Title
Verbumculus and the discovery of unusual words

Permalink
https://escholarship.org/uc/item/5m66k36w

Journal
Journal of Computer Science and Technology, 19(1)

ISSN
1000-9000

Authors
Apostolico, A
Gong, F C
Lonardi, S

Publication Date
2004

Peer reviewed

Verbumculus and the Discovery of Unusual Words

Alberto Apostolico[*] *U. of Padova & Purdue*

Fangcheng Gong[†] *Celera Genomics*

Stefano Lonardi[‡] *U. of California, Riverside*

Abstract

Measures relating word frequencies and expectations have been constantly of interest in Bioinformatics studies. With sequence data becoming massively available, exhaustive enumeration of such measures have become conceivable, and yet pose significant computational burden even when limited to words of bounded maximum length. In addition, the display of the huge tables possibly resulting from these counts poses practical problems of visualization and inference.

VERBUMCULUS is a suite of software tools for the efficient and fast detection of over- or under-represented words in nucleotide sequences. The inner core of VERBUMCULUS rests on subtly interwoven properties of statistics, pattern matching and combinatorics on words, that enable one to limit drastically and *a priori* the set of over- or under-represented candidate words of all lengths in a given sequence, thereby rendering it more feasible both to detect and visualize such words in a fast and practically useful way. This paper is devoted to the description of the facility at the outset and to report experimental results, ranging from simulations on synthetic data to the discovery of regulatory elements on the upstream regions of a set of genes of the yeast.

The software VERBUMCULUS is accessible at `http://www.cs.ucr.edu/~stelo/Verbumculus/` or `http://wwwdbl.dei.unipd.it/Verbumculus/`

Keywords: verbumculus, unusual words, subword statistics, pattern discovery, regulatory elements, suffix trees

[*] Dipartimento di Ingegneria dell' Informazione, Università di Padova, Padova, Italy, and Department of Computer Sciences, Purdue University, Computer Sciences Building, West Lafayette, IN 47907, USA. Work supported in part by NSF Grant CCR-9700276, by Purdue Research Foundation Grant 690-1398-3145, by the Italian Ministry of University and Research, and by the Research Program of the University of Padova. E-MAIL: `axa@dei.unipd.it`

[†] Celera Genomics, 45 W. Gude Drive, Rockville, MD 20850. E-MAIL: `Fangcheng.Gong@celera.com`

[‡] Department of Computer Science and Engineering, University of California, Riverside, CA 92521, USA. Work supported by Purdue Research Foundation Grant 690-1398-3145, by the Italian Ministry of University and Research, by the Research Program of the University of Padova and by Bourns College of Engineering, University of California, Riverside. E-MAIL: `stelo@cs.ucr.edu`

1 Introduction

Over one decade after National Research Council mapped out the plan for the Human Genome Project, the project has been developed into a full fledged research field – Genomics, which transforms biology researches from cottage endeavor into enterprise operation. Genomic researches are mainly organized around two areas: high throughput DNA sequencing and genome-wise detection of gene expression. Sequence data in both public and private databases have been accumulating at exponential rate under continuous improvement of sequencing technology and steady increase of funding [24, 16]. Whole genome sequences have been constantly churning out from the genomic centers around the world since the first whole genome sequence was published [19]. Over fifty whole genome sequences are currently available for prokaryotic and eukaryotic organisms.

High density array technologies such as DNA microarray and gene chip not only provide a systematic and global snapshot of genome expression in relation to developmental stages, anatomical structures, and/or external cues, but also offer a powerful means to cluster genes based on their temporal, spatial and amplitude patterns of expression [44, 32, 17, 15]. Together with sequence data, expression data would enable the assignment of functional information to genes of otherwise unknown functions. The conceptual assumption of the approach is that genes that exhibit similar expression patterns contribute to the same biological process or functions. Therefore such genes share more or less common regulatory domains in the upstream regions for the coordinated control of gene expression.

A large number of nucleotide motifs show distinct distribution patterns within the genomes of various organisms, and can be distinguished from each other. Moreover, during the evolutionary process, living organisms have accumulated certain biases towards or against some specific motifs in their genomes, which are used as regulatory elements. Highly recurring nucleotide words are often observed to correspond to regulatory regions or protein binding sites of genes. Vice versa, rare nucleotide motifs may be discriminated against due to structural constraints of genomes or specific reservations for global transcription controls, such as in early cascade of embryo development.

Whole genome sequences together with genome-wise expression data offer a global view of genomic structure and functions of a living system. However, it is a great challenge to assign functions to DNA sequences. For example, out of circa 30-40 thousands genes in human genome, only about 10,000 are associated with known functions. Thus, global and systematic search for sequence patterns in genomes is a necessary step to link structural sequences to defined functions. This paper describes our software development (VERBUMCULUS) in a genome-wise searching system for over- or under-represented nucleotide motifs, and our initial effort to attribute biological functions to those motifs. The main advantages brought about by VERBUMCULUS are in terms of speed, flexibility and visualization efficiency. This rests on the core structure of the program, which takes advantage of strong properties at the intersection of statistics, pattern matching and combinatorics on words [5, 4]. The facility at the outset can conduct global pattern discovery in linear time and space.

In general, the task of detecting, enumerating and testing nucleotide word frequencies in large genomes, which are typical cases for eukaryotic organisms, requires significant computational resources even when limited to the words up to some maximum length. It often becomes infeasible to detect and visualize such words in a fast and practically useful way. Among the tools available for these purposes, we list WORDUP [39], YEAST-TOOLS [49], and R'MES [43, 42]. Conceived primarily as an aid in intercepting conserved segments in related sequences in a family, WORDUP is based on a first order Markov model. It detects statistically significant sequence motifs of 6-10 nucleotides in a family of sequences by comparing the expected number of sequences containing a given pattern with the number of observed sequences that contain that pattern. The facilities under YEAST-TOOLS comprise tables of frequencies for nucleotide words of up to 10 bases as observed in all coding and non-coding regions of the yeast genome. The related analyses of frequencies and expectations invest the entire target genome, with the objective of identifying relatively simple statistically relevant patterns represented by short motifs with a highly conserved core. R'MES is a general-purpose set of programs to detect words that appear in a given DNA sequence with unexpected frequency. Two classes of models are used to model the sequence: stationary Markov chains and 3-periodic stationary Markov chains. Under either probabilistic model, the number of occurrences of a word in a sequence is considered to be statistically interesting if it differs significantly from an estimator of its expected value. Estimators of the expected counts are obtained using a Gaussian or (for long words) a compound Poisson approximation. In either case, R'MES provides a score indicating whether the word is under- or over-represented.

The search for unexpectedly frequent or rare substrings is only one component of the broader quest for interesting patterns of more general kinds. Along these lines, patterns and families thereof have been variously characterized, and criteria, algorithms and software developed in correspondence. Without pretending to be exhaustive, we mention SPEXS [10, 11], MEME [8], PRATT [28, 27], YEBIS [52], SPLASH [14], TEIRESIAS [41], CONSENSUS [25], GIBBS SAMPLER [30, 36], WINNOWER [40, 29], PROJECTION [13], WEEDER [38] MITRA [18], among others. The performance of these tools has been increasing dramatically over the years, though the hardest challenges (see, e.g., [40]) are still terrain for contests.

This paper is organized as follows. In the next three sections, we describe the basic structure and features of the VERBUMCULUS facility, leaving out most algorithmic and combinatorial details, for which we refer to [5, 4]. Following that, we show the results of its application to the test analysis of regulatory motifs of genes.

2 Statistical analysis of sequences

In the following, we use w to denote a generic nucleotide word, and $w_{[i,j]}$ to indicate the substring of w that begins at position i and ends at position j. Two main notions of frequency are considered of interest in our context. The first notion is given in terms of the number of occurrences of w within a single given sequence. The second is concerned instead with the number of sequences containing at least one occurrence of w out of the total number of sequences in a family. The expressions and computations of the expected values, moments and related scores of significance depend substantially on the particular notion at hand. We discuss first those based on a single sequence, where we identified five common scores used in the scientific literature (shown on the left of Table 1).

Here, $f(w)$ is the number of observed occurrences of w in the input sequence, $E(w)$ is the number of expected occurrences of w under a Bernoulli (i.i.d.) model, $Var(w)$ is the variance on the number of occurrences of w under the same model, and $\hat{Var}(w)$ is an easier to compute, first-order approximation of the true variance, which matches in fact the simplifying assumption of uncorrelated symbol occurrences. Specifically, it is seen [5, 4] that for an input sequence of length n and a pattern w of length $m \leq (n+1)/2$ we have:

$$E(w) = (n - m + 1)p(w)$$

and

$$Var(w) \ = \ E(w)(1 - p(w)) - p(w)^2(2n - 3m + 2)(m - 1)$$

$$\boxed{\begin{aligned} z_1(w) &= f(w) - E(w) \\ z_2(w) &= \frac{f(w)}{E(w)} \\ z_3(w) &= \frac{f(w) - E(w)}{\sqrt{\hat{Var}(w)}} \\ z_4(w) &= \frac{f(w) - E(w)}{\sqrt{Var(w)}} \\ z_5(w) &= \frac{f(w) - \mathcal{E}(w)}{\max\{\sqrt{\mathcal{E}(w)}, 1\}} \end{aligned}} \qquad \boxed{\begin{aligned} z_7(w) &= c(w) - E_c(w) \\ z_8(w) &= \frac{c(w)}{E_c(w)} \\ z_9(w) &= \frac{(c(w) - E_c(w))^2}{E_c(w)} \end{aligned}}$$

Figure 1: Scores based on the number of occurrences (left) and scores based on the number of sequences containing at least one occurrence (right)

$$+ \ 2p(w) \sum_{l=1}^{s} (n - m + 1 - d_l) \prod_{j=m-d_l+1}^{m} p(w_{[j]})$$

where $p(w)$ is the probability of occurrence of w and $\{d_1, d_2, \ldots, d_s\}$ are the lengths of the periods of w. A string w has a *period* z if w is a prefix of z^k for some integer k. Alternatively, a string z is a period of a string w if $w = z^l v$ and v is a possibly empty prefix of z. We refer to [6, 5, 4] for details and discussion. Truncating $Var(w)$ after the first term yields $\hat{Var}(w) = E(w)(1 - p(w))$. Score z_5 is after Brendel *et al.* (see [12] for details): $\mathcal{E}(w)$ is the expected frequency of w based on the observed frequency of $(m-1)$-mers and $(m-2)$-mers, i.e.,

$$\mathcal{E}(w) = \frac{f(w_{[1,m-1]}) f(w_{[2,m]})}{f(w_{[2,m-1]})}.$$

We now turn to the scores associated with frequencies defined on a set of strings $\{x_1, x_2, \ldots, x_k\}$, also called a *multisequence*. For multisequences, the three additional scores shown on the right of Table 1 have been selected. Here $E_c(w)$ and $c(w)$ are, respectively, the expected and observed number of sequences that contain at least one occurrence of w. Given k sequences of respective sizes n_i for $i \in [1, k]$, Pesole *et al.* [39] define $E_c(w)$ as follows

$$E_c(w) = \sum_{i=1}^{k} \left(1 - e^{-\mathcal{E}_i(w)}\right)$$

where $\mathcal{E}_i(w)$ is the expected number of occurrences of w in the i-th sequence. An estimator of the true expectation is calculated after Stuckle *et al.* [48] by assuming a first order stationary Markov chain

$$\mathcal{E}_i(w) = \frac{f_i(w_{[1,2]}) f_i(w_{[2,3]}) \ldots f_i(w_{[m-1,m]})}{f_i(w_{[2]}) f_i(w_{[3]}) \ldots f_i(w_{[m-1]})}$$

where $f_i(w)$ is the observed number of occurrences of w in the i-th sequence.

In a typical, on-line application, parameters such as the probabilities of the individual symbols are estimated from the corresponding frequencies in the input sequence. Alternatively, our algorithm allows the submission of a separate model sequence from which probabilities are estimated. Likewise, the analysis of the target sequence may proceed considering the sequence as a whole as well as by performing computations independently within a number of consecutive segments in a suitable cover of the input, and analyzing one such "window" at a time.

3 Methods

For a given choice of a probabilistic setting and score(s) one would like ideally to compute exhaustive frequency tables reporting values for all substrings of a sequence, or perhaps at least for the statistically most "surprising" among them. Even setting aside for a moment the effort involved in the computation of the necessary parameters and scores, the sheer number of entries associated with such an exhaustive tabular presentation would quickly become unfit for human inspection, hence hardly useful at the outset. To see this, assume to be given *a priori* a source model, one of the above z-scores, and some arbitrarily fixed threshold value t for that score, whereby a word will be considered surprising and thus included in the output table if its score exceeds t. Now an observed sequence x of n bases might in principle exceed t with all of its substrings. Since x may contain about $n^2/2$ distinct substrings, then a sequence of a modest n =1,000 bases might force us to output about half a million subwords as being surprisingly frequent. While such an extreme case seems entirely unrealistic, it does help illustrating a point of great practical relevance, namely, that the volumes of data produced in this and other tasks of motif discovery risk to rapidly saturate the perceptual bandwidth of the final user [3].

One of the main assets of VERBUMCULUS comes in form of a powerful property that limits drastically the number of surprising subwords that one needs to consider. The property holds under reasonable assumptions for scores that fit the general format $z(w) = (f(w) - E(w))/N(w)$, where $N(w)$ is a nonnegative normalizing factor for the difference such as, e.g., the standard deviation for the count. For scores of this kind, it is possible [5, 4] to confine the computation to only a number of candidate surprising words linear in the length of the host sequence. Moreover, the set $\mathcal{W}$ of these candidates can be identified *a priori*, and their relationship to any other, e.g., over-represented word not in $\mathcal{W}$ is as follows (under-represented words obey a symmetric property). For any word w not in $\mathcal{W}$ such that $z(w) > t$, there is a word w' in $\mathcal{W}$ such that:

1. $w' = wv$ for some nonempty word v, i.e., the "neglected" word is embedded in a word of $\mathcal{W}$ as a prefix;

2. $z(w') > z(w)$, i.e., w' is at least as surprising as w.

Such a drastic limitation on the order of the number of candidates, as well as their identification, weighing and display are all inextricably interwoven reflections of a same combinatorial property, which has to do with the score being monotone within certain families of patterns. This property requires that if w and an extension $w' = wv$ of w are nonempty substrings of the text x such that $f(w) = f(wv)$, then the score of w does not exceed that of w'. Under these conditions, w can be neglected as the surprise it conveys is subsumed by w'.

The tables below display a collection of monotonicity results established about the models and z-scores considered. We refer to [4] for the corresponding proofs and discussion. For convenience of notation, we set $\rho(w) \equiv E(w)/N(w)$, where $N(w)$ appears in the score as the expected value of some function of w. The interpretation of the tables is straightforward. For example, Property 1.1 states a simple fact on the monotonicity of $E(w)$ given the monotonicity of $\rho(w)$ and $N(w)$. Under some general conditions on $N(w)$ and $\rho(w)$ we can prove the monotonicity of any score functions of the form described above.

Some of the properties are not straightforward. For example, Property 1.2 says that these scores are monotonically decreasing when

$$f < E^* = E(w)\frac{\gamma N(w) + N(wv)}{N(w) + N(wv)}$$

	Property	Conditions
(1.1)	$\frac{f(wv) - E(wv)}{N(wv)} > \frac{f(w) - E(w)}{N(w)}$	$N(wv) < N(w), \quad \rho(wv) \leq \rho(w)$
(1.2)	$\left\lvert \frac{f(wv) - E(wv)}{N(wv)} \right\rvert > \left\lvert \frac{f(w) - E(w)}{N(w)} \right\rvert$	$N(wv) < N(w), \quad \rho(wv) \leq \rho(w)$ and $f(w) > E(w)\frac{\gamma N(w) + N(wv)}{N(w) + N(wv)}$
(1.3)	$f(wv) - E(wv) > f(w) - E(w)$	$E(wv) < E(w)$
(1.4)	$\frac{f(wv)}{E(wv)} > \frac{f(w)}{E(w)}$	$E(wv) < E(w)$
(1.5)	$\frac{f(wv) - E(wv)}{E(wv)} > \frac{f(w) - E(w)}{E(w)}$	$E(wv) < E(w)$
(1.6)	$\frac{f(wv) - E(wv)}{\sqrt{E(wv)}} > \frac{f(w) - E(w)}{\sqrt{E(w)}}$	$E(wv) < E(w)$
(1.7)	$\left\lvert \frac{f(wv) - E(wv)}{\sqrt{E(wv)}} \right\rvert > \left\lvert \frac{f(w) - E(w)}{\sqrt{E(w)}} \right\rvert$	$E(w) > E(wv), \quad f(w) > E(w)\sqrt{\gamma}$
(1.8)	$\frac{(f(wv) - E(wv))^2}{E(wv)} > \frac{(f(w) - E(w))^2}{E(w)}$	$E(w) > E(wv), \quad f(w) > E(w)\sqrt{\gamma}$

Table 1: General monotonicities for scores associated with the counts f, under the hypothesis $f(w) = f(wv)$. We have set $\rho(w) \equiv E(w)/N(w)$ and $\gamma \equiv E(wv)/E(w)$.

	Property	Conditions
(2.1)	$E(wv) < E(w)$	none
(2.2)	$f(wv) - E(wv) > f(w) - E(w)$	$f(w) = f(wv)$
(2.3)	$\frac{f(wv)}{E(wv)} > \frac{f(w)}{E(w)}$	$f(w) = f(wv)$
(2.4)	$\frac{f(wv) - E(v)}{E(wv)} > \frac{f(w) - E(w)}{E(w)}$	$f(w) = f(wv)$
(2.5)	$\frac{f(wv) - E(wv)}{\sqrt{E(wv)}} > \frac{f(w) - E(w)}{\sqrt{E(w)}}$	$f(w) = f(wv)$
(2.6)	$\left\lvert \frac{f(wv) - E(wv)}{\sqrt{E(wv)}} \right\rvert > \left\lvert \frac{f(w) - E(w)}{\sqrt{E(w)}} \right\rvert$	$f(w) = f(wv), \quad f(w) > E(w)\sqrt{\gamma}$
(2.7)	$\frac{(f(wv) - E(wv))^2}{E(wv)} > \frac{(f(w) - E(w))^2}{E(w)}$	$f(w) = f(wv), \quad f(w) > E(w)\sqrt{\gamma}$
(2.8)	$\frac{f(wv) - E(wv)}{\sqrt{E(wv)(1 - p(w)q(v))}} > \frac{f(w) - E(w)}{\sqrt{E(w)(1 - p(w))}}$	$f(w) = f(wv), \quad p(w) < 1/2$
(2.9)	$Var(wv) < Var(w)$	$p_{max} < 1/\sqrt[m]{4m}$
(2.10)	$\frac{E(wv)}{\sqrt{Var(wv)}} < \frac{E(w)}{\sqrt{Var(w)}}$	$p_{max} < \sqrt{2} - 1$
(2.11)	$\frac{f(wv) - E(wv)}{\sqrt{Var(wv)}} > \frac{f(w) - E(w)}{\sqrt{Var(w)}}$	$f(w) = f(wv), \quad p_{max} < \min\{1/\sqrt[m]{4m}, \sqrt{2} - 1\}$
(2.12)	$\left\lvert \frac{f(wv) - E(wv)}{\sqrt{Var(wv)}} \right\rvert > \left\lvert \frac{f(w) - E(w)}{\sqrt{Var(w)}} \right\rvert$	$f(w) = f(wv), \quad p_{max} < \min\{1/\sqrt[m]{4m}, \sqrt{2} - 1\}$ and $f(w) > E(w)\frac{\gamma\sqrt{Var(w)} + \sqrt{Var(wv)}}{\sqrt{Var(w)} + \sqrt{Var(wv)}}$

Table 2: Monotonicities for scores associated with the number of occurrences f under the Bernoulli model. We set $\gamma \equiv E(wv)/E(w)$.

and monotonically increasing when $f > E^*$. We can picture the dynamics of the score as follow. Initially, we can assume $E^* > f$, in which case the score is decreasing. As we extend the word, keeping the count f constant, E^* decreases (recall that E^* is always in the interval $[E(wv), E(w)]$). At some point, $E^* = f$, in which case the score stays constant. By extending the word even more, E^* becomes smaller than f, and the score starts to grow. Some consequences of Property 1.2 are captured by Properties 1.7 and 1.8. Property 1.2 also holds by exchanging the condition $\rho(wv) \leq \rho(w)$ with $f(w) > E(w) > E(wv)$.

Turning now to Table 2, we summarize monotonicity results for the Bernoulli, or i.i.d., model. In this case, each symbol is generated from the same probability distribution, and independently from its context. A comprehensive study of other models and scores can be found in [4].

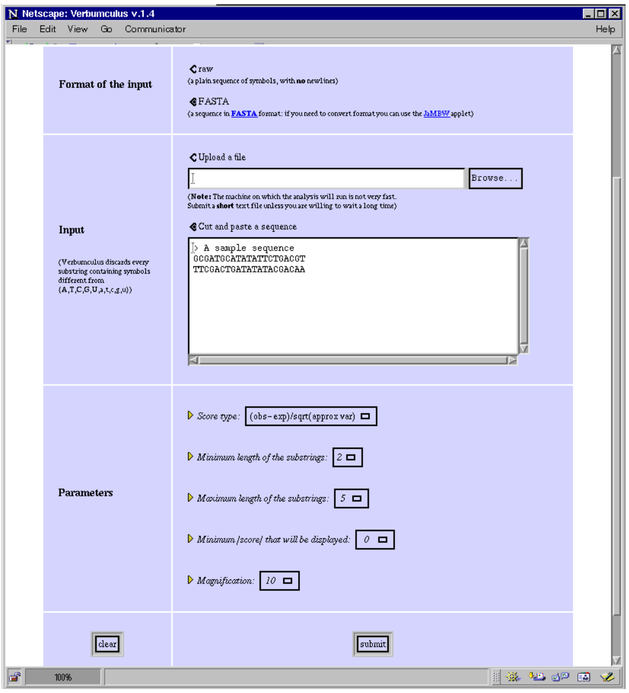
As already observed, a tabular representation of surprising words in a sequence is bound to become rapidly bulky with increasing sequence length, even if the number of candidates is linear in that length. In our approach, the computation, storage and display of the statistical parameters of interest are all organized around the structure of a special compact trie represented by a suitably pruned version of a *suffix tree*, the trie of all suffixes of a given sequence (see e.g., [34, 2, 23]). By the trie being implemented in compact form it is meant that all nodes in it are branching nodes, whence arcs are labeled by substrings of the input sequence rather than by individual symbols. In a full-fledged tree, the leaves are in one-to-one correspondence with the suffixes of the input. Since every subword of the input is a prefix of some such suffix(es), then any subword of the input will be spelled out on a unique path leading from the root to some leaf and ending at a node or perhaps in the middle of an arc. For a sequence of length n, the tree will have a number of leaves and hence also of internal nodes bounded by n, so that always less than $2n$ subwords of the input can end precisely at a node in any pruned version of the trie. The property exploited by VERBUMCULUS is that, within the ample domains of monotonicity of the scores considered, the set $\mathcal{W}$ coincides precisely with the set of these subwords [5, 4]. In other words, for a monotone score z

> *the largest positive values of z and hence most over-represented words will occur at the internal nodes of the trie rather than in the middle of an arc. Symmetrically, the most under-represented words occur only as unit symbol extensions of those nodes.*

Combined with the advantages of trie visualization over table listing, this remarkable property opens the way to compactly displaying all of the unusual subwords of a sequence at once. Moreover, the computations involved can be speeded up significantly. The tree itself is built in time linear in the input by a number of well known methods. Once the tree is built and perhaps pruned to some preliminary maximum length, subword occurrences and other similar counts can be similarly obtained in linear time. For instance, the number of leaves in the subtree rooted at some node represents the number of observed occurrences of the word spelled out on a path from the root to that node or anywhere on the immediately preceding arc. VERBUMCULUS annotates the tree with one or more of the above scores, depending on the type of analysis one wants to perform. The typical process of annotation also takes linear time, which in cases like z_4 is achieved through resort to rather complex algorithmics, due to the structure of *Var*. For scores that require multiple tries to be built and superimposed to one another, like in the computation of $c(w)$ for z_7, z_8 and z_9, the linear time algorithm by Hui [26] is used.

4 Software description and usage

VERBUMCULUS is composed by three modules: the tree builder VERBUM, the graph drawing program DOT, and the graphic interface TREEVIZ. The entire package consists of about ten thousand



Format of the input

○ raw
(a plain sequence of symbols, with **no** newlines)

● FASTA
(a sequence in FASTA format: if you need to convert format you can use the JaMBW applet)

Input

(Verbumculus discards every substring containing symbols different from {A,T,C,G,U,a,t,c,g,u})

○ Upload a file

(**Note:** The machine on which the analysis will run is not very fast.
Submit a **short** text file unless you are willing to wait a long time)

● Cut and paste a sequence

```
> A sample sequence
GCGATGCATATATTCTGACGT
TTCGACTGATATATATACGACAA
```

Parameters

- *Score type:* (obs−exp)/sqrt(approx var)
- *Minimum length of the substrings:* 2
- *Maximum length of the substrings:* 5
- *Minimum |score| that will be displayed:* 0
- *Magnification:* 10

Figure 2: VERBUMCULUS' web interface

lines of code.

VERBUM is written in C++ using the Standard Template Library which should allow great portability under different platforms. The code has been compiled, without any change, under Solaris and Linux. VERBUM reads the input sequence(s) and the various parameters supplied by the user, and creates a (possibly pruned) suffix trie annotated with the score selected at the beginning by the user. The output is a text file representing the tree in the dot format (see below). VERBUM is particularly fast: although the time taken for the analysis depends heavily on the particular score and other input parameters, it is typically in the order of few seconds for the most common choices.

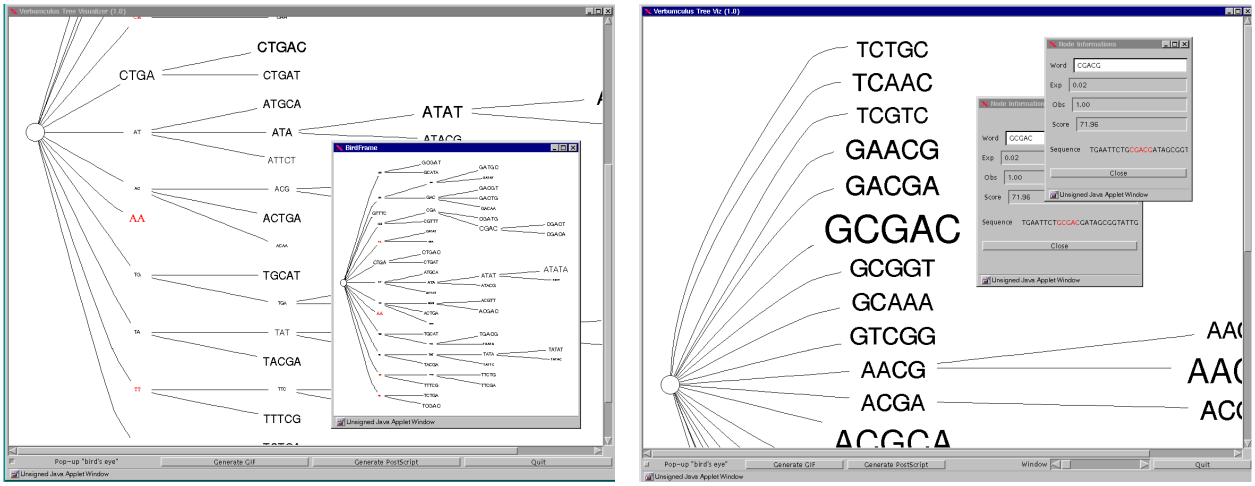
DOT is the graph drawing program by AT&T Labs, part of the GRAPHVIZ package [20]. It reads graphs in the dot representation and outputs drawings in a dozen of formats, among which Postscript and GIF. The source code and binary executables for common platforms are available from their site, and licensing is almost open source.

TREEVIZ is the graphical user interface that runs on the client side, and more specifically on the browser of the user. It is entirely written in Java, and uses the GRAPPA libraries by AT&T Labs.

A couple of thousand lines of Perl glue everything together. Perl scripts generate the HTML for the input forms and control the execution of the various stages, handling exceptions and errors.

The user of VERBUMCULUS is presented with the form shown in Fig. 2. He has the option of submitting the input either as a raw sequence of letters or in FASTA format. The input can be "pasted" into the window or uploaded to the server. For analyzing long sequences, we strongly advise to download the executable VERBUM and DOT and work locally, in order to avoid the overhead of network communications and the relative inefficiencies of Perl scripts and Java.

Various parameters can be adjusted. The most important choice is the type of score to be used in tree annotation. Additionally, a wide range of different filters is available to limit the size of the



Word GCGAC
Exp 0.02
Obs 1.00
Score 71.96
Sequence TGAATTCTGCGACGATAGCGGTATTG

Word CGACG
Exp 0.02
Obs 1.00
Score 71.96
Sequence TGAATTCTGCGACGATAGCGGT

Figure 3: Example outputs of TreeViz

tree. The user can set the minimum and maximum length of the nucleotide words, a lower bound on the absolute value of the score, a lower bound on the value of the expectation (to avoid "rare" words) and a forbidden substring (to avoid, for example, words contains `TATA`).

For better performance, we have limited the visualization of TreeViz to 100 nodes: if the tree is bigger, Verbumculus will send a Postscript file with the drawing of the tree. If the user wants to take advantage of the interactive facilities of TreeViz, he will have to increase the effectiveness of the filters in order to produce a smaller tree.

The magnitude of a score value is transduced through font size, in the sense that for every word w, the higher the absolute value of the score of w, the bigger the font used to represent w. Words with a negative score are, in addition, printed in red italics (see Fig. 3, on the left).

Once TreeViz has drawn the tree, the user can navigate it looking for conspicuous nucleotide motifs. At any time, clicking on an nucleotide word will visualize information about its number of occurrences, the expected value of this number, and the corresponding value of the z-score. Along with these values, a graphical representation of the positions of the occurrences of the nucleotide word in the original sequence is produced and displayed (see Fig. 3, on the right).

Since the tree can be fairly big, TreeViz offers the option to get an overall picture of the tree by clicking on the "bird's eye" button, which produces the small window inset of Fig. 3. Finally, TreeViz can generate a drawing of the tree in Postscript or GIF that can be saved on the machine of the user for further and more accurate scrutiny.

5 Simulations and Dithering

Before showing the results of using Verbumculus on real biological data, we report on some tests performed on artificial sequences. In our present context, this is meant primarily to show the effectiveness of the tool in the pattern discovery process. In practice, this or a similar procedure may be followed fruitfully as a preliminary treatment, for the purpose of fine tuning the sensitivity of the tool and adapt it to the particular sequence or family under study.

An example dithering procedure could be as follows. First, we generate and process several pseudo-random strings assuming a symmetric Bernoulli model. For every random sequence produced, we generate and annotate the corresponding tree. As expected, we find that unless the random sequence is very short the tree does not display any surprising word.

Next, we inject into the random sequences a controlled number of non-overlapping repetitions of words. In our example, we use the two words `GATTA` and `AAAAA`, in separate experiments. Since the process of overwriting the original random letters with occurrences of a given word changes the probability distribution, we have to make some adjustments in the probability distribution. Let p_a denote the probability of symbol $a \in \Sigma$ in the original sequence and w some word of length $|w| = m$, with a proportion of a's given by q_a. Forcing h substrings in our sequence to coincide with w will change the probability accounting for the "free" occurrences of a outside the h copies of w into

$$\bar{p}_a = \frac{p_a n - h\, m\, q_a}{n - h\, m}.$$

As the left part of Figure 4 displays, five occurrences of `GATTA` in a text of size 1,000 can be enough, with our settings, in order for the program to output that word as the highest scoring pattern. If the size of the text is increased to 10,000, then typically twenty occurrences of the word turn out to be enough to produce the same visual effect (see right half of Figure 4).

For a broader analysis, we run 1,000 trials for all choices of $h = 0, 1, \ldots, 15$, $n =$1,000 counting the number of times that `GATTA` is the highest scoring pattern in the entire tree. The graph on the left of Figure 5 shows the relative proficiency of the scores z_2, z_3 and z_4 in separating the "signal" `GATTA`, from "noise". Specifically, the plots show the fraction of the 1,000 trials in which the word `GATTA` was the highest scoring word in the tree, for increasing number h of injections. From the graph we can observe that scores z_3 and z_4 have identical performance (as one would expect, since `GATTA` has no periods) and they seem to be better than z_2. On the right of Figure 5 we instead plotted three pairs of curves respectively for scores z_2, z_3, z_4 (it so happens that in this particular case the pairs for z_3 and z_4 are on top of each other) with increasing number h of injections. For each pair, the upper curve represents the average score for the word that achieves the largest score in the tree, while the lower curve represents the average score for the word `GATTA`. Some observations are in order. First, the score of `GATTA` grows linearly with h. Second, the average of the highest z_2-score is bigger than z_3 and z_4. Third, at some point h^* the lower curve touches the upper curve and the pattern is "discovered". Note that the lower curve touches the upper curve sooner for z_3 and z_4 than z_2.

If the pattern is periodic, like e.g. `AAAAA`, then we need fewer copies, i.e., a smaller value of h in order to obtain a comparable visual impact. Figure 6 displays the results using four forced occurrences in a sequence of size 1,000 and ten in a sequence of size 10,000. We run the same simulation on 1,000 trials as before, this time injecting $h = 0, 1, \ldots, 15$ occurrences of `AAAAA` (see left half of Figure 7). We were expecting the score z_4 to have an advantage over the other because of the high periodicity of the word. Surprisingly, the figure shows that the score that detects sooner the presence of `AAAAA` is z_3. In the right half of Figure 7 we collected as before the average scores for the words achieving the largest score in the tree (upper curve), and the average score for the word `AAAAA` (lower curve). This time, the tree families of curves corresponding to z_2, z_3 and z_4 are clearly distinguishable. The function that returns the biggest scores is again z_2, followed by z_3 and then z_4. For all three, the score of `AAAAA` grows linearly with h. Note, however that z_2 and z_3 have different slopes than z_4.

6 Tests and Experiments

We report here some results on experiments running VERBUMCULUS on the *upstream regions* of some genes of the yeast. The upstream region of a gene is the untraslated region that precedes the

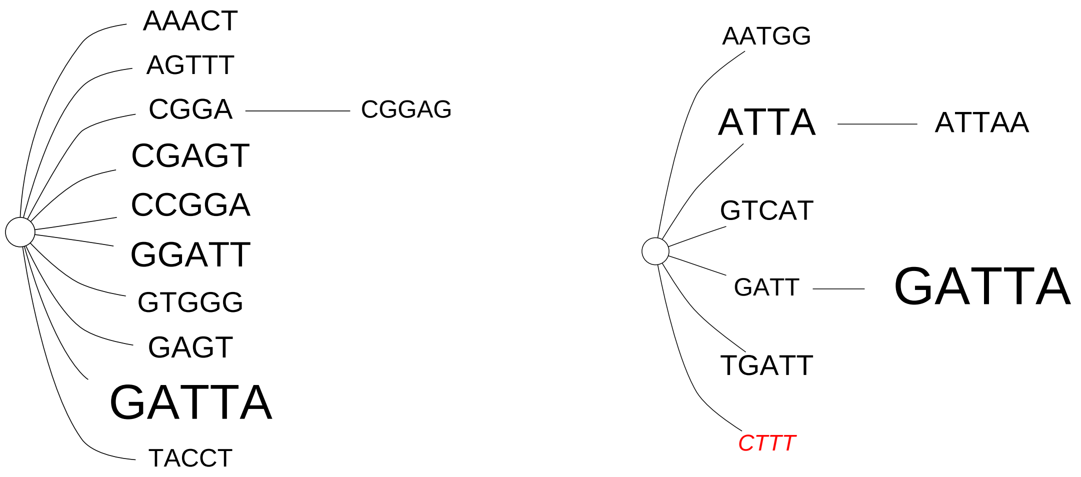


Figure 4: LEFT: trie from a random string of size 1,000 with 5 forced occurrences of the word `GATTA`. RIGHT: trie from another random string of size 10,000 with 20 forced occurrences of `GATTA`. Both tries are annotated using z_3, with threshold 3.0

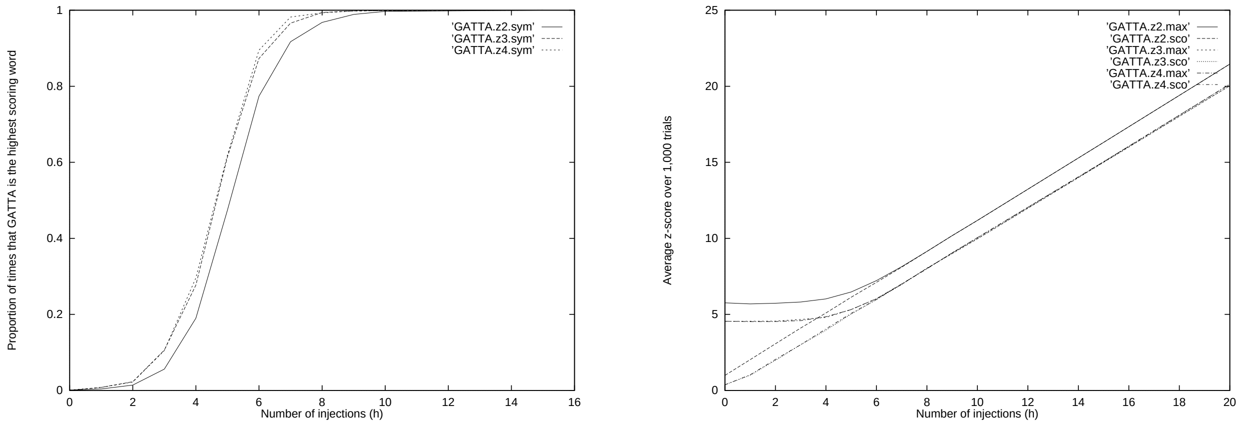


Figure 5: LEFT: The fraction of 1,000 trials in which the word `GATTA` is the *highest* scoring word in the tree for z-scores z_2, z_3, z_4, versus number of injections h. RIGHT: Curve pairs for scores z_2, z_3, z_4 versus h. The upper curve in each pair represents the average score for the word that achieves the maximum score, the lower curve represents the average score of the word `GATTA`. The curves for z_3 and z_4 are hardly distinguishable due to substantial overlap.

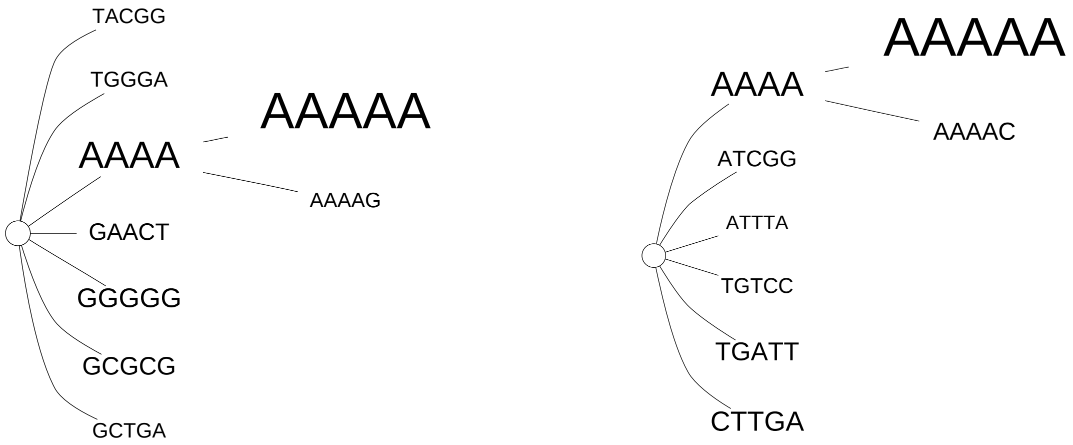


Figure 6: LEFT: trie from a random string of size 1,000 with 4 forced occurrences of the word **AAAAA**. RIGHT: trie from another random string of size 10,000 with 10 forced occurrences of **AAAAA**. Both trees are annotated using z_3, with threshold 3.0

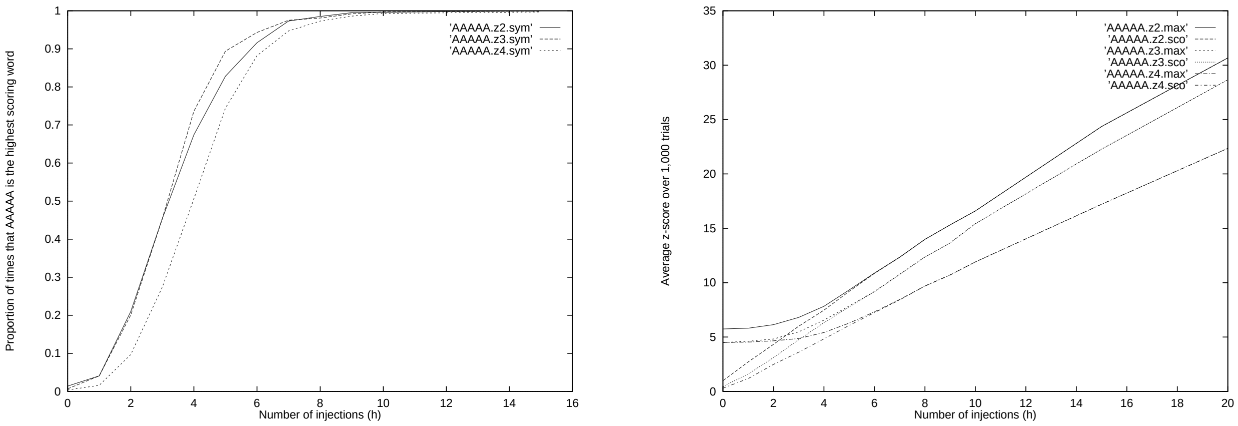


Figure 7: LEFT: fraction of times versus h that **AAAAA** is the *highest* scoring word in the tree, for z-scores, z_2, z_3, z_4. RIGHT: curve pairs for the word **AAAAA** and the highest scoring words under scores z_2, z_3, z_4.

start codon ATG of size 500-1000 base pairs, when reading the sequence in the standard orientation 5' to 3'. It is usually known to contain several control signals that regulates the production of the mRNA, called *promoters* or *regulatory sites*. Finding such motifs is usually the first step in understanding how genes interact with the environment and with each other.

The first dataset we analyze is related to ten families of genes isolated by van Helden *et al.* [49]. Each family contains a set of co-regulated genes, that is, genes that have similar expression under the same external conditions. The hypothesis is that in each family the upstream region will contain some common motif. Moreover, one can also expect that such signals are going to be over-represented across the family. In this first experiment we use the same parameters and score type on all the multisequences to test the general performance of our tool.

The second dataset comes from the work on the *sporulation* of the budding yeast conducted by Chu *et al.* [15]. Seven families of co-regulated genes have been characterized using DNA microarray technology. Again, one of the purposes of the investigation is to find unusual words in the upstream regions of these genes. Here we concentrate on a couple of families and we show the sensitivity of our tool to different choices of parameters and score functions. Both experiments also expose the limitations of our approach.

6.1 Regulatory sites in yeast

The metabolism of the yeast has been widely studied and provides several examples of known regulatory sites. In many cases, the transcriptional factor involved in the common response is known, as well as its binding site. van Helden *et al.* selected ten families of genes based on prior biological knowledge on their activity. For each gene in a family its 800 bps upstream sequence was extracted. The set of all upstream sequences belonging to the same family constitutes the multisequence on which we performed the analysis using VERBUMCULUS.

The parameters of the analysis are as follows. We use scores based on the number of occurrences (z_3 and z_4), a threshold between 3 and 10 depending of the maximum size of pattern, the latter being between 5 and 8 symbols. We adjusted the threshold to obtain a tree of about twenty nodes. We also filtered out words containing one or more of the subwords TATA, AAAA and TTTT, when these words were predominant.

Tables 3, 4 and 5 summarize the result of our tests. For each multisequence we report the identifier, the number k of sequences, the motif previously known and characterized by experiments, the motifs found by van Helden *et al.*, and the trees produced with VERBUMCULUS. For the sake of clarity, we manually circled the words that match the biologically significant motif.

VERBUMCULUS is capable of discovering the biologically significant patterns in families NIT, MET, PHO, PDR, GAL, GCN and TUP, although sometimes partially. Moreover, these motifs can be found among the highest scoring words. Also note that other patterns which are also scoring high are usually in a suffix-prefix relation with the highest, suggesting that their occurrences are correlated.

However, in the multisequences INO, HAP and YAP VERBUMCULUS assigns low scores to the motifs and therefore they do not show up in the final tree. In two of these three cases, though, the tool by van Helden *et al.* is also not capable of detecting these patterns as shown in the Tables. Additionally, the tool by van Helden *et al.* does not give any satisfactory answer for the GAL family, where instead VERBUMCULUS catches CGGCG and GCCGC which correspond the the beginning and the end of the motif. Finally, note that, in general, VERBUMCULUS has great difficulty to handle motifs contains multi-valued symbols, for example the ones for GAL and TUP families.

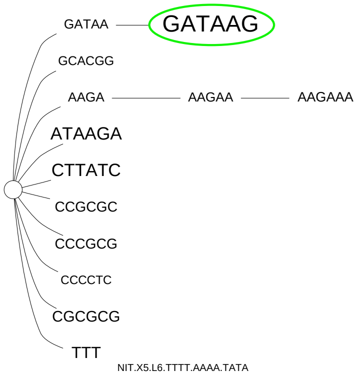
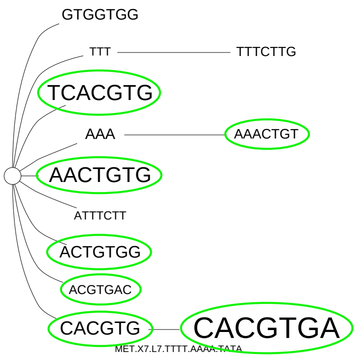
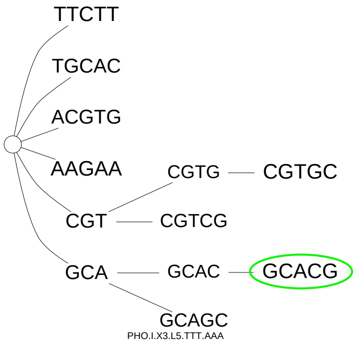
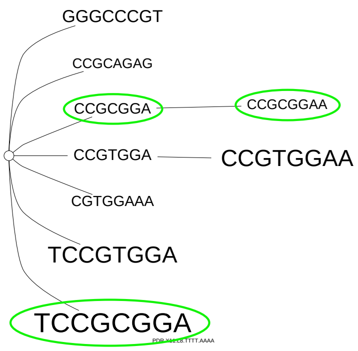
Family	k	Motif	van Helden *et al.*	VERBUMCULUS
NIT	7	GATAAG	CTGATAAGA CCGCGC CGGCAC ACATCT	 GATAA GATAAG GCACGG AAGA AAGAA AAGAAA ATAAGA CTTATC CCGCGC CCCGCG CCCCTC CGCGCG TTT NIT.X5.L6.TTTT.AAAA.TATA
MET	11	TCACGTG AAAACTGTGG	GTCACGTG AACTGTGGC ATATAT TATATA GCTTCC	 GTGGTGG TTT TTTCTTG TCACGTG AAA AAACTGT AACTGTG ATTTCTT ACTGTGG ACGTGAC CACGTG CACGTGA MET.X7.L7.TTTT.AAAA.TATA
PHO	5	GCACGTGGG GCACGTTTT	CGCACGTGGG CACGTTT CTGCAC TGCCAA	 TTCTT TGCAC ACGTG AAGAA CGTG CGTGC CGT CGTCG GCA GCAC GCACG GCAGC PHO.I.X3.L5.TTT.AAA
PDR	7	TCCGCGGA	TCCG{C|T}GGAA GCGCGA AGGCACC	 GGGCCCGT CCGCAGAG CCGCGGA CCGCGGAA CCGTGGA CCGTGGAA CGTGGAAA TCCGTGGA TCCGCGGA PDR.X11.L8.TTTT.AAAA

Table 3: van Helden dataset of co-regulated genes

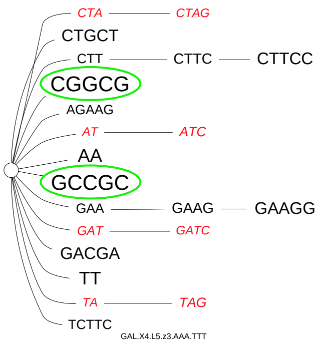
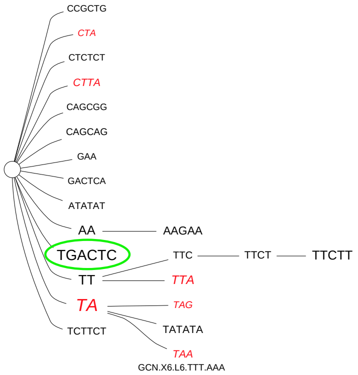
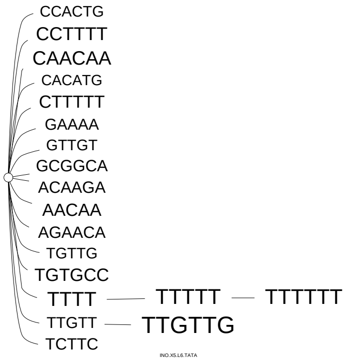
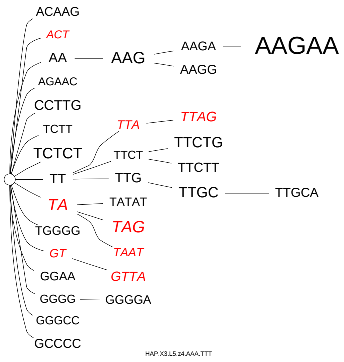
Family	*k*	Motif	van Helden *et al.*	VERBUMCULUS
GAL	6	CGGN^{5}WN^{5}CCG	?	 CTA CTAG CTGCT CTT CTTC CTTCC CGGCG AGAAG AT ATC AA GCCGC GAA GAAG GAAGG GAT GATC GACGA TT TA TAG TCTTC GAL.X4.L5.z3.AAA.TTT
GCN	38	RRTGACTCTTT	A{G|A}TGACTC{A|T} CAGCGG AACCGGC CATCGAA AGAGAG	 CCGCTG CTA CTCTCT CTTA CAGCGG CAGCAG GAA GACTCA ATATAT AA AAGAA TGACTC TTC TTCT TTCTT TT TTA TA TAG TATATA TCTTCT TAA GCN.X6.L6.TTT.AAA
INO	10	CATGTGAAWT	CAACAA{C|G} CATGTGAA TCTTCA GTTCAA GTCGCA	 CCACTG CCTTTT CAACAA CACATG CTTTTT GAAAA GTTGT GCGGCA ACAAGA AACAA AGAACA TGTTG TGTGCC TTTT TTTTT TTTTTT TTGTT TTGTTG TCTTC INO.X5.L6.TATA
HAP	8	CCAA{T|C}	AGAGAGA ATGGGGC	 ACAAG ACT AA AAG AAGA AAGAA AAGG AGAAC CCTTG TTAG TCTT TTA TCTCT TTCT TTCTG TTCTT TT TTG TTGC TTGCA TA TATAT TAG TGGGG TAAT GT GGAA GTTA GGGG GGGGA GGGCC GCCCC HAP.X3.L5.z4.AAA.TTT

Table 4: van Helden dataset of co-regulated genes (continued)

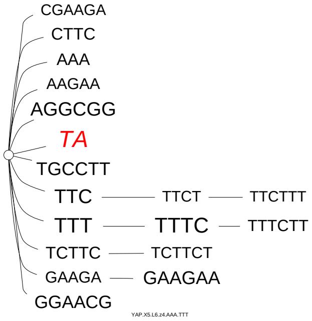
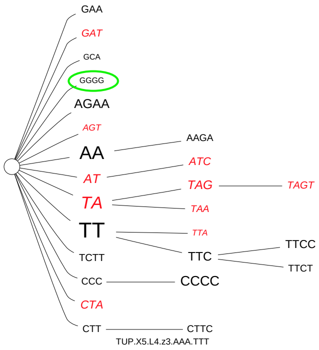
Family	k	Motif	van Helden *et al.*	VERBUMCULUS
YAP	16	TTACTAA	CGTTCCGT CATTAC CTGAAG	 CGAAGA CTTC AAA AAGAA AGGCGG TA TGCCTT TTC TTCT TTCTTT TTT TTTC TTTCTT TCTTC TCTTCT GAAGA GAAGAA GGAACG YAP.X5.L6.z4.AAA.TTT
TUP	25	KANW^{4}ATSYG^{4}W	TT{C|T}{C|G}NG^4{T|C}{A|C} AGGCACGGG AAA{A|G}AA AAGGAGGA ACAAACA CTCCGC {T|C}CTGCA CGTCGC	 GAA GAT GCA GGGG AGAA AGT AAGA AA ATC AT TAG TAGT TA TAA TT TTA TTCC TCTT TTC TTCT CCC CCCC CTA CTT CTTC TUP.X5.L4.z3.AAA.TTT

Table 5: van Helden dataset of co-regulated genes (continued)

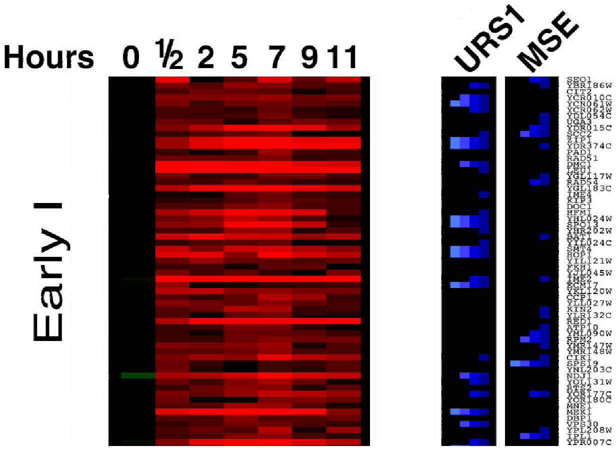


Figure 8: Genes of the Early(I) cluster induced or repressed during sporulation. Adaptated, by kind permission, from a figure at `http://cmgm.stanford.edu/pbrown/sporulation/figures/fig5ss.html`

6.2 Sporulation of the Yeast

We report here some results from testing VERBUMCULUS on the dataset involved in the work on sporulation conducted at Stanford by Chu *et al.* [15]. The authors used DNA micro-array technology to expose the temporal patterns of gene expression of *Saccharomyces Cerevisiae* during meiosis and spore formation. This was done along the lines of a rather standard procedure, as follows. First, changes in the concentration of mRNA transcript from known genes of the budding yeast were measured during seven consecutive intervals. Next, the average expression profiles were used to classify the genes. Figure 8 reproduces an adaptation of the image at the outset, available in full view at `http://cmgm.stanford.edu/pbrown/sporulation/figures/fig5ss.html`. As usual, higher and higher degrees of expression translate into darker and darker shades of red, while lower concentrations yield progressively darker shades of green. Seven clusters were produced in this particular experiment, labeled as `Metabolic`, `Early(I)`, `Early(II)`, `EarlyMiddle`, `Middle`, `MidLate`, and `Late`.

The two bands of columns with blue bars in Figure 8 identify genes of which the promoters contain a putative URS1 or MSE regulatory sequence, respectively. The degree to which the sequence matches the consensus for each of these regulatory elements is indicated by the brightness of the bar: the best matches are represented by the bright blue bars that appear to be concentrated towards the left of each band, the less stringent matches cause the darker blue bars more visible towards the right. The most stringent match for the URS1 site is 5'-`TCGGCGGCTDW`-3', and the least stringent is 5'-`GGCGGC`-3'. The most stringent match for the MSE site is 5'-`HDVKNCACAAAAD`-3', and the least stringent is 5'-`DNCRCAAAWD`-3'.

Figure 5 of [15] shows that the upstream sequences relative to the genes in the clusters `Metabolic`, `Early(I)`, and `Early(II)` contain several occurrences of the regulatory element URS1, while the ones in the clusters `EarlyMiddle` and `Middle` contain many MSE sites. We report here results regarding the analysis of the cluster `Early(I)` with the objective to expose the URS1 site. We also report a less satisfactory analysis of the cluster `Middle`.

6.3 The `Early(I)` cluster

The cluster `Early(I)` contains 36 genes, namely RTS2, MEK1, NDJ1, MNE1, EHD2, DBP1, IPL1, VPS30, UGA3, PCH2, SEO1, CIT2, SCC2, KIP3, RAD51, IME4, ZIP1, DMC1, RAD54, HFM1, LEU1, PAD1, ATP10, CIK1, FKH1, HOP1, SPS19, KIN2, ECM17, RPM2, CCP1, BAT1, IME2, SPO13, RED1, SMT4. We extracted the upstream region of 600 base pairs (allowing overlaps with other ORFs) using the tool developed by VanHelden *et al.* [49].

Table 6 shows the trees produced by VERBUMCULUS on the family of 36 upstream regions and annotated with the scores z_2, z_3, z_4, z_8 and z_9, maximum length 6 bps. Only patterns with a score higher than 4.0 (in absolute value) are shown. For comparison purposes, Table 7 lists a few most notable words along with their statistics. In [15] it is reported that 43% of the upstream regions of the genes in the cluster `Early(I)` have a core URS1 motif, while we found only 33%. However, the expected number of `GGCGGC` is so small that the reported occurrences of this word have to be considered surprising by any measure, whether in terms of total number of occurrences or number of sequences containing it.

Observe that the words in the z_2-tree of Table 6 are not independent. Instead, prefixes of some words are suffixes of others, which suggests that their occurrences might be correlated. We used sequence alignment (see, e.g. [7]) to "assemble" short sequences in longer sequences (see Tables 8-10). As seen in this example, the alignment step can be used to partially overcome the limitation of VERBUMCULUS to discover exact motifs.

Table 8 shows an alignment produced by using four such overlapping words from the tree. The consensus of the alignment `TCGGCGGCA` exactly matches two motifs in in the Transcription Element Search System (TESS/TRANSFAC) database [51, 50], namely Y$HSP70_02 and Y$SSA1_01. This pattern contains the core `GGCGGC`, mentioned repeatedly in [15].

Next, we used four more motifs from the tree to build another alignment (see Table 9). The consensus `TGCGCGGCT` matches one motif in TESS, Y$G3PDH_01, that is already known in the literature [33].

Finally, we chose five motifs from the tree and build the multiple alignment of Table 10. The consensus `TAGCCGCGGA` exactly matches five motifs in TESS, namely Y$CAR1_02, Y$CAR2_01, Y$MES1_01, Y$SPO13_01, Y$TOP1_01. For example, Y$SPO13_01 is known to be a key regulatory of nitrogen repression and meiotic development [46]. However, the authors of [15] did not report the finding of this regulatory element.

In conclusion, VERBUMCULUS not only succeeded in identifying the regulatory elements we were looking for, but also found some other interesting new patterns in the cluster that were possibly overlooked. At the same time, in the z_2-tree of Table 6 there were also patterns such as `CTTTTC`, `AAAAAA`, and `ACCGGC`. The former two have been detected as elements in scaffold/matrix attachment regions (MARs) of eukaryotic genomes [1, 47]. MARs are basic components for high level genome compaction and organization, therefore are highly frequent in genomic sequences [21, 9]. In addition, biological experiments have shown that they anchor genomic sequences to proteinaceous nuclear matrix and affect gene expression [45, 35, 9]. For the latter pattern, we have not found any biological significance yet.

We went on producing a few more suffix trees annotated with other scores. Table 6 shows the tree decorated with scores z_3, z_4, z_8 and z_9. Some remarks are in order. The z_3 and z_4 trees are quite similar except for the following: the tree for z_3 enhances `GCCGCC`, `TTTTT` while the tree for z_4 does not. Vice versa, the tree for z_4 emphasizes `TA` as being under-represented, a phenomenon that is missed in the tree annotated with z_3.

The word `GGCGGC` appears again in both z_3- and z_4-trees. However, in the case of z_3 the pattern

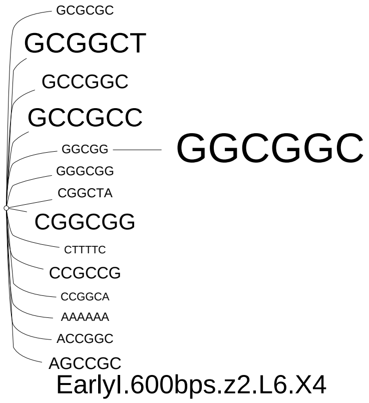
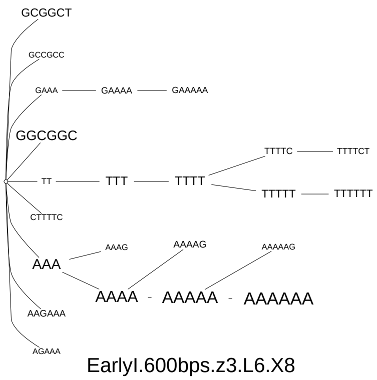
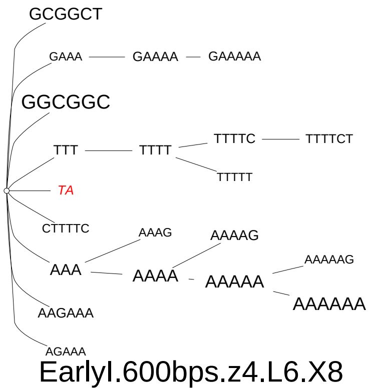
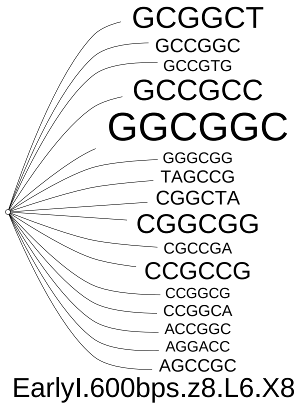
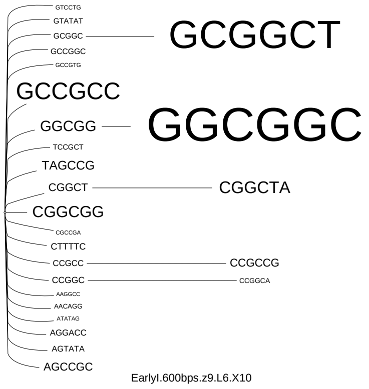
<table>
<tr><th>Score</th><th>T</th><th>Verbumculus</th><th>Score</th><th>T</th><th>Verbumculus</th></tr>
<tr><td>z_2</td><td>4.0</td><td>
</td><td>z_3</td><td>8.0</td><td>
</td></tr>
<tr><td>z_4</td><td>8.0</td><td>
</td><td>z_8</td><td>9.0</td><td>
</td></tr>
<tr><td>z_9</td><td>10.0</td><td>
</td><td></td><td></td><td></td></tr>
</table>

Table 6: `Early(I)` cluster as seen through Verbumculus, T is the threshold

	occurrences		*sequences*	
w	$E(w)$	$f(w)$	$E_c(w)$	$c(w)$
AAAAAA	26.44	109	25.74	22
TTTTTT	29.38	110	27.63	25
GGCGGC	1.15	25	1.51	12
TAGCCG	2.43	9	2.11	9

Table 7: Explicit statistics for some most devious words (in term of number of occurrences and/or number of sequences containing at least one occurrence) in the 36 sequences that form cluster `Early(I)`. The individual symbol probabilities are .31 for `A` and `T` and .18 for `G` and `C`

C	G	G	C	G	G	-
g	G	G	C	G	G	-
-	G	G	C	G	G	-
-	G	G	C	G	G	C
C	G	G	C	G	G	C

Table 8: Alignment of four highly overlapping words picked from the z_2-tree of Table 6

G	g	G	C	G	G	-	-	-
G	C	G	C	G	c	-	-	-
-	-	G	C	G	G	C	T	-
-	-	-	C	G	G	C	T	A
G	C	G	C	G	G	C	T	A

Table 9: Alignment of four other highly overlapping words picked from the tree of z_2-tree of Table 6

A	G	C	C	G	C	-	-	-
-	G	C	C	G	C	-	G	-
-	G	C	C	G	C	C	-	-
-	-	C	C	G	C	g	G	-
-	-	C	C	G	g	C	-	A
A	G	C	C	G	C	C	G	A

Table 10: Alignment of five other highly overlapping words picked from the z_2-tree of Table 6

GGCGGC is no more the highest scoring one. GGCGGC ranks fifth after AAAAAA, AAAAA, AAAA, AAA. In fact, as one would expect, the approximation of the variance used in z_3 does affect in particular the scores of highly periodic words.

It is somewhat surprising that the families of words $\mathtt{A}^+$ and $\mathtt{T}^+$ appear as strongly marked in the tree under z_4. The pervasive over-representation of AAA, TTT, TAT, ATA have been reported by Nussinov [37], Brendel *et al.* [12] and Leung *et al.* [31]. They correspond to the highly occurring 'A box' (AATAAAYAAA) and 'T box' (TTWTWTTWTT) binding sites of MARs across different species [22, 1, 35, 9, 47]. In any case, it is comforting to see that the word GGCGGC is still the highest scoring motif in the tree.

The z_8- and z_9-trees pertain both to scores defined in terms of sequence families. It is interesting to compare the tree for the score z_8 with the tree for z_2. The tree for z_8 exposes GCCGTG, TAGCCG, CGCCGA, CCGGCG and AGGACC while the tree annotated with z_2 does not. Vice versa, the tree for z_2 shows GCGCGC, GGCGG, CTTTTC and AAAAAA. The fact that these trees are very similar seems to suggest that, under our conditions, words that occur at least once in unexpectedly many sequences in a family might be spotted just by looking for words with a high occurrence score in the family as a whole. The function z_9 (defined as in WordUp) happens to assign a high score the two motifs GGCGGC and GCGGCT. Surprisingly, it exposes at least two words present in the tree for the score z_2 but not appearing in the tree for z_8, namely, GGCGG and CTTTTC.

We also analyzed the dataset replacing the underlying model with a Markov chain. We produced five trees assuming models of order $M = 0, \ldots, 4$. Order $M = 0$ corresponds to the i.i.d. model, whereas higher orders $M > 0$ assumes that the source is generating symbols with a probability distribution which depends on the M preceding symbols. Figure 11 shows the trees computed by Verbumculus using a common score and parameters. To clarify the differences we produced another figure where we removed the tree, we connected common words with a red line, and we circled in green the singletons. We observe that for $M = 0, 1, 2$ there is some general agreement: in particular the word GGCGGC consistently achieves the highest score. However, for the order 3 we are left only with one word (AGGACC) that does not appear anywhere else. If we had lowered the threshold to 3, GGCGGC would have appeared among other other fifteen words. The fourth order tree contains the prefix GGCGG, although other three unknown words scores a little higher. This phenomenon can be expected, since we train our model on the sequences themselves. As the size of the model grows its capability of prediction grows as well. Therefore there are less and less surprising words. In order to get roughly the same amount of nodes in the trees, we should have lowered the threshold as M increases.

6.4 The Middle cluster

We describe tests conducted with Verbumculus on other cluster in the same dataset from [15]. This will show that when the core of consensus is not a fixed pattern, but admits instead multi-valued positions, then it becomes more difficult to find by our method.

The cluster Middle is composed by 63 genes, namely: STE5, PBP2, MRPL37, APC11, YSW1, UBC1, EKI1, CDC10, SPS2, SPS1, SPR6, GPI8, CDC26, CDH1, ISC10, CLB6, SUT1, HXT10, PES4, SPR28, CDC20, GNP1, SPR3, YCK3, FET5, CDA2, CDA1, SPS18, CDC5, REV7, PIG1, NMT1, MIP6, SPO20, CNM67, YCK2, SUR4, TEP1, RNH70, BNR1, CDC3, KAR1, CWP1, HYM1, ORC1, NDT80, SPO12, FUS2, ORC3, APC9, CDC16, SSP1, PCT1, STO1, BBP1, MUD13, AUT1, HXT14, SPS4, UBC11, SPR1, HST1, ECM23, SSP2.

The family of upstream sequences should display frequent occurrences of the MSE sites ranging from HDVKNCACAAAAD (most stringent, appearing in 7 sequences) to DNCRCAAAWD (least stringent,

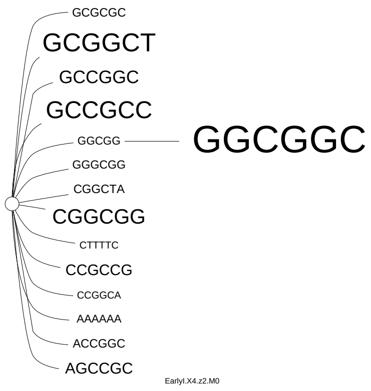
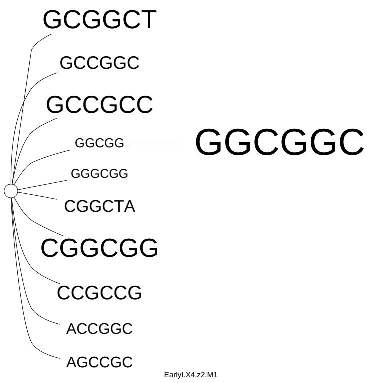
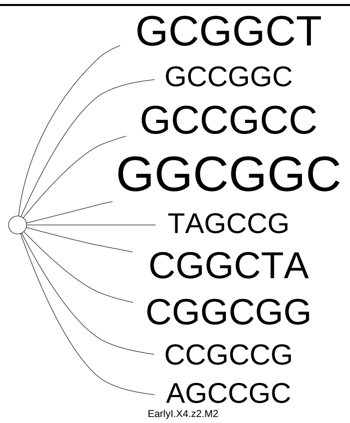
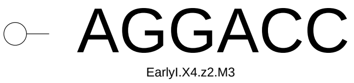
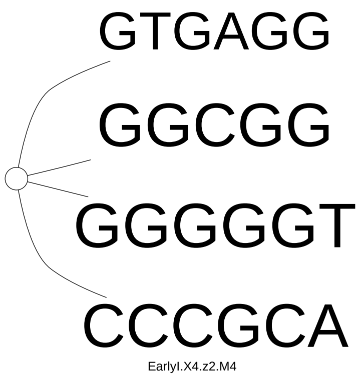
M	Verbumculus	M	Verbumculus
0	 GCGCGC GCGGCT GCCGGC GCCGCC GGCGG GGCGGC GGGCGG CGGCTA CGGCGG CTTTTC CCGCCG CCGGCA AAAAAA ACCGGC AGCCGC EarlyI.X4.z2.M0	1	 GCGGCT GCCGGC GCCGCC GGCGG GGCGGC GGGCGG CGGCTA CGGCGG CCGCCG ACCGGC AGCCGC EarlyI.X4.z2.M1
2	 GCGGCT GCCGGC GCCGCC GGCGGC TAGCCG CGGCTA CGGCGG CCGCCG AGCCGC EarlyI.X4.z2.M2	3	
4			

Table 11: `Early(I)` cluster, score z_2, threshold 4.0, M is the order of the Markov chain

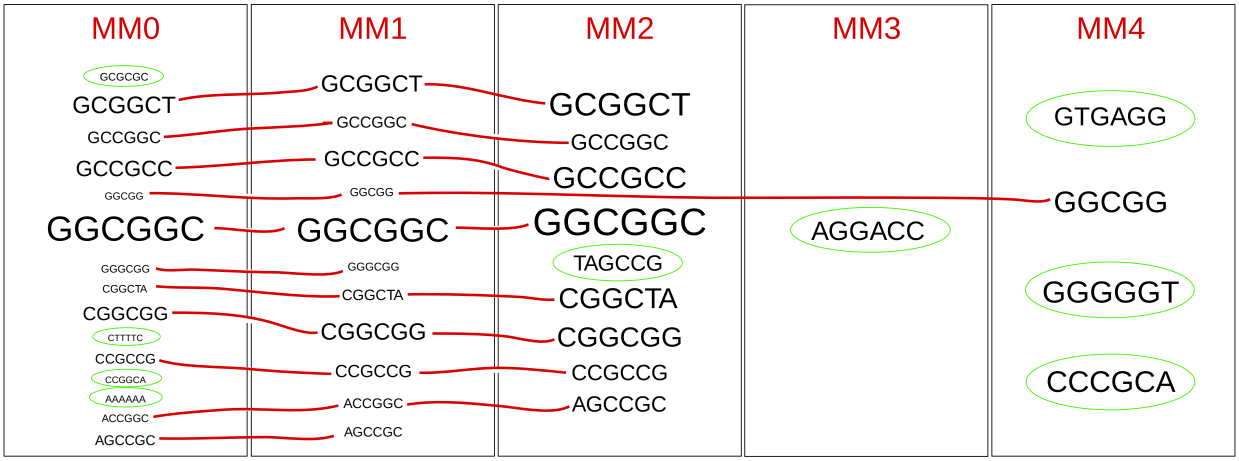


Figure 9: The collection of words from Figure 11. Identical words are connected with lines, while singleton words are circled

	occurrences		*sequences*	
w	$E(w)$	$f(w)$	$E_c(w)$	$c(w)$
CAAA	242.76	349	62.37	63
CTTT	197.31	346	60.82	62
TTTT	350.02	884	62.65	63
CACAAA	14.11	40	16.09	37
TTTTTT	19.82	78	30.42	46

Table 12: Statistics for some notable words of the cluster Middle (63 sequences)

appearing in 31 sequences). The complication for VERBUMCULUS is that these patterns are not fixed strings but rather regular expression, however trivial: for example N is a wildcard denoting any letter in the set A, C, G,T, whereas R can be substituted only with a purine (A or G), Y with a pyrimidine (C or T), etc.

We would like to isolate the core CAAA or, even better, CACAAA from the upstream regions relative to the genes in the cluster Middle. Six sequences share another prominent motif, namely CWBYSCTTT.

Unfortunately, it seems difficult to catch CACAAA in Middle. The z_2-tree in Table 13 shows the 15 words with highest z_2 score: CACAAA does not show up among them. We have to lower the threshold on that score to 5.0 before we can see CACAAA, but this has the simultaneous effect of raising the tree size to almost 100 nodes. The same happens when we look for CAAA (see Table 13-right). However, at least CTTT pops up now among the highest scoring motifs. Table 12 shows the statistics of these and other notable words.

We used other scores to see if we could get CACAAA somewhere. Table 13 shows the z_4-tree for the cluster Middle that suffers again from the presence of the family of words $\mathtt{A}^+$ and $\mathtt{T}^+$, but no CACAAA. It also shows the z_8-tree, but again no CACAAA. Finally, a surprise. The z_9-tree shows CACAAA in the high scoring patterns.

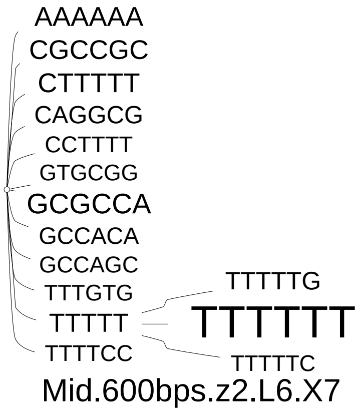
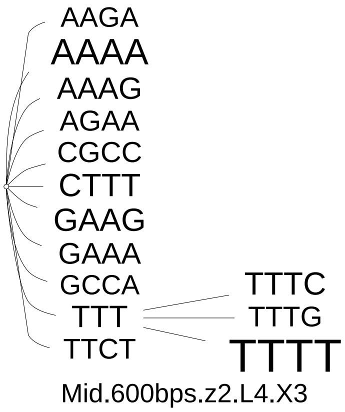
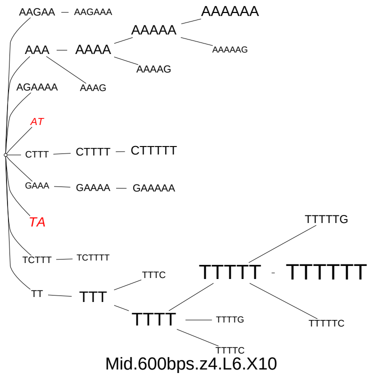
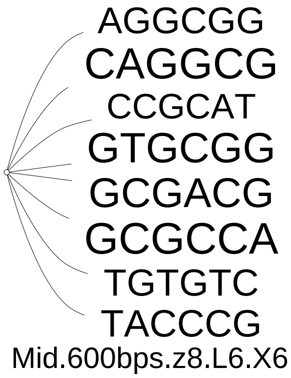
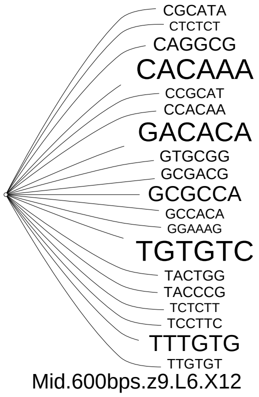
Score	T	Verbumculus	Score	T	Verbumculus
z_2	7.0	 AAAAAA CGCCGC CTTTTT CAGGCG CCTTTT GTGCGG GCGCCA GCCACA GCCAGC TTTGTG TTTTT TTTTCC TTTTTG TTTTTT TTTTTC Mid.600bps.z2.L6.X7	z_2	3.0	 AAGA AAAA AAAG AGAA CGCC CTTT GAAG GAAA GCCA TTT TTCT TTTC TTTG TTTT Mid.600bps.z2.L4.X3
z_4	10.0	 AAGAA AAGAAA AAAAAA AAAAA AAA AAAA AAAAAG AAAAG AGAAAA AAAG AT CTTT CTTTT CTTTTT GAAA GAAAA GAAAAA TA TTTTTG TCTTT TCTTTT TTTC TTTTT TTTTTT TT TTT TTTT TTTTG TTTTTC TTTTC Mid.600bps.z4.L6.X10	z_8	6.0	 AGGCGG CAGGCG CCGCAT GTGCGG GCGACG GCGCCA TGTGTC TACCCG Mid.600bps.z8.L6.X6
z_9	12.0	 CGCATA CTCTCT CAGGCG CACAAA CCGCAT CCACAA GACACA GTGCGG GCGACG GCGCCA GCCACA GGAAAG TGTGTC TACTGG TACCCG TCTCTT TCCTTC TTTGTG TTGTGT Mid.600bps.z9.L6.X12			

Table 13: `Middle` cluster, T is the threshold

7 Conclusion

This paper describes a genome-wise searching system for over- or under-represented nucleotide motifs, called VERBUMCULUS. The main advantages brought about by this tool are speed, low memory requirements and visualization capabilities. This rests on the core structure of the algorithm, which takes advantage of strong properties at the intersection of statistics, pattern matching and combinatorics on words. As a result, the facility at the outset can detect over- or under-represented patterns in linear time and space for most of the scores in use. An array of experimental tests, ranging from simulations on synthetic data to the discovery of regulatory elements on the upstream regions of a set of genes of the yeast, was used to demonstrate the strengths as well as some limitations of the tool.

References

[1] AMATI, B., AND GASSER, S. Drosophila scaffold-attached regions bind nuclear scaffolds and can function as MARS elements in both budding and fission yeast. *Mol. Cell. Biol. 10* (1990), 5442–5454.

[2] APOSTOLICO, A. The myriad virtues of suffix trees. In *Combinatorial Algorithms on Words*, A. Apostolico and Z. Galil, Eds., vol. 12 of *NATO Advanced Science Institutes, Series F.* Springer-Verlag, Berlin, 1985, pp. 85–96.

[3] APOSTOLICO, A. Pattern discovery and the algorithmics of surprise. In *Artificial Intelligence and Heuristic Methods for Bioinformatics* (2003), P. Frasconi and R. Shamir, Eds., IOS Press, pp. 111–127.

[4] APOSTOLICO, A., BOCK, M. E., AND LONARDI, S. Monotony of surprise and large-scale quest for unusual words (extended abstract). In *Proc. of Research in Computational Molecular Biology (RECOMB)* (Washington, DC, April 2002), G. Myers, S. Hannenhalli, S. Istrail, P. Pevzner, and M. Waterman, Eds. Also in *J. Comput. Bio.*, 10:3-4, (July 2003), 283–311.

[5] APOSTOLICO, A., BOCK, M. E., LONARDI, S., AND XU, X. Efficient detection of unusual words. *J. Comput. Bio. 7*, 1/2 (Jan. 2000), 71–94.

[6] APOSTOLICO, A., AND GALIL, Z., Eds. *Pattern matching algorithms.* Oxford University Press, 1997.

[7] APOSTOLICO, A., AND GIANCARLO, R. Sequence alignment in molecular biology. *J. Comput. Bio. 5*, 2 (1998), 173–196.

[8] BAILEY, T. L., AND ELKAN, C. Unsupervised learning of multiple motifs in biopolymers using expectation maximization. *Machine Learning 21*, 1/2 (1995), 51–80.

[9] BOULIKAS, T. Chromatin domains and prediction of MAR sequences. *Int. Rev. Cytol. 162A* (1995), 279–388.

[10] BRĀZMA, A., JONASSEN, I., EIDHAMMER, I., AND GILBERT, D. Approaches to the automatic discovery of patterns in biosequences. *J. Comput. Bio. 5*, 2 (1998), 277–304.

[11] BRĀZMA, A., JONASSEN, I., UKKONEN, E., AND VILO, J. Predicting gene regulatory elements in silico on a genomic scale. *Genome Research 8*, 11 (1998), 1202–1215.

[12] BRENDEL, V., BECKMANN, J. S., AND TRIFONOV, E. N. Linguistics of nucleotide sequences: Morphology and comparison of vocabularies. *J. Biomol. Struct. Dynamics 4*, 1 (1986), 11–21.

[13] BUHLER, J., AND TOMPA, M. Finding motifs using random projections. *J. Comput. Bio. 9*, 2 (2002), 225–242.

[14] CALIFANO, A. SPLASH: Structural pattern localization analysis by sequential histograming. *Bioinformatics 15* (2000), 341–357.

[15] Chu, S., DeRisi, J. L., Eisen, M. B., Mulholland, J., Bodstein, D., Brown, P. O., and Herskowitz, I. The transcriptional program of sporulation in budding yeast. *Science 282* (Oct. 1998), 699–705.

[16] Collins, F., Patrinos, A., Jordan, E., Chakravarti, A., Gesteland, R., Walters, L., and the members of the DOE and NIH planning groups. New goals for the U.S. human genome project: 1998-2003. *Science 282* (1998), 682–689.

[17] DeRisi, J. L., Iyer, V. R., and Brown, P. O. Exploring the metaboblic and genetic control of gene expression on a genomic scale. *Science 278* (1997), 680–686.

[18] Eskin, E., and Pevzner, P. A. Finding composite regulatory patterns in DNA sequences. In *Proc. of the International Conference on Intelligent Systems for Molecular Biology* (2002), AAAI press, Menlo Park, CA, pp. Bioinformatics S181–S188.

[19] Fleischmann, R., Adams, M., White, O., Clayton, R., and Kirkness, E. *et al.* Whole-genome random sequencing and assembly of *Haemophilus influenzae Rd. Science 269* (1995), 496–512.

[20] Gansner, E. R., Koutsofios, E., North, S., and Vo, K.-P. A technique for drawing directed graphs. *IEEE Trans. Software Eng. 19*, 3 (1993), 214–230.

[21] Gasser, S. Nuclear scaffold and high-order folding of eukaryotic DNA. In *Architecture of Eukaryotic Genes*, G. Kahl, Ed. VCH Verlagsgeselschaft, Wienheim, Germany, 1988, pp. 461–471.

[22] Gasser, S., and Laemmli, U. Cohabitation of scaffold binding regions with upstream/enhancer elements of three developmentally regulated genes of *D. melanogaster. Cell 46* (1986), 521–530.

[23] Gusfield, D. *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology.* Cambridge University Press, 1997.

[24] Guyer, M., and Collins, F. How is the human genome project doing, and what have we learned so far? *Proc. Natl. Acad. Sci. U.S.A. 92* (1995), 10841–10848.

[25] Hertz, G. Z., and Stormo, G. D. Identifying DNA and protein patterns with statistically sign ificant alignments of multiple sequences. *Bioinformatics 15* (1999), 563–577.

[26] Hui, L. C. K. Color set size problem with applications to string matching. In *Proceedings of the 3rd Annual Symposium on Combinatorial Pattern Matching* (Tucson, AZ, 1992), A. Apostolico, M. Crochemore, Z. Galil, and U. Manber, Eds., no. 644 in Lecture Notes in Computer Science, Springer-Verlag, Berlin, pp. 230–243.

[27] Jonassen, I. Efficient discovery of conserved patterns using a pattern graph. *Comput. Appl. Biosci. 13* (1997), 509–522.

[28] Jonassen, I., Collins, J. F., and Higgins, D. G. Finding flexible patterns in unaligned protein sequences. *Protein Science 4* (1995), 1587–1595.

[29] Keich, and Pevzner. Finding motifs in the twilight zone. In *Annual International Conference on Computational Molecular Biology* (Washington, DC, Apr. 2002), pp. 195–204.

[30] Lawrence, C. E., Altschul, S. F., Boguski, M. S., Liu, J. S., Neuwald, A. F., and Wootton, J. C. Detecting subtle sequence signals: A Gibbs sampling strategy for multiple alignment. *Science 262* (Oct. 1993), 208–214.

[31] Leung, M. Y., Marsh, G. M., and Speed, T. P. Over and underrepresentation of short DNA words in herpesvirus genomes. *J. Comput. Bio. 3* (1996), 345–360.

[32] Lockhart, D., Dong, H., Byrne, M., Follettie, M., Gallo, M., Chee, M., Mittmann, M., Wang, C., Kobayashi, M., Horton, H., and Brown, E. Expression monitoring by hybridization to high-density oligonucleotide arrays. *Nature Biotechnology 14* (1996), 1675–1680.

[33] Luche, R. M., Sumrada, R., and Cooper, T. G. A cis-acting element present in multiple genes serves as a repressor protein binding site for the yeast CAR1 gene. *Mol. Cell. Biol. 10* (1990), 3884–3895.

[34] McCreight, E. M. A space-economical suffix tree construction algorithm. *J. Assoc. Comput. Mach. 23*, 2 (Apr. 1976), 262–272.

[35] McKnight, R. A., Shamay, A., Sankaran, L., Wall, R., and Hennighausen, L. Matrix-attachment regions can impart position-independent regulation of a tissue-specific gene in transgenic mice. *Proc. Natl. Acad. Sci. 89* (1992), 6943–6947.

[36] Neuwald, A., Liu, J., and Lawrence, C. Gibbs motif sampling: Detecting bacterial outer membrane protein repeats. *Protein Science 4* (1995), 1618–1632.

[37] Nussinov, R. Strong adenine clustering in nucleotide sequences. *J. Theor. Biol. 85* (1980), 285–291.

[38] Pavesi, G., Mauri, G., and Pesole, G. An algorithm for finding signals of unknown length in DNA sequences. In *Proc. of the International Conference on Intelligent Systems for Molecular Biology* (2001), AAAI press, Menlo Park, CA, pp. S207–S214.

[39] Pesole, G., Prunella, N., Liuni, S., Attimonelli, M., and Saccone, C. Wordup: An efficient algorithm for discovering statistically significant patterns in DNA sequences. *Nucleic Acids Res. 20*, 11 (1992), 2871–2875.

[40] Pevzner, P. A., and Sze, S.-H. Combinatorial approaches to finding subtle signals in DNA sequences. In *Proc. of the International Conference on Intelligent Systems for Molecular Biology* (2000), AAAI press, Menlo Park, CA, pp. 269–278.

[41] Rigoutsos, I., and Floratos, A. Combinatorial pattern discovery in biological sequences: The Teiresias algorithm. *Bioinformatics 14*, 1 (1998), 55–67.

[42] Schbath, S. An efficient statistic to detect over- and under-represented words in DNA sequences. *J. Comput. Bio. 4* (1997), 189–192.

[43] Schbath, S., Prum, B., and de Turckheim, E. Exceptional motifs in different Markov chain models for a statistical analysis of DNA sequences. *J. Comput. Bio. 2* (1995), 417–437.

[44] Schena, M., Shalom, D., Davis, R., and Brown, P. Quantitative monitoring of gene expression patterns with a complementary DNA microarray. *Science 270* (1995), 467–470.

[45] Stief, A., Winter, D., Stratling, W., and Sippel, A. A nuclear DNA attachment element mediates elevated and position-independent gene activity. *Nature 341* (1989), 343–345.

[46] Strich, R., Surosky, R. T., Steber, C., Dubois, E., Messenguy, F., and Esposito, R. E. UME6 is a key regulator of nitrogen repression and meiotic development. *Genes Dev. 8* (1994), 796–810.

[47] Strissel, P., Dann, H., Pomykala, H., Diaz, M., Rowley, J., and Olopade, O. Scaffold-associated regions in the human type I interferon gene cluster on the short arm of chromosome 9. *Genomics 47* (1998), 217–229.

[48] Stückle, E., Emmrich, C., Grob, U., and Nielsen, P. Statistical analysis of nucleotide sequences. *Nucleic Acids Res. 18*, 22 (1990), 6641–6647.

[49] van Helden, J., André, B., and Collado-Vides, J. Extracting regulatory sites from the upstream region of the yeast genes by computational analysis of oligonucleotides. *J. Mol. Biol. 281* (1998), 827–842.

[50] Wingender, E., Chen, X., Hehl, R., Karas, H., Liebich, I., Matys, V., Meinhardt, T., Prüss, M., Reuter, I., and Schacherer, F. Transfac: An integrated system for gene expression regulation. *Nucleic Acids Res. 28* (2000), 316–319. `http://transfac.gbf-braunschweig.de/TRANSFAC/`.

[51] Wingender, E., Dietze, P., Karas, H., and Knüppel, R. Transfac: A database on transcription factors and their DNA binding sites. *Nucleic Acids Res. 24* (1996), 238–241. `http://transfac.gbf-braunschweig.de/TRANSFAC/`.

[52] Yada, T., Totoki, Y., Ishikawa, M., Asai, K., and Nakai, K. Automatic extraction of motifs represented in the hidden Markov model from a number of DNA sequences. *Bioinformatics 14* (1998), 317–325.