Regulatory Element Detection using a Probabilistic Segmentation Model

Harmen J. Bussemaker[1], Hao Li[2,3], and Eric D. Siggia[2,4]

[1]Swammerdam Institute for Life Sciences and Amsterdam Center for Computational Science, University of Amsterdam, Kruislaan 318, 1098 SM Amsterdam, The Netherlands; [2]Center for Studies in Physics and Biology, The Rockefeller University, Box 25, 1230 York Avenue, New York, N.Y. 10021; [3]Current Address: Departments of Physiology and Cellular and Molecular Pharmacology, University of California, San Francisco, CA 94143; [4]Department of Physics, Cornell University, Ithaca, N.Y. 14853-2501

bussemaker@bio.uva.nl, haoli@phy.ucsf.edu, siggia@eds1.rockefeller.edu

Abstract

The availability of genome-wide mRNA expression data for organisms whose genome is fully sequenced provides a unique data set from which to decipher how transcription is regulated by the upstream control region of a gene. A new algorithm is presented which decomposes DNA sequence into the most probable "dictionary" of motifs or words. Identification of words is based on a probabilistic segmentation model in which the significance of longer words is deduced from the frequency of shorter words of various length. This eliminates the need for a separate set of reference data to define probabilities, and genome-wide applications are therefore possible. For the 6000 upstream regulatory regions in the yeast genome, the 500 strongest motifs from a dictionary of size 1200 match at a significance level of 15 standard deviations to a database of cis-regulatory elements. Analysis of sets of genes such as those up-regulated during sporulation reveals many new putative regulatory sites in addition to identifying previously known sites.

Introduction

The detection of multiple regulatory elements from a large set of upstream regulatory regions is a challenging problem in the field of computational genomics. One approach has been to delineate, as sharply as possible, a group of 10-100 co-regulated genes (Eisen *et al.* 1998; van Helden, André, & Collado-Vides 1998) and then find a pattern common to most of the upstream regions. The analysis tools employed range from general multi-alignment algorithms yielding a weight matrix (Stormo & Hartzell 1989; Lawrence *et al.* 1993; Bailey & Elkan 1994) to comparison of the frequency counts of substrings with some reference set (van Helden, André, & Collado-Vides 1998). These approaches typically reveal a small number of responsive elements per group of genes in the specific experiment analyzed.

Here we present a new algorithm suitable for discovering multiple motifs from a large collection of sequences, e.g., all the upstream regions of the yeast genome, or all the genes up-regulated during sporulation. The approach we took formalizes how one would proceed to decipher a "text" consisting of a long string of letters written in an unknown language that does not delineate words. The algorithm is based on a statistical mechanics model in which the sequence of symbols is segmented probabilistically into words and a "dictionary" of words is built concurrently. Our algorithm can simultaneously find hundreds of different motifs, each of them only shared by a small subset of the sequences, e.g., 10 to 100 copies out of 6000 upstream regions in the yeast genome. The algorithm does not need an external reference data set to calibrate probabilities, and finds the optimal lengths of motifs automatically.

We start by formulating the model, and explain in some detail how the probabilities associated with words in the dictionary can be fixed using a maximum-likelihood principle. Next we show how a consistent dictionary for a sequence set can be constructed iteratively, starting from the alphabet only. We illustrate the approach by segmenting a string of letters derived from an English novel, and end by discussing the dictionary obtained for the combined set of upstream regions in the yeast genome.

A Dictionary-Based Sequence Model

At the heart of our approach to analyzing an unknown DNA sequence is the assumption that the sequence data can be viewed as the output of a statistical model generating sequences according to well-defined rules. We use a model in which the sequence is a random concatenation of building blocks that are added with a certain probability at each step, independent of what was added earlier; these building blocks however are not single bases but oligonucleotides of various length. We will refer to the building blocks as *words*, and each word α has a probability p_α associated with it. The probabilities are normalized: $\sum_\alpha p_\alpha = 1$.

The set of all words and their associated probabilities is referred to as the *dictionary*. Not all words of a given length have to be present in the dictionary. This holds in particular for the longer words, and a great reduction in number of parameters with respect to Markov models is the result. In general there will be correlations between neighboring bases in the sequences output by the model: the model has no memory at the level of words, but this is not true at the level of bases. Note the distinction between the probability of a dictionary word and the frequency of a string in the data. The latter "knows" nothing about the dictionary nor respects the word boundaries which vary from realization to realization of the data. The frequency of a string which is also a dictionary word is at least as great as the word probability, but could be substantially larger due to the chance juxtaposition of fragments. Dictionary words for which the two are nearly equal are of high statistical significance, and these are the ones we consider as biologically relevant. There is no sharp distinction between signal and background, although short words tend to have lower significance.

Maximum-Likelihood Procedure

Suppose we are handed a sequence S that was generated using a dictionary for which we know the words but not p_α. It turns out that if S is long enough we can recover these probabilities by maximizing the likelihood of observing S with respect to p_α under the constraint $\sum_\alpha p_\alpha = 1$. More explicitly we have

$$\text{(likelihood of observing } S) = \frac{Z(S)}{\sum_{S'} Z(S')}. \qquad (1)$$

The partition function $Z(S)$ is defined as

$$Z(S) = \sum_P \prod_\alpha (p_\alpha)^{N_\alpha(P)}, \qquad (2)$$

where a sum is performed over all possible partitionings or segmentations P of the sequence S into dictionary words, i.e., all possible ways the sequence can be generated from concatenations of words from the dictionary, and $N_\alpha(P)$ denotes the number of words of type α used in partitioning P.

The denominator $\sum_{S'} Z(S')$, in which the sum is over all sequences whose length equals that of S, acts as a normalization constant, but it is essentially equal to unity for large enough sequences, so that it can be omitted from the analysis. The object to be maximized is therefore the partition function $Z(S)$, which we will simply refer to as Z in the remainder of this paper.

To see why the denominator is very close to unity, observe that we can estimate it by

$$\sum_{S'} Z(S') \simeq \left(\sum_\alpha p_\alpha \right)^{N_{\rm av}} = 1. \qquad (3)$$

In this expression $N_{\rm av}$ equals the average number of words in the partitioning of S' and the second equality follows from the normalization $\sum_\alpha p_\alpha = 1$. A more rigorous analysis using complex contour integration shows that the strict length constraint causes corrections to this result that decreases exponentially with L and thus can be safely ignored.

Fixed-Point Equation

Just like in statistical physics, the partition function $Z(S)$ can be used as a generating function for averages and correlation functions by taking derivatives. For instance, the average number of words of each type used to partition the sequence can be calculated by taking the derivative with respect to p_α:

$$\begin{aligned} \langle N_\alpha \rangle &= p_\alpha (\partial/\partial p_\alpha) \log Z \\ &= \frac{1}{Z} \sum_P \prod_\alpha N_\alpha(P) (p_\alpha)^{N_\alpha(P)}. \end{aligned} \qquad (4)$$

Using a Lagrange multiplier λ for the constraint $\sum_\alpha p_\alpha = 1$, it can be shown that the requirement that $Z(S) - \lambda(\sum_\alpha p_\alpha - 1)$ be stationary with respect to λ and p_α for all α is equivalent to

$$p_\alpha = \frac{\langle N_\alpha \rangle}{\sum_\beta \langle N_\beta \rangle} \equiv f_\alpha. \qquad (5)$$

Using a vector notation where $\mathbf{p} = (p_\alpha | \alpha \in D)$ contains the probabilities for all words in the dictionary, this equation can be written as $\mathbf{p} = \mathbf{f}(\mathbf{p})$, where each entry in the vector $\mathbf{f}$ depends on all components of $\mathbf{p}$. Finding the vector $\mathbf{p}$ that maximizes Z is equivalent to finding the fixed point $\mathbf{p}^\star$ of the nonlinear mapping $\mathbf{f}(\mathbf{p})$, satisfying $\mathbf{f}(\mathbf{p}^\star) = \mathbf{p}^\star$.

Quasi-Invariance and Slow Convergence

In principle one can find the fixed point by starting with an arbitrary initial $\mathbf{p}(0)$, and then converge towards $\mathbf{p}^\star$ by iterating the mapping: $\mathbf{p}(0) \to \mathbf{p}(1) = \mathbf{f}(\mathbf{p}(0)) \to \mathbf{p}(2) = \mathbf{f}(\mathbf{p}(1)) \to \cdots$. In practice however this procedure is very slow, the reason being the existence of what we will call quasi-invariances in $\mathbf{p}$-space. To explain this, let us compare two dictionaries. The first dictionary only contains single-letter words $\alpha = r$ with probabilities p_r; here r denotes a letter from the alphabet. The second dictionary also contains length-two words $\alpha = rs$ made of letters r and s, and its probabilities p'_r and p'_{rs} are related to those of the first one by

$$p'_r = (1-\rho) p_r \qquad p'_{rs} = \rho\, p_r p_s. \qquad (6)$$

Here ρ is a parameter that can take arbitrary values between 0 and 1, The intuitive idea behind this transformation is that when a symbol r is about to be added to a random sequence, one could decide to postpone adding it with probability ρ, draw a second symbol r' and then add both symbols to the sequence at the same time as a length-two word. Of course nothing is really changed by this transformation and the two dictionaries are completely equivalent (hence the term invariance). In particular, the invariance means that $Z(\mathbf{p}') = Z(\mathbf{p})$, even though $\mathbf{p}'$ and $\mathbf{p}$ can be quite different. The invariance transformation can be generalized by using independent parameters ρ_r for each r (the probability with which addition of the first symbol is postponed depends on which symbol was drawn) and the corresponding families of equivalent dictionaries lie on a multi-dimensional manifold in $\mathbf{p}$-space. Similar invariance transformations exist for general dictionaries containing words of arbitrary length.

If a fixed point $\mathbf{p}^\star$ happens to lie on an invariant manifold, it follows from the invariance that all other points on the manifold are fixed points of the mapping as well. Therefore, starting from a vector $\mathbf{p}$ outside the manifold, iteration will have the effect of rapidly moving $\mathbf{p}$ closer to the manifold along perpendicular directions, but there will be no changes in $\mathbf{p}$ in directions parallel to the manifold. The invariant manifolds corresponding to the various families of equivalent models lie rather dense in $\mathbf{p}$-space. It follows that in general the fixed point $\mathbf{p}^\star$ will not be very far from an invariant manifold, and a quasi-invariance holds that will cause iteration of the mapping to be quite slow in the parallel directions. Fortunately, we can exploit our geometrical insight in this problem to improve the converge, as decribed in what follows.

Description of Convergence Procedure

To overcome the problems caused by the quasi-invariance described above, we have implemented a convergence scheme consisting of three parts:

1. Simple iteration of mapping $\mathbf{p} \to \mathbf{f}(\mathbf{p})$;
2. Iteration via powers of the matrix of the linearized map;
3. Newton's method near the fixed point.

We will now describe these steps steps in some detail. The first step consists of performing a number of straightforward iterations of the mapping $\mathbf{p} \to \mathbf{f}(\mathbf{p})$ until the difference between $\mathbf{p}$ and $\mathbf{f}(\mathbf{p})$ becomes small enough to make linearization of the mapping useful. To this end we define the matrix $\partial\mathbf{f}/\partial\mathbf{p}$ with entries $(\partial\mathbf{f}/\partial\mathbf{p})_{\alpha\beta} = \partial f_\alpha/\partial p_\beta$. It will serve as a basis for both part 2 and 3 of the convergence procedure.

The eigenvalues of $\partial\mathbf{f}/\partial\mathbf{p}$ give an indication of whether we are near a stable fixed point, as in that case all eigenvalues should have an absolute value smaller than unity. When this is true, Newton's method provides an efficient way of finding the fixed point: linearizing $\mathbf{f}(\mathbf{p})$ around the current probability vector $\mathbf{p}$, the approximate fixed point $\mathbf{p}^*$ is solved from the vector equation

$$\mathbf{f}(\mathbf{p}) + \left.\frac{\partial\mathbf{f}}{\partial\mathbf{p}}\right|_{\mathbf{p}} (\mathbf{p}^* - \mathbf{p}) = \mathbf{p}^*. \tag{7}$$

Due to the nonlinearity of $\mathbf{f}(\mathbf{p})$, this procedure may have to be repeated a number of times to achieve convergence to machine accuracy, but the convergence is quadratic and therefore quite rapid (Press *et al.* 1992). The quasi-invariance discussed above are responsible for the fact that one or more eigenvalues of $\partial\mathbf{f}/\partial\mathbf{p}$ are very close to unity, so that the component of $\mathbf{p} - \mathbf{p}^*$ parallel to the associated eigenvector may be quite large, thus overcoming the slow convergence associated with straightforward iteration.

The Newton method constitutes the final and third step of our convergence scheme. It is preceded by a procedure that is followed when there are still one or more unstable eigenvalues with an absolute value larger than unity. In this case $\mathbf{p}$ lies in a region of mixed stability, a situation that frequently occurs in the course of finding the fixed point as we found. The same quasi-invariance that slow the convergence to the fixed point when straightforward iteration is applied now prevent the escape from the mixed-stability region along the unstable direction. We have implemented a scheme that applies powers of the linearized mapping as an efficient way of getting out of the mixed-stability region. Starting from $\mathbf{p}(0)$ and $\mathbf{p}(1) \equiv \mathbf{f}(\mathbf{p}(0))$, and defining $\mathbf{p}(n) = \mathbf{f}^n(\mathbf{p})$ as the n-th iteration of the mapping, we have in linear approximation the following set of recursion relations:

$$\begin{aligned} \mathbf{p}(2) - \mathbf{p}(1) &= [\partial\mathbf{f}/\partial\mathbf{p}]\,(\mathbf{p}(1) - \mathbf{p}(0)) \\ \mathbf{p}(4) - \mathbf{p}(2) &= [\partial\mathbf{f}/\partial\mathbf{p}]^2\,(\mathbf{p}(2) - \mathbf{p}(0)) \end{aligned} \tag{8}$$

etc. The series $\mathbf{p}(2^m)$ with $m = 0, 1, 2, 3, \ldots$ is monitored and extended until Z no longer increases or the constraint $0 < p_\alpha < 1$ is violated, in which case the last iteration for which $F(\mathbf{p}(2^m)) < F(\mathbf{p}(0))$ is used. In our experience, repeated application of this procedure would always move $\mathbf{p}$ to a region where all eigenvalues were stable and Newton's method could be applied. We also found that no matter from which $\mathbf{p}$ one starts, the same fixed point $\mathbf{p}^*$ was reached. The three parts of our convergence scheme together therefore constitute a robust method for determining the apparently unique optimal set of dictionary probabilities.

Computation of $\partial\mathbf{f}/\partial\mathbf{p}$ and other quantities

We will now discuss the details of calculating $\partial\mathbf{f}/\partial\mathbf{p}$ from the dictionary probabilities. It is useful to define a free energy function F as the negative of the logarithm of the partion function Z,

$$F = -\log Z. \tag{9}$$

Solving the fixed point equation is equivalent to minimizing F. Using

$$\langle N_\alpha \rangle = -p_\alpha (\partial/\partial p_\alpha) F, \tag{10}$$

it follows from the definition of f_α in Eq. (5) that

$$\begin{aligned} \frac{\partial f_\alpha}{\partial p_\beta} &= \left(p_\alpha \frac{\partial^2 F}{\partial p_\alpha \partial p_\beta} + \delta_{\alpha\beta} \frac{\partial F}{\partial p_\beta} \right) \left(\sum_\gamma p_\gamma \frac{\partial F}{\partial p_\gamma} \right)^{-1} \\ &\quad - \left(p_\alpha \frac{\partial F}{\partial p_\alpha} \right) \left(\frac{\partial F}{\partial p_\beta} + \sum_\gamma p_\gamma \frac{\partial^2 F}{\partial p_\gamma \partial p_\beta} \right) \\ &\quad \times \left(\sum_\gamma p_\gamma \frac{\partial F}{\partial p_\gamma} \right)^{-2}. \end{aligned} \tag{11}$$

Here the Kronecker delta $\delta_{\alpha\beta}$ equals unity if $\alpha = \beta$ and zero otherwise. The calculation of $\partial\mathbf{f}/\partial\mathbf{p}$ involves both the first and second derivatives of F, which can be computed efficiently using recursive methods common in statistical mechanics. To derive a recursion relation for F itself, we start by defining $Z(1, i)$ as the partition sum for the subsequence of S starting at the leftmost position and ending at position i. Let the maximum word length occurring in the dictionary be $\ell_{\max}$. We then have

$$Z(1,i)=\sum_{\ell=1}^{\ell_{\max}} p_{\sigma(i,\ell)} Z(1,i-\ell). \tag{12}$$

Here $\sigma(i,\ell)$ denotes the substring of length ℓ that ends at position i in the sequence S; when $\alpha(i,\ell)$ is not in the dictionary $p_{\alpha(i,\ell)}=0$. The boundary condition for Eq. (12) is given by $Z(1,1)=p_{\sigma(1,1)}$. Iterating up to L, the length of the full sequence S, we can determine $F=-\log Z(1,L)$. For numerical purposes however this form of the recursion relation is not practical, as underflow will occur for large L. We therefore define the ratio

$$R(i)=\frac{Z(1,i)}{Z(1,i-1)}, \tag{13}$$

and rewrite the recursion relation for $Z(1,i)$ as

$$R(i)=p_{\alpha(i,1)}+\sum_{\ell=2}^{\ell_{\max}} p_{\alpha(i,\ell)}\left(\prod_{k=i-\ell+1}^{i-1} R(k)\right)^{-1}. \tag{14}$$

The free energy can be calculated as

$$F=-\sum_i \log R(i). \tag{15}$$

All $R(i)$ terms are of the same order and no underflow problems will occur. The complexity of the computation of F is $\mathcal{O}(L)$, i.e. the time required to compute F scales linearly with the length of the sequence S. The memory required scales as $\mathcal{O}(\ell_{\max})$.

We will next show how the first derivative of F with respect to $\mathbf{p}$, required for the calculation of both $\mathbf{f}$ and $\partial\mathbf{f}/\partial\mathbf{p}$, can be computed in an efficient manner. The naive numerical differentiation of F with respect to each p_α would take a total time $\mathcal{O}(L\times D)$, where D is the number of words in the dictionary. We will show how to directly compute $\partial F/\partial\mathbf{p}$ using recursion relations similar to those for F, in a time scaling as $\mathcal{O}(L\times\ell_{\max})$. As typically $D\gg\ell_{\max}$, a significant speedup is achieved. The following expression holds for the derivative of F:

$$\frac{\partial F}{\partial p_\alpha}=-\sum_{i=1}^{L} G(i,\ell_\alpha)\delta_{\alpha,\sigma(i,\ell_\alpha)}, \tag{16}$$

where

$$G(i,\ell)=\frac{Z(1,i-\ell)Z(i+1,L)}{Z(1,L)}, \tag{17}$$

and $Z(i+1,L)$ is the partition function of the subsequence from position $i+1$ to the rightmost position. The Kronecker delta in Eq. (16) restricts the sum to positions at which the substring $\sigma(i,\ell_\alpha)$ is precisely the dictionary word α for which the derivative is calculated. When evaluating $\partial F/\partial p_\alpha$ numerically for all dictionary words however, rather than evaluating Eq. (16) for all α, one should compute $G(i,\ell)$ for all $i=1,\ldots,L$ and $\ell=1,\ldots,\ell_{\max}$, determine $\alpha(i,\ell)$ for each pair (i,ℓ) and add $G(i,\ell)$ to the appropriate component of $\partial F/\partial\mathbf{p}$ unless the substring $\alpha(i,\ell)$ does not occur in the dictionary. It is exactly this procedure that allows the speedup from $\mathcal{O}(L\times D)$ to $\mathcal{O}(L\times\ell_{\max})$. To evaluate $G(i,\ell)$ we first introduce $R'(i)$ as the partition function ratio defined with respect to the rightmost position of S,

$$R'(i)=\frac{Z(i,L)}{Z(i+1,L)}. \tag{18}$$

The recursion relation for $R'(i)$ reads

$$R'(i)=p_{\alpha(i,1)}+\sum_{\ell=2}^{\ell_{\max}} p_{\alpha(i+\ell-1,\ell)}\left(\prod_{k=i+1}^{i+\ell-1} R'(k)\right)^{-1}. \tag{19}$$

We can now express $G(i,\ell)$ in terms of $R(i)$ and $R'(i)$ as

$$G(i,\ell)=\left(\prod_{k=1}^{i-\ell}\frac{R(k)}{R'(k)}\right)\left(\prod_{k=i-\ell+1}^{i}\frac{1}{R'(k)}\right). \tag{20}$$

One pays for the speedup by having to keep $R(i)$ and $R'(i)$ in memory, a requirement scaling as $\mathcal{O}(L)$, i.e. on the order of what is needed to keep the sequence S in memory and therefore typically not posing any problems.

Computation of the second derivative matrix $\partial^2F/\partial p_\alpha\partial p_\beta$ occurs along the same lines. We will not provide explicit expressions here. In this case the computation time scales as $\mathcal{O}(L\times\ell_{\max}\times D)$ and the space requirement is still $\mathcal{O}(L)$.

Constructing the Dictionary Iteratively

When it is known which words are present in the dictionary (the lexicon), the maximum-likelihood procedure described above can be used to determine the probabilities $\mathbf{p}$ for all the words given sequence data S. However, when analyzing a set of unknown DNA sequences, one has no a priori information about which words should be in the dictionary. Intuitively, to build a dictionary from the sequence, one would start from the frequency of individual letters, find overrepresented pairs and add them to the dictionary, determine their probabilities, and continue to build larger fragments in this way. Here we describe approaches to add new words to the existing dictionary.

If a given dictionary is supposed to give an accurate statistical description of S, any quantity derived from the sequence should be correctly predicted by the model. In particular, the frequency with which two dictionary words occur as neighbors in a partitioning P of S can be computed and compared with the model prediction. We have

$$\langle N_{\alpha\beta}\rangle=p_\alpha p_\beta\sum_{i=1}^{L} G(i,\ell_\alpha+\ell_\beta)\delta_{\alpha,\sigma(i-\ell_\beta,\ell_\alpha)}\delta_{\beta,\sigma(i,\ell_\beta)}. \tag{21}$$

The model predicts $\langle N_{\alpha\beta}\rangle = N_{\rm av}p_\alpha p_\beta$, where $N_{\rm av} = L/\langle\ell\rangle$ is the average number of words in a partition, with $\langle\ell\rangle = \sum_\alpha \ell_\alpha p_\alpha$ the average word length. There are of course statistical fluctuations arising from the finite length of S. By using a Z-score, quantifying the difference from the expected value in units of the standard deviation of these fluctuations,

$$Z_{\alpha\beta} = \frac{\langle N_{\alpha\beta}\rangle - N_{\rm av}p_\alpha p_\beta}{\sqrt{N_{\rm av}p_\alpha p_\beta}}, \tag{22}$$

all possible pairs (α,β) can be tested for overrepresentation on the same footing.

The detection of overrepresented word pairs suggests an iterative scheme that starts from a dictionary containing only the four single bases. At each step, the probilities **p** are determined, Z-scores are calculated as defined in Eq. (22) and pairs with a Z-score above a specified threshold are added to the dictionary as new words. This procedure is repeated until the Z-score for all word pairs is below threshold. In a more systematic approach that takes the changing size of the dictionary into account, the Z-scores are converted into P values using Gaussian distribution, and a P value threshold is used. Typically we set $P = 1/N_{\rm pair}$, where $N_{\rm pair}$ is the number of pairs tested.

For DNA sequences, it is possible to check all oligomers up to a certain length and add those that are over-represented to the dictionary. A practical upper limit for this length is 8. The threshold cutoff for P value in this case is typically set to $1/N_{\rm oligo}$, where $N_{\rm oligo}$ is the total number of oligomers screened.

An Example Using English Text

To illustrate the power of the iterative approach described above, consider a string of letters from the English alphabet that results when all spaces and special characters are removed from the first ten chapters of the novel "Moby Dick" by Melville:

```
chapterloomingscallmeishmaelsomeyearsago
nevermindhowlongpreciselyhavinglittleorn
omoneyinmypurseandnothingparticulartoint
erestmeonshoreithoughtiwouldsailaboutali
ttleandseethewaterypartoftheworlditisawa
yihaveofdrivingoffthespleenandregulating
thecirculationwheneverifindmyselfgrowing
ynovemberinmysoulwheneverifindmyself....
```

When the initial dictionary {a,b,c,...,z} consisting only of single letters is fit to this sequence, the probability p for each word equals the frequency at which it occurs in the sequence data:

Word	p
e	0.120937
t	0.091602
a	0.081246
o	0.077359
n	0.070587
i	0.068853
s	0.066141
l	0.043776
⋮	⋮

To determine which new words should be added to the dictionary, overrepresented adjacent pairs of letters are identified by calculating the Z-score for all pairs. The result of this procedure, sorted by Z-score, sorted as follows:

Left Word	Right Word	Z-score
t	h	106.20
n	g	71.09
i	n	68.42
q	u	67.12
h	e	64.02
⋮	⋮	⋮
t	n	-21.80
a	o	-22.77
a	a	-23.95
o	e	-26.34
a	e	-28.84

Only the most positively and negatively correlated pairs are shown, but there are many more significantly overrepresented word pairs. These are all added to the dictionary, and the new dictionary is fit to the sequence data:

Word	p
e	0.094990
s	0.063027
t	0.054711
a	0.054319
i	0.049299
th	0.039866
o	0.039622
d	0.038558
r	0.030454
n	0.028323
⋮	⋮

The combined steps of adding overrepresented pairs to the dictionary followed by a new fit represent one step in an iterative procedure. After another such iteration step has been performed, the dictionary is:

Word	p
e	0.072890
s	0.067527
a	0.043389
t	0.040361
d	0.036406
i	0.034153
the	0.025849
o	0.022180
r	0.018498
f	0.018392
⋮	⋮

After only three such iterations, words up to length eight can be present in the dictionary.

To quantify the significance of a word α with respect to its fragments, we define a significance score or word quality Q as

$$Q_\alpha = \frac{N_{\rm av} p_\alpha}{\sqrt{N_{\rm av}(Z(\alpha) - p_\alpha)}}, \qquad (23)$$

where $Z(\alpha)$ is the partition function for the word α. Note that Q has the character of a Z-score and gives a measure of how much more often the word α occurs in S than is expected by chance due to concatenations of fragments of α that also occur as dictionary words. Alternatively, one could rank words by $\langle N_\alpha \rangle / \Xi_\alpha$, where Ξ_α denotes the total number of occurrences of the word α in the sequence S, in which case one measures with what frequency the string α is delineated as a separate word in the partitioning of S.

That the iteration procedure works well is illustrated by the fact that after three steps the statistically most significant dictionary words are:

Word	Q
enough	7080.30
harpoon	3646.40
black	2874.70
rough	1835.40
would	1294.60
though	1068.60
from	995.20
could	853.40
jona	763.90
hipma	701.60
with	679.50
you	480.30
ship	423.00
think	341.60
and	328.00
world	313.80
whale	253.70
wild	250.50
hould	245.80
back	236.90
whaling	232.70
⋮	⋮

Analyzing Upstream Regions in Yeast

The present paper is intended as a detailed description of our novel, dictionary-based method for discovering motifs in large sequence datasets by means of an intrinsic analysis of the sequence based on statistics alone. We have applied our algorithm to all the upstream regions of the yeast *Saccharomyces cerevisiae* and will describe some of the results in this section. Some of the data on which this discussion is based are available at `www.physics.rockefeller.edu/siggia/ISMB-2000`. Further details are provided in a separate publication (Bussemaker, Li, & Siggia 2000).

For applications to yeast we took for our sequence data all regions upstream of the ~ 6000 ORFs in yeast of maximum length 600bp and not overlapping with any coding region on either strand. All exact repeats of length 16 or greater were removed since almost all of these are poly A strings, transposons, or other repeats that do not contain regulatory sites. The restriction to strictly noncoding sequence increases the density of functional sites somewhat and more importantly does not mix data with very different background statistics as signified, say by the frequency of length 3 words as codons.

A dictionary was prepared starting from single bases, and words were added in order of increasing length so as to fit the data with the fewest parameters possible. After each cycle of word addition, the dictionary probabilities were converged, and marginal words with low Q_α or p_α corresponding to less than one copy per data set were discarded. The prediction of new words based only on the juxtaposition of the current dictionary entries missed relevant signal in contrast to the English example above. Thus we checked that the dictionary model fit the frequency of all oligomers up to length 8, and added those which were underpredicted. (Less than 1% of all 8-mers actually occur in the dictionary, the rest are fit by smaller fragments.) Subsequently we checked clusters of oligomers with similar sequences which can be statistically significant even when the component oligomers agree with theory within fluctuations. Finally, juxtaposition alone was used to predict words longer than 8.

The final dictionary had 1200 words, of which 100 were clusters of similar oligonucleotides. Two thirds of the sequence data was segmented into single letter words, and an additional 15% into words of length 4 or less. About 500 dictionary entries fell above a plausible significance level. These included good matches to about half of the motifs found in (Spellman *et al.* 1998; Chu *et al.* 1998; van Helden, André, & Collado-Vides 1998) including the MCB, SCB, and MCM1 cell-cycle motifs and URS1 for sporulation. We also found ~ 400 significant dimer motifs in the form of two short segments separated by a gap. These could be clustered by sequence similarity into 20-30 groups, which included matches to the ABF1, RAP1, GAL4, and MCM1 sites plus many other statistically significant clusters which did not match any known sites.

Out of the 443 non-redundant sites of length 5 or greater in the database of (Zhu & Zhang 1999), our dictionary words hit 94 by strict inclusion (i.e., our predicted words are inside the experimentally determined sites), and hit 135 sites by 4 or more bases overlap. Our overlap criterion is not unreasonable since almost all the dictionary words are longer than 5 bases, and while the database reports only sites with experimental evidence for function, there is no guarantee that the complete functional region has been isolated. For either definition of "hit" we computed the statistical significance of the number of sites identified by first marking the dictionary words on the data set and then treating the 443 data base sites as intervals of corresponding length to be placed at random in the non-coding regions. This eliminates the effects of correlations in the dictionary words. Under either definition of "hit" the number of sites hit by our dictionary words is 15 standard deviations beyond expectation. The sites hit by the dimer motifs, by inclusion, is 26 sites vs 2.7 expected by chance. As additional controls for the statistical significance of our matches, we scrambled the letters in each dictionary word and hit by inclusion only 33 sites (vs 94 with the dictionary words). (The motif lists of Brazma et al 1998 when subject to the same filters as our dictionary words hit 30 sites by inclusion.) In contrast to less systematic assignments of probability, our dictionary fits the copy number of all strings of length 8 or less, plus certain classes of clusters of which the members are related by a couple of single base mutations. Thus we can say that for the categories of motifs we search for exhaustively, $\sim 64-75\%$ of the experimental sites are *not* statistically over-represented. This of course does not exclude the possibility that some of these experimental sites escape our detection due to their great variability, or some of the experimental assignments are incorrect.

More specific dictionaries were prepared for the upstream regions of all genes that respond to sporulation (Chu *et al.* 1998) or the cell cycle (Spellman *et al.* 1998). They recover most of the known motifs and suggest many new ones. In contrast to other algorithms that have been used on these data sets, we do not perform any preliminary clustering of the expression data and can detect motifs represented in only ~ 10 out of ~ 6000 genes. For the sporulation data set in particular we obtain clusters of dictionary words that match very well with the two well characterized regulatory sites, URS1 (consensus 5'-DSGGCGGC) and MSE (5'-CRCAAAW). Several of our most significant words are refinements of the mid-sporulation element (MSE) and do a better job than the canonical motif in distinguishing genes that respond during sporulation from those that do not. (The canonical MSE pattern is contained in 134 out of 480 up regulated genes (Chu *et al.* 1998), but 832 other genes with the same motif within 600bp upstream do not respond in sporulation.) We also obtained several new clusters, some of which correlate significantly with the expression data.

Previous approaches to the detection of cis-regulatory elements have generally worked from small (100 or less) clusters of genes. The authors (van Helden, André, & Collado-Vides 1998) compare oligomer frequencies between the cluster of interest and a random sample of upstream regions. This does not work genome-wide and requires one to preselect the cluster rather than letting the presence of common motifs define the clusters (which are probably overlapping). Genome-wide approaches that use a random sample of the yeast genome to assign probabilities can be biased by gross differences between coding and noncoding sequence (Brazma *et al.* 1998). Codes that find all maximal regular expressions (those that can not be made more precise without loosing elements) also can characterize a cluster of genes (Rigoutsos & Floratos 1998). However there is no reference to a probability model in the enumeration process, so a long fragment of a dictionary word will be recorded if by chance it has a few more copies than the word itself; furthermore only the simplest possible sequence model (based on single base frequencies only) is used to assess the significance of the results a posteriori. Data compression algorithms, despite a similarity in terminology (e.g. "adaptive dictionary" for the Ziv-Lempel code family), satisfy very different design criteria (an invertible coding constructed on a single pass) than the data model we fit to. They would attempt to compress data constructed by randomly drawing single bases by encoding repeated substrings. Hidden Markov models are a common way to segment biological data, but they are generally employed with relatively few segment types (e.g. promoter, exon, intron) each described by many parameters; we work with many segments, most described by a single parameter. We can not treat as yet long, fuzzy patterns such as protein motifs over the 20 letter alphabet, for which weight matrix methods were designed (Stormo & Hartzell 1989; Lawrence *et al.* 1993; Bailey & Elkan 1994) but there are many interesting motifs in yeast which are less than 10bp long and have only a few variable sites, and thus within the purview of our method.

References

Bailey, T., and Elkan, C. 1994. Fitting a mixture model by expectation maximization to discover motifs in biopolymers. *Proceedings ISMB'94* 28–36.

Brazma, A.; Johnassen, I.; Vilo, J.; and Ukkonen, E. 1998. Predicting gene regulatory elements in silico on a genomic scale. *Genome Res.* 8:1202–1215.

Bussemaker, H. J.; Li, H.; and Siggia, E. D. 2000. Building a dictionary for genomes: Identification of presumptive regulatory sites by statistical analysis. Submitted.

Chu, S.; DeRisi, J.; Eisen, M.; Mulholland, J.; et al. 1998. The transcriptional program of sporulation in budding yeast. *Science* 282:699–705.

Eisen, M. B.; Spellman, P. T.; Brown, P. O.; and Botstein, D. 1998. Cluster analysis and display of genome-wide expression patterns. *Proc. Natl. Acad. Sci. U.S.A.* 95:14863–14868.

Lawrence, C. E.; Altshul, S. F.; Boguski, M. S.; Liu, J. S.; et al. 1993. Detecting subtle sequence signals: a gibbs sampling strategy for multiple alignment. *Science* 262:208–214.

Press, W. H.; Teukolsky, S. A.; Vetterling, W. T.; and Flannery, B. P. 1992. *Numerical Recipes in C.* Cambridge University Press.

Rigoutsos, I., and Floratos, A. 1998. Combinatorial pattern discovery in biological sequences: The TEREISIAS algorithm. *Bioinformatics* 14:55–67.

Spellman, P. T.; Sherlock, G.; Zhang, M. Q.; et al. 1998. Comprehensive identification of cell cycle-regulated genes of the yeast saccharomyces cerevisiae by microarray hybridization. *Mol. Biol. Cell* 9:3273–3297.

Stormo, G. D., and Hartzell, G. W. 1989. Identifying protein-binding sites from unaligned DNA fragments. *Proc. Natl. Acad. Sci. U.S.A.* 86:1183–1187.

van Helden, J.; André, B.; and Collado-Vides, J. 1998. Extracting regulatory sites from the upstream region of yeast genes by computational analysis of oligonucleotide frequencies. *J. Mol. Biol.* 281:827–842.

Zhu, J., and Zhang, M. Q. 1999. SCPD: A promoter database of yeast saccharomyces cerevisiae. *Bioinformatics* 15:607–611.