the model, represented by the $1 - \chi^2$ values. If the actual data produces significantly higher $1 - \chi^2$ values than the randomized data, this shows that the motifs fit the model better than they would do just by chance. To produce randomized data with the same characteristics as the actual data, the same expression levels and promoter regions were used but they were randomly combined: Starting with expression levels $E_1, E_2, ..., E_n$ where n is the number of genes, and associated promoter regions $P_1, P_2, ..., P_n$, the promoter regions were permuted so that each permutation was equally probable.

6.2 Significance of TransFac motifs

The label permutation test was carried out on 10 different randomized data sets. Using only motifs from TransFac, the results shown in Fig. 6.1 were obtained:

Fig. 6.1 shows a number of things: The number of occurences of motifs from TransFac in the promoter regions typically explain a few percent of the variation

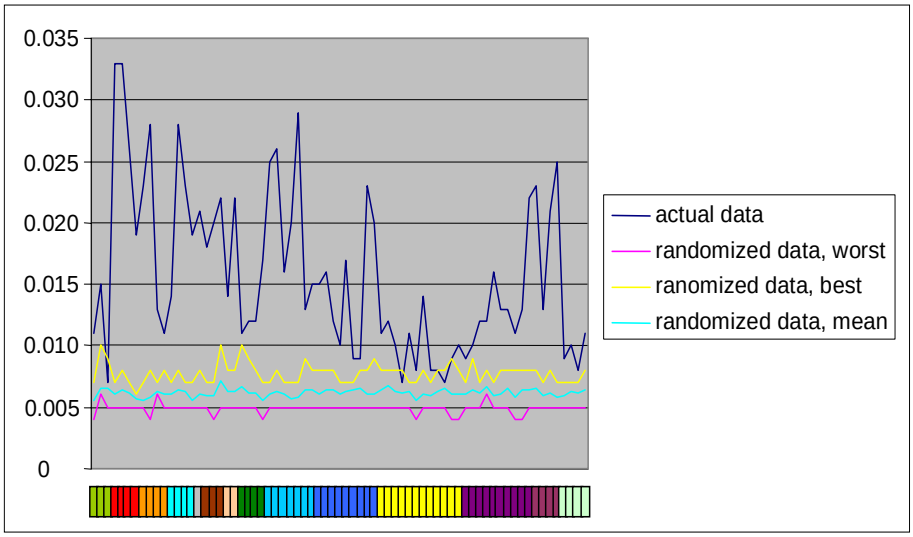

Figure 6.1: The figure shows the results of running REDUCE on the actual data and on 10 sets of randomized data, by plotting the $1 - \chi^2$ values for each sample. The colour coding represents different groups of samples, receiving similar treatments.

in gene expression. This is considerably higher than what is obtained from randomized data, which suggests that the results are significant. For each sample the mean of the $1 - \chi^2$ values for the set of randomized data is around 0.5% and the standard deviation is typically around 0.1%. This means that a $1 - \chi^2$ value of 1.0% for the actual data is 5 standard deviations above the mean of the $1 - \chi^2$ values for the randomized data. For a $1 - \chi^2$ value of 2.5% for the actual data, the corresponding number is 20 standard deviations. The $1 - \chi^2$ varies a lot between samples, and between groups of samples with similar treatments. In some samples the motifs from TransFac can explain a lot of the expression patterns. In other samples the $1 - \chi^2$ values for the actual data is not very much higher than for he randomized data.

6.3 Significance of new motifs

As an alternative to using REDUCE to find motifs from the TransFac data set, the method can be used to perform an unconditional search for new motifs. How this is done is described in more detail in section 5.2. The first step however, is to run REDUCE on all motifs of a certain length. Here, motifs of length 4 to 7 were used. When the REDUCE method was run on these motifs one could observe that motifs with similar sequences often have similar effect (*F*-values).

Fig 5.5 shows a subset of the motifs found by REDUCE and their *F*-values, illustrating this behavour. It is important to note that this behavour is not trivial. Since the fixed sequences reqire perfect matches, two similar sequences (for example AAATTC and AAGTTC) will not match at exactly the same locations in the promoter regions. (Overlapping sequences, such as AAATT and ATT will of course have overlapping matches.) Because of this there is no reason to assume that such sequences will have similar *F*-values, unless they are biologically significant. The many examples where several similar sequences have similar effect on the expressions suggest that such sequences are biologically significant, and that such groups of sequencens are associated with the same transcription factor.
Therefore 24 weight matrices (WMs) were constructed from the 2554 significant motifs found by REDUCE, using the algorithms described in section 5.2. To observe whether the new WMs could improve the fit of the models, the $1-\chi^2$ values for all samples were plotted, both when REDUCE is run on all motifs from TransFac, and when the 24 new WMs are used (in addition to the TransFac motifs). Fig 6.2 summarizes these results.
Fig. 6.2 shows that the fit of the model improved significantly for some samples,

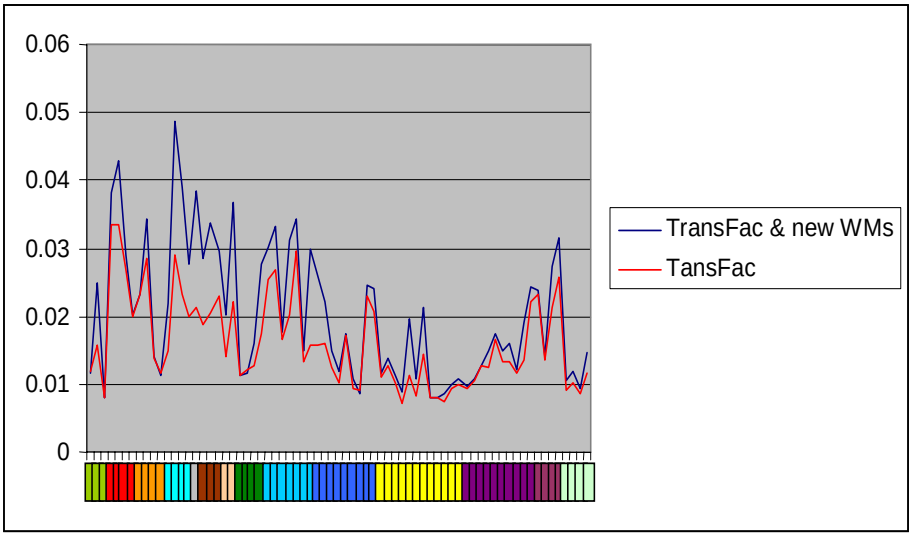

Figure 6.2: A plot of $1-\chi^2$ values for all samples, both when using only TransFac motifs, and when using the 24 new WMs as well as the motifs from TransFac. Again, the samples are colour coded so that groups of samples with similar treatments get the same colour.

whereas the result didn´t change very much in others. A possible explanation for this is that some samples contain expression patterns that correspond well to some motifs, but these motifs are not in TransFac. Other samples may have no

expression patterns that correspond with any motifs at all. In the latter case an unconditional search for motifs would not improve the results.

7 Possible extensions of the GENKOMB project

A number of methods have been implemented in the GENKOMB project. The most central method is REDUCE which finds binding sites for transcription factors (TFs) by correlating sequence data with expressions data. Below are some possible applications of the REDUCE method as well as some other extensions of the GENKOMB project.

Restricting sequence data to conserved regions

One possible extension would be to improve the quality of the sequence data. In the GENKOMB project, upstream sequences of genes are scanned for possible TF binding sites. Thus, the results we obtain depend heavily on the quality of these upstream sequences. Improving the quality of upstream sequences would give better results. This could be accomplished by a filtering mechanism, so that only regions that are conserved between species (eg. human and mouse) are scanned. There are several ways to do this, ranging from straightforward BLAST runs to more sophisticated methods such as the SMASH algortihm [4] or the CONSITE method [13].

Improving the REDUCE model

Our current model is linear, and variations of expression levels are explained only by the number of transcription factor binding sites in the promoter regions. This model could be improved by trying non-linear models, where prior knowledge about TF binding sites is taken into account. For example, the position of binding sites relative to transcription start could be included in the model. Since it is possible to evaluate models (with a label permutation test), it is easy see if prior knowledge improves the results.

Examining combinatorial regulation

Another possible way to extend the GENKOMB project would be to look at combinations of transcriptions factors, instead of just looking at each TF in isolation. This would enable us to come up with a better description of the combinatorial aspects of gene regulation at the transcription level. There are several ways this

could be accomplished: One is to use tools for clustering, statistics and visualization to find sets of motifs that often co-occur in the upstream sequences of the genes. Such sets of motifs could be interpreted as common modules of TFs. Another approach would be to use the REDUCE method on such modules, instead of just running it on single motifs as in the current setup. This would make it possbile to calculate how much of the variations in expression can be explained by motifs and modules of motifs.

Back tracking gene regulatory networks

In the current setup we can only detect TF binding sites in certain samples. From this information it is straightforward to find genes that are regulated by those TFs. This means that we can obtain information on the form "gene A regulates gene B". One possible way to apply the methods and software developed in the GENKOMB project is to use them to investigate the regulatory networks further. The idea would the be to start with some genes (lets say on level 1) and look for the genes that control them (on level 2). This could be done just by looking at the sequence data to find common binding sites for TFs, or by using the expression data as well (as in the REDUCE method). From there it would be possible to backtrack the regulatory network further, to find the genes controlling the genes on level 2, and so on. Besides the sequence data and the expression data this approach also requires a mapping from TF binding sites to genes (more specific to Affymetrics probe IDs). Such data is publicaly available, for example from TransFac [5] and Affymetrix website [12] but rather incomplete.

Matching transcription factors to DNA sequences

Transcription factors are represented as weight matrices, where each element is the probability of having a certain base at a position. We match weight matrices to promoter regions to detect the presence of transcription factor binding sites. It is not obvious how to match weight matrices to DNA sequences, and we have developed a new method based on entropies for this. The continuation of this is to compare the performance of our method to other weight matrix matching approaches, and to improve our program.

Predicting better TF binding sites

TF binding sites in public databases are often based on quite few observations, and therefore they are not so reliable. A local search approach could be a way of making predictions of what the TF sites really should look like. Starting with a known weight matrix, and by making small changes to it in a series of steps

as long as it is improving, we end up with a weight matrix that is better than the original one. The new matrix can be seen as an improvement of the original one.

Investigating biological meaning of results

Another direction that the GENKOMB project could be extended in, is to examine the biolocial meaning of the results. For example, if several genes are controlled by the same TF it is interesting to see if they have similar functionality. This requires a lot of knowledge about the systems we study, but some basic investigations can be carried out with relative small efforts. One such approach is to cluster genes on the TFs that control them, and then match these clusters against Gene Ontololgy [21] data to obtain rough information about the functionality of the genes. This could give information of the type "TF X is in involved function Y", where Y could be a specific cell signaling path or some sort of metabolism for example.

8 APPENDIX A - Weight matrix matching

This section is a description of our approach for matching weight matrices to nucleotide sequences. Given a weight matrix and a sequence, the problem is to determine whether the sequnece is likely to be sampled from the matrix or not. The main benefit of this approach is that it, unlike many other methods, doesn't rely on arbitrarily selected threshold values. The theory behind the weight matrix matching is described in the first subsection.
Two applications of the weight matrix matching have been implemented. The first is a program that can be used for detecting transcription factor binding sites (represented as weight matrices) in the upstream regions of genes (nucleotide sequences). The second is a program that aligns multiple short nucleotide sequences. The two applications are described in the two last subsections.

Theory

A weight matrix W is a matrix:

$$W = \begin{bmatrix} N_{1A} & N_{1C} & N_{1G} & N_{1T} \\ N_{2A} & N_{2C} & N_{2G} & N_{2T} \\ \vdots & \vdots & \vdots & \vdots \\ N_{lA} & N_{lC} & N_{lG} & N_{lT} \end{bmatrix}$$

where l is the length of W, and $W[i, \alpha] = N_{i\alpha}$ is the number of occurences of base α at position i. Let W_i be the total number of bases at a given position,

$$W_i = \sum_{\alpha \in A,C,G,T} N_{i\alpha}$$

Let ω_i^α be the frequency of base α at position i,

$$\omega_i^\alpha = \frac{N_{i\alpha}}{W_i}$$

These notations are used in the definition of the entropy of a position in a weight matrix.

Definition 8.1 *The entropy of position i in a weight matrix W is defined as*

$$E_i(W) = \frac{1}{1-\beta} \log \left(\sum_{\alpha \in \{A,C,G,T\}} (\omega_i^\alpha)^\beta \right)$$

The entropy is a measure of the information content at a certain position. Positions with low entropies have a lot of variation of the frequences ω_i^α (and thus a high information content), whereas positions with high entropies have more similar frequences. The parameter β is a non-negative number. (Techically we use the Renyi [14] or Tsallis [15] entropies. Standard statistical mechanics or Shannon entropy is the limit of $\beta \to 1$).
Other important definitions are about adding nucleotide sequences to weight matrices.

Definition 8.2 *A nucleotide sequence S of length l is a sequence $S = \{s_i\}_{i=1}^{l}$ where $s_i \in \{A, C, G, T\}$, $1 \leq i \leq l$*

Definition 8.3 *A nucleotide sequence S can be added to a weight matrix W of the same length l to obtain a new weight matrix W_S, where*

$$W_S[i,\alpha] = \begin{cases} W[i,\alpha] + 1 & \textit{if } s_i = \alpha \\ W[i,\alpha] & \textit{otherwise} \end{cases}$$

The idea behind our matching is that a sequence that matches a weight matrix must decrease the entropy of the positions when beeing added to it. More formally, we define a score that distinguishes between matching and non-matching sequences.

Definition 8.4 *Let W be a weight matrix.*

$$\textit{A sequence } S \textit{ matches } W \iff Score(W,S) \geq 0 \textit{ , where}$$

$$Score(W,S) = \sum_{i=1}^{l} W_i \left(E_i(W) - E_i(W_s)\right)$$

In the following two subsections, we show how the definition above is used in practice in the programs 'wm_match' and 'wm_align'.

'wm_match' - Matching weight matrices to sequences

The most obvoius application of the weight matrix matching is to use the score in definition 8.4 to build an algorithm for matching weight matrices to nucleotide sequences. Our implementation, 'wm_match', is described below.

The algorithm

The parameter β in the definition of entropy (definition 8.1) is set to a small value ($\beta = 0.0001$). Small values of gives the kind of behaviour that we want when computing the score: the penalty of a mismatch at a position is greater than the benefit of a match. Our algorithm searches for small sections of the nucleotide sequence that match a given weight matrix. This is done by computing the score of matching the weight matrix to every possible subsequence (of same length as the weight matrix). Every time when the score is non-negative, a new match is reported.

Executing the program

The code is written in C and consists of the two modules wm_match and weight_matrix. The program is executed by the command:

```
>./wm_match [Weight_matrix_file] [Sequence_file] [Revcomp]
```

- **Weight_matrix_file** contains a weight matrix on the form:

  ```
  N[1,A] N[1,C] N[1,G] N[1,T]
  N[2,A] N[2,C] N[2,G] N[2,T]
    ...    ...    ...    ...
  N[l,A] N[l,C] N[l,G] N[l,T]
  ```

- **Sequence_file** is a file with nucleotide sequences on fasta format.

- **Revcomp** is an optional flag. If it is set to '-revcomp', we report matches on both the given sequence and its reverse complement strand. Otherwise, only the given sequence is scanned.

- **Output** from the program is written to stdout, on the format:

  ```
  >Sequence_ID1 Hit1
  >Sequence_ID2 Hit2
  ...
  >Sequence_IDN HitN
  ```

  The Sequence_IDs are the IDs stored in the fasta headers. The Hits are the number of matches for the sequences in the fasta file to the weight matrix.

We have optimized the code to make the program run fast and tested the performance of our program against fuzznuc from the EMBOSS package [19], a publicly available program for aligning short nucleotide sequences to longer sequences. It is only possible to compare the performance of the two programs when matching sequences that are totally unambigous, since the fuzznuc program doesn't handle weight matrices. The two programs gave the same results when matching short sequences to a file with long nucleotide sequences, but our program was considerably faster.

'wm_align' - Multiple alignment of DNA sequences

We have also constructed a multiple alignment algorithm based on the weight matrix matching. The algorithm is suitable for short nucleotide sequences that shall be aligned without gaps. Like the program 'wm_match', 'wm_align' uses the score in definition 8.4.

The algorithm

Input is a set of short DNA sequences that we want to align in the best possible way. The result of the alignment is a weight matrix. For example, the sequences ATCGGTT, CGGTT, CCGGT and ACCTGT can be aligned to:

```
ATCGGTT
  CGGTT
 CCGGT
ACCTGT
```

In weight matrix form, this looks like:

```
A T C G
2 0 0 0
0 2 0 1
0 4 0 0
0 0 3 1
0 0 4 0
0 0 0 4
0 0 0 2
```

The idea behind our alignment algorithm is to add one sequence at a time to a weight matrix. The weight matrix is longer than the sequences, so there are several possible ways of adding them. Finding the optimal way of adding the sequences is a non-trivial problem. Another problem is that the result depends on the order in

which the sequences are added to the weight matrix, and it is not clear what order to choose. We deal with both these problems with heuristics based on the score from definition 8.4. In the pseudo-code algorithm below, the following functions are used:

- **empty()** Returns an emtpy weight matrix
- **add(s,W)** Adds the sequence **s** to a weight matrix **W** in the best possible way, i.e the way that gives the best score. The resulting weight matrix is returned. If the score is negative for all possible ways of adding the sequence, then no sequence is added.
- **bestScore(s,W)** Returns the score of adding the sequence **s** to the weight matrix **W** in the best possible way.
- **entropy(W)** Returns the entropy of a weight matrix, which is the sum of entropies over all positions.

```
S = {all sequences}
for all s in S
 We <- empty()
 Wstart <- add(s,We)
 S' <- S-{s}
 for all s' in S'
  Compute bestScore(s',Wstart)
 O <- all sequences in S' ordered according bestScore
 Ws = Wstart
 for all o in O
  Ws <- add(o,Ws)
 Compute entropy(Ws)
return the weight matrix with the lowest entropy
```

It is possible to make an implementation of the algorithm above so that the reverse complement of sequences may also be aligned. When computing the best score of a match between a sequence an a weight matrix, we also consider all possibilities of matching the reverse complement of the sequence. Then, either the sequence or its reverse complement is added to the weight matrix (or possibly none of them).

Executing the program

The code is written in C, and basically consists of the two modules wm_align and weight_matrix. The program is executed by the command:

```
>./wm_align [Sequence_file] [Revcomp]
```

- **Sequence_file** is a file with short nucleotide sequences (at most 50 bp)

  ```
  DNA_sequence1
  DNA_sequence2
  ...
  DNA_sequenceN
  ```

  The DNA sequences consist of the letters A,C,G,T.

- **Revcomp** is an optional flag. If it is set to '-revcomp', the reverse complements of the DNA sequences are also considered.

- **Output** from the program is a weight matrix that is written to stdout:

  ```
  N[1,A] N[1,C] N[1,G] N[1,T]
  N[2,A] N[2,C] N[2,G] N[2,T]
   ...    ...    ...    ...
  N[l,A] N[l,C] N[l,G] N[l,T]
  ```

  $N[i, \alpha]$ is the number of sequences in the resulting multiple alignment with base α at position i.

9 APPENDIX B - Transcription start qualities

Since we don't know very much about how the transcription start data is obained we cannot be sure of its quality. To make a rough estimate of the quality the transcription starts, these were compared to the RIKEN database with mouse cDNA [17]. Since the transcription start data used in this project originates from the ensembl database [16], all transcription starts from ensembl for both human and mouse were compared to the RIKEN database. The comparison between human ensembl data and RIKEN mouse cDNA can give information about the quality of our transcription starts, but is a bit misleading since there are considerable differences between species as well. The comparison between mouse data from ensembl and RIKEN cDNA on the other hand is more accurate, but doesn't tell us anything about the data we use. Even so, it can provide a rough estimate of the overall quality of the data in ensembl.
To compare the transcription starts from ensembl to the cDNA in RIKEN, we extracted sequeces from the genome, based on the ensembl transcription starts. We used entire genes (as annotated by ensembl) as well as 10000 bp upsteam. Since only matches close the the actual transcription starts are interesting, we only used the first 200 bp of the RIKEN cDNA. Moreover, the RIKEN cDNA were masked for repeats with RepeatMasker [2]. The RIKEN data contains 60770 cDNA fragments, all of which were used. The comparisons were carried out by aligning each sequence from ensembl with each sequence from RIKEN using the software BLAST [3].
The blast runs of ensembl mouse data against RIKEN cDNA were done with the following requirements:

- E-value < 1e-75
- length > 180
- identity > 0.98

Thus the conditions for a match are rather strict. Moreover, if a cDNA fragment had several matchings, only the one with the lowest E-value was selected. The results from the blast runs are summarized below: Out of all 22444 mouse genes from ensembl 13141 matched some cDNA, which is 59%. Out of those 13141 genes, 3812 genes had a cDNA fragment matching within 200 bp from the ensembl transcription start, which is 29%. Fig. 9.1 shows the distribution of matching cDNA fragments throughout the ensembl genes and their upstream sequences.

The blast runs of human genes from ensebl against RIKEN cDNA had somewhat different settings:

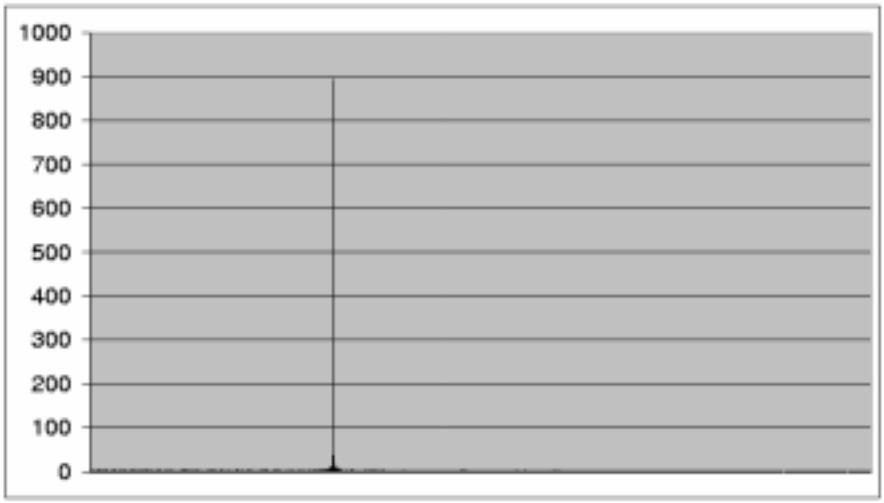

Figure 9.1: Results of blasting cDNA fragments from RIKEN against mouse genes from ensembl. The positions in the genomic sequence relative the transcription starts are on the X-axis and the number of cDNA fragments matching in each postion are on the Y-axis. The peak corresponds to the transcription starts fron the ensembl annotation.

- E-value < 1e-10
- length > 100
- identity > 0.60

These requirements are not as strict as in the mouse-mouse alignments above, since there is a difference between the species. Also, we want to be able to estimate where in the human ensembl genes the RIKEN transcription starts match. This is only possible if the match starts close to the beginning of the RIKEN fragment. If a match would begin (for example) hundreds of bases into the cDNA fragment, it would be hard to tell where the transcription start would match the human ensebl sequence. It is only possible to extrapolate backwards if the matching starts very close to the start of the cDNA fragment. Therefore there is an additional requirement, that the start of the match is no longer than 20 bp away from the start of the RIKEN cDNA.
The following results were obtained when human genes from ensembl were blasted against mouse cDNA from RIKEN: Out of 22980 human genes from ensembl, matchings for 2827 genes satisfied the requirements above, which is 12%. Of these 2827 genes, 981 had matchings within 200 bp from the ensembl transcrip-

tion start, which is 35%. Fig. 9.2 show the distribution of matching cDNA fragments throughout the ensembl genes and their upstream sequences.

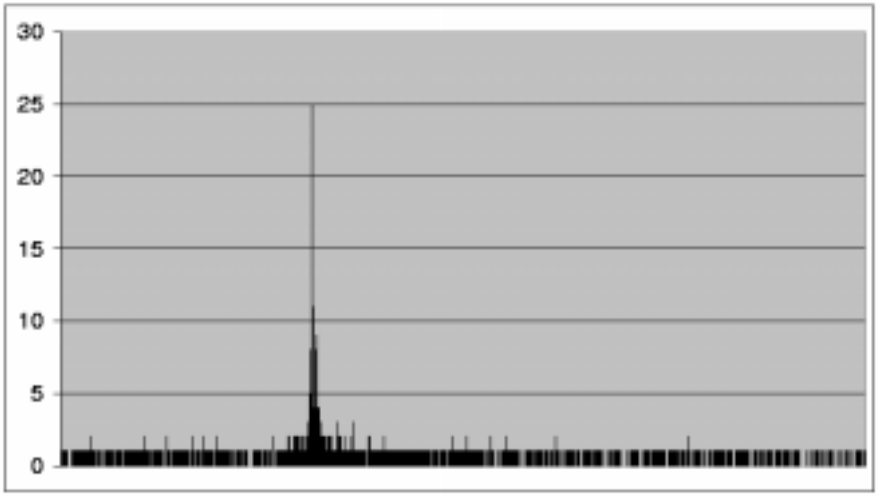

Figure 9.2: Results of blasting cDNA fragments from RIKEN against human genes from ensembl. As in the previous figure, the positions in the genomic sequence relative the transcription starts are on the X-axis and the number of cDNA fragments matching in each postion are on the Y-axis. Again, the peak corresponds to the transcription starts from the ensembl annotation.

These are just a couple of quick tests and no definitive conclusion about the quality of the transcription start data can be reached from them. But the tests at least suggest that the transcription starts are not completely wrong. To come up with a reliable quantitative measure of the quality of the data would require more systematic tests. This would involve using more prior knowledge about the sequence data, as well as fine tuning the paramenters in blast.

References

[1] H. J. Bussemaker, H. Li, E. D. Siggia. Regulatory element detection using correlation with expression. *Nature Genetics vol 27, 167, 2001*

[2] A. F. A. Smit, P. Green. RepeatMasker *www.repeatmasker.genome.washington.edu, 2002*

[3] W. Gish, D.J States. Identification of protein coding regions by database similarity search. *Nature Genetics 3:266-272, 1993*

[4] M. Zavolan, N. Rajewsky, N. D. Socci, T. Gaasterland. SMASHing regulatory sites in DNA by human-mouse sequence comparisons. *Submitted to Genome Research, September 2002.*

[5] E. Wingender, X. Chen, R. Hehl, H. Karas, I. Liebich, V. Matys, T. Meinhardt, M. Pruss, I. Reuter and F. Schacherer. TRANSFAC: an integrated system for gene expression regulation. *Nucleic Acids Res. 28, 316-319, 2000*

[6] M. Blatt, S. Wiseman, E. Domany. Superparamagnetic clustering of data. *Phys. Rev. Lett. 76, 3251-3255, 1996*

[7] G. Getz, E. Levine, E. Domany. Coupled two-way clustering analysis of gene microarray data. *PNAS, 2000*

[8] J.E. Stajich, D. Block, K. Boulez, S.E. Brenner, S.A. Chervitz, C. Dagdigian, G. Fuellen, J.G.R. Gilbert, I. Korf, H. Lapp, H. Lehvaslaiho, C. Matsalla, C.J. Mungall, B.I. Osborne, M.R. Pocock, P. Schattner, M. Senger, L.D. Stein, E.D. Stupka, M. Wilkinson, E. Birney. The Bioperl Toolkit: Perl modules for the life sciences. *Genome Research. 2002 Oct;12(10):1161-8.*

[9] J.D. Thompson, D.G. Higgins, T.J. Gibson. CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting,position-specific gap penalties and weight matrix choice. *Nucleic Acids Res. 22:4673-4680. (1994)*

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

[11] Z. Bar-Joseph, D.K. Gifford, T.S. Jaakkola. Fast optimal leaf ordering for hierarchical clustering. *9th International Conference on Intelligent*

Systems for Molecular Biology, Tivoli Gardens, Copenhagen, Denmark, July, 2001.

[12] Affymetrix website *www.affymetrix.com*

[13] B. Lenhard, A. Sandelin, L. Mendoza, P. Engström, N. Jareborg. & Wasserman, W. W. Identification of Conserved Regulatory Elements by Comparative Genome Analysis *J. Biol, in press (2003).*

[14] A. Renyi. Probability Theory, *Noth-Holland, Amsterdam, 1970*

[15] C. Tsallis. Possible generalization of Boltzmann-Gibbs statistics. *J. Stat. Physics. vol. 52, 479 1988*

[16] T. Hubbard, D. Barker, E. Birney, G. Cameron, Y. Chen, L. Clark, T. Cox, J. Cuff, V. Curwen, T. Down, R. Durbin, E. Eyras, J. Gilbert, M. Hammond, L. Huminiecki, A. Kasprzyk, H. Lehvaslaiho, P. Lijnzaad, C. Melsopp, E. Mongin, R. Pettett, M. Pocock, S. Potter, A. Rust, E. Schmidt, S. Searle, G. Slater, J. Smith, W. Spooner, A. Stabenau, J. Stalker, E. Stupka, A. Ureta-Vidal, I. Vastrik, and M. Clamp, The Ensembl genome database project, *Nucleic Acidns Res. 2002 30: 38-41. 2002. www.ensembl.org*

[17] The FANTOM Consortium and the RIKEN Genome Exploration Research Group Phase I & II Team. Analysis of the mouse transcriptome based on functional annotation of 60,770 full-length cDNAs. *Nature 420:563-573, 2002*

[18] Stanford University, Center For Molecular and Genetic Medicine. Bioinformatics Manual. *http://cmgm.stanford.edu/help/manual/ (2003)*

[19] Rice, P. Longden, I. and Bleasby, A. EMBOSS: The European Molecular Biology Open Software Suite. *Trends in Genetics June 2000, vol 16, No 6. pp.276-277*

[20] IUPAC-IUB SYMBOLS FOR NUCLEOTIDE NOMENCLATURE: Cornish-Bowden *Nucl. Acids Res. 13: 3021-3030. 1985*

[21] The Gene Ontology Consortium. Gene Ontology: tool for the unification of biology. *Nature Genetics 25: 25-29. 2000*