CABIOS

Vol.5. no.2. 1989
Pages 89 – 96

Methods for calculating the probabilities of finding patterns in sequences

Rodger Staden

Abstract

This paper describes the use of probability-generating functions for calculating the probabilities of finding motifs in nucleic acid and protein sequences. Equations and algorithms are given for calculating the probabilities associated with nine different ways of defining motifs. Comparisons are made with searches of random sequences. A higher level structure – the pattern – is defined as a list of motifs. A pattern also specifies the permitted ranges of spacing allowed between its constituent motifs. Equations for calculating the expected numbers of matches to patterns are given.

Introduction

One way of trying to interpret nucleic acid and protein sequences is to search them for the presence of subsequences that are of known function in other sequences. How these individual subsequences are defined will depend on the amount of information we have about them and also on the sophistication of the available software. We might also search for other features, that often have functional importance, such as direct or inverted repeats. In order to help evaluate the results of searches for any of these sequence motifs it is useful to know the *a priori* probability that they will be found, and methods to perform such calculations are presented here. However the main purpose of the calculations given here is to overcome a problem arising from some work described in a previous publication. The earlier paper (Staden, 1988) outlined methods of defining and searching for patterns of motifs in nucleic acid and protein sequences, and we summarize these techniques below.

A motif is defined as a contiguous section of a sequence. A pattern is a higher order of structure formed from motifs. The pattern specifies how the individual motifs are described and also their range of allowed spacings relative to one another. Nine different ways of defining motifs are given (see Figure 1), and each motif is included into the pattern using one of the logical operators AND, OR and NOT. The algorithm used to scan sequences for the occurrence of a pattern proceeds by looking for each motif in turn, only progressing to search for the next motif if the previous part of the pattern has already matched.

MRC Laboratory of Molecular Biology, Hills Road, Cambridge CB2 2QH, UK

It is, of course, important to be able to calculate the probabilities of finding matches to any particular motif. However, when we wish to combine motifs, defined in different ways, and consequently using different scoring methods and scales, such calculations are also useful because they allow us to combine their individual cut-off scores into a score for the whole pattern. That is, probability can be used to normalize the scores, and it is solving this normalization problem that was the major motivation for the work described here.

The general method used to calculate the probabilities for motif classes is that of probability-generating functions, and they are outlined below.

System and methods

The programs are written in FORTRAN 77 and we run them on a VAX 8600 under the VMS operating system. Great care has been taken to make it easy to port the programs to machines other than VAXes. All codes that deal with areas of the FORTRAN 77 language that are not precisely defined in the standard, such as logical unit numbers, file opening and other specific I/O statements, are separated into special subroutines. Two of the programs described produce both text and graphics output on the user's terminal screen. The graphics conform to the Tektronix 4010 standard. The FORTRAN routines that control the graphics are separate and designed for portability. We use several different types of terminal all of which are available throughout the world: the Hirez 100XL made by Selanar Corp., Sunnyvale, CA, USA; the MG200 made by Pericom PLC, Milton Keynes, UK; the QVT311GX made by Qume, Newbury, UK. We make increasing use of emulators for the IBM PC and Macintosh. For the IBM: EMUTEK, marketed by Boeing Computer Services (Europe) Ltd, Watford, UK, and for the Macintosh: Versaterm Pro from Abelbeck Software, which is distributed by PCS Inc., Mt Penn, PA, USA. The programs work, without change, on all these terminals and emulators and so would be expected to work on many others.

General outline of probability-generating functions

There are two characteristics of generating functions used below. First, if a random variable V can take the values 0,1,2,3...,T with probabilities $g_0,g_1g_2,g_3\ldots,g_T$ the probability-generating function for V is defined by the polynomial

$$G(x) = g_0 + g_1x + g_2x^2 + g_3x^3 + \ldots + g_Tx^T \quad (1)$$

The coefficient of x^N gives the probability that V has exactly the value N.

Secondly, if $V_1, V_2, V_3, \ldots V_m$ are independent random variables, the generating function for the probability distribution of $V_1 + V_2 + V_3 + \ldots + V_m$ is the product $G_1(x)G_2(x)G_3(x)\ldots G_m(x)$. That is

$$F(x) = \Pi G_j(x) \text{ for } j = 1,m \quad (2).$$

Application to the probabilities of finding motifs

Assumptions. All equations for motifs assume that the sequences are random and very long so that the probability of finding any base or amino acid is simply equal to its frequency in the particular sequence being analysed. All equations for patterns assume that the motifs in a pattern are independent and that the probability of finding a motif at position r is independent of the probability of finding it at $r - 1$.

Weight matrices. A weight matrix is a table that gives weights or scores for finding each of the residue types at each position along a motif. The values in the table represent the relative importance of each residue type at each position. In contrast to the score matrix described in the next section, a weight matrix therefore allows the same residue type to give different scores at different positions in a motif. When the matrix is applied to a sequence, a score for every segment of the sequence is calculated by adding or multiplying (see Discussion) the scores for the individual residue types found. Only those segments whose score reaches some predetermined cut-off value are reported as matches.

Below we show that we need a different generating function for each column of the weight matrix, and that the generating function for the whole matrix is the product of these individual functions. We define the following symbols.

Character set size	$= k$	
Sequence composition as frequencies	$= f_i$	$i = 1,k$
Weight matrix length	$= J$	
Weight matrix weights	$= w_{ij}$	$0 \leq w_{ij} \leq T$

The probability of getting any particular score for any column of the matrix is a function of the frequency of each character in the sequence and the number of different sequence characters that give each score.

Let the probability generating function for column j of the weight matrix be

$$G_j(x) = \Sigma f_i x^{w_{ij}} \quad (3)$$

From equation 2 we get

$$F(x) = \Pi \Sigma f_i x^{w_{ij}} \quad i = 1,k, j = 1,J \quad (4)$$

and the probability of getting exactly score S_N for the weight matrix is given by the coefficient of x^N in this equation.

CLASS	DESCRIPTION
1	Exact match to a short defined sequence. The IUB symbols can be used for DNA sequences.
2	Percentage match to a defined short sequence. In nucleic acids, the IUB symbols can be used.
3	Match to a defined sequence, using a score matrix and cutoff. The DNA matrix (see Figure 2) gives scores to IUB symbols depending on the amount of overlap between them. MDM78 (Dayhoff (1978)) is used for proteins.
4	Match to a weight matrix with cutoff score.
5	As class 4 but on the complementary strand.
6	Inverted repeat or stem-loop. Fixed stem length, range of loop sizes, and cutoff score using A-T, G-C=2; G-T=1.
7	Exact match to short sequence but with a defined step size.
8	Direct repeat. Fixed repeat length, range of gap sizes, cutoff score. For protein sequences scores are calculated using MDM78, but an identity matrix is used for nucleic acids.
9	Membership of a set. A list of sets of allowed amino acids for each position in the motif. The sets are separated by commas (,). For example IVL,,,DEKR,FYWILVM defines a motif of length 5 amino acids in which one of I,V or L must be found in the first position, then anything in the next two positions, D,E,K or R in the fourth position and F,Y,W,I,L,V or M in the fifth. This class only applies to protein sequences because for nucleic acids "membership of a set" can be achieved using IUB symbols.

Classes 1 - 4, 8 and 9 apply to protein sequences, and classes 1-8 to nucleic acids.

Fig. 1. Motif classes. The current ways of defining sequence motifs.

String and score matrix. Score matrices give scores for each possible character being aligned with each of the others. They are less specific than a weight matrix because each residue type gives the same score irrespective of its position in the motif. For this method of defining motifs a score matrix is used in conjunction with a string of symbols. The string is usually a consensus sequence representing the motif. A score for each segment of the sequence is obtained by adding the appropriate elements in the matrix. The most widely used score matrix for protein sequences is MDM78 (Schwartz and Dayhoff, 1978). A score matrix for the IUB symbols denoting nucleic acid sequence characters is shown in Figure 2. The values measure the level of overlap between each pair of symbols.

For this motif type we need the following symbols to be defined:

Character set size	$= k$	
Sequence composition as frequencies	$= f_i$	$i = 1,k$
String length	$= J$	
Position in string	$= j$	$1 \leq j \leq J$
Character index for position j	$= g$	$1 \leq g \leq k$
Score matrix weights	$= w_{ig}$	$0 \leq w_{ig} \leq T$

	T	C	A	G	-	R	Y	W	S	M	K	H	B	V	D	N	?
T	36	0	0	0	9	0	18	18	0	0	18	12	12	0	12	9	0
C	0	36	0	0	9	0	18	0	18	18	0	12	12	12	0	9	0
A	0	0	36	0	9	18	0	18	0	18	0	12	0	12	12	9	0
G	0	0	0	36	9	18	0	0	18	0	18	0	12	12	12	9	0
- ACGT	9	9	9	9	36	18	18	18	18	18	18	27	27	27	27	36	0
R AG	0	0	18	18	18	36	0	9	9	9	9	6	6	12	12	18	0
Y CT	18	18	0	0	18	0	36	9	9	9	9	12	12	6	6	18	0
W AT	18	0	18	0	18	9	9	36	0	9	9	12	6	6	12	18	0
S CG	0	18	0	18	18	9	9	0	36	9	9	6	12	12	6	18	0
M AC	0	18	18	0	18	9	9	9	9	36	0	12	6	12	6	18	0
K GT	18	0	0	18	18	9	9	9	9	0	36	6	12	6	12	18	0
H ACT	12	12	12	0	27	6	12	12	6	12	6	36	8	8	8	27	0
B CGT	12	12	0	12	27	6	12	6	12	6	12	8	36	8	8	27	0
V ACG	0	12	12	12	27	12	6	6	12	12	6	8	8	36	8	27	0
D AGT	12	0	12	12	27	12	6	12	6	6	12	8	8	8	36	27	0
N ACGT	9	9	9	9	36	18	18	18	18	18	18	27	27	27	27	36	0
? GAP	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fig. 2. A score matrix for nucleic acids. The scores represent the level of overlap between the IUB symbols.

	T	C	A	G	-	R	Y	W	S	M	K	H	B	V	D	N	?
T	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
G	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
- ACGT	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
R AG	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0
Y CT	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
W AT	1	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0
S CG	0	1	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0
M AC	0	1	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0
K GT	1	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0
H ACT	1	1	1	0	0	0	0	1	0	1	0	1	0	0	0	0	0
B CGT	1	1	0	1	0	0	1	0	1	0	1	0	1	0	0	0	0
V ACG	0	1	1	1	0	1	0	0	1	1	0	0	0	1	0	0	0
D AGT	1	0	1	1	0	1	0	1	0	0	1	0	0	0	1	0	0
N ACGT	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
? GAP	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fig. 3. A matrix of matching characters for nucleic acids, using IUB symbols.

	T	C	A	G	?
T	0	0	2	1	0
C	0	0	0	2	0
A	2	0	0	0	0
G	1	2	0	0	0
?	0	0	0	0	0

Fig. 4. A matrix for inverted repeats in nucleic acids.

For each position j within the string define the polynomial

$$G_j(x) = \Sigma f_i\, x^w ig \qquad (5)$$

So the probability generating function for the whole string is given by

$$F(x) = \prod \Sigma f_i\, x^w ig \qquad i=1,k \qquad j=1,J \qquad (6)$$

Strings and percentage matches. This method is an even simpler way of defining a motif in which we ask for a percentage match to a short string of symbols representing the consensus sequence of the motif. Typically we would ask for a minimum number of the symbols to match. Again we can use probability-generating functions. Suppose we have a matrix that represents a match between two characters with the value 1 and a mismatch by a 0. For example see Figure 3 that shows the matrix for IUB symbols in nucleic acid sequences. (This is derived from the matrix of Figure 2 but is not symmetrical because only definite matches are permitted.) This allows us to define a string containing any IUB symbols and to ask for a given proportion of the characters to match. By using exactly the same generating function as for the string and score matrix method but with the matrix in Figure 3 we can generate the probabilities of finding 0,1,2,3,4,...J matches with a string of J characters. For protein sequences an identity matrix is used.

Repeats in sequences. Here we choose a repeat length and a range of gap sizes between the two occurrences of the subsequence. For nucleic acids we use an identity matrix, and for proteins MDM78 to calculate the scores. This problem has already been solved by McLachlan (1971) in his 'double matching probability'. Here the probabilities for any position in a repeat depend on the frequencies of each of the pairs of aligned residues. So for each position j we have the generating function

$$G(x) = \Sigma f_j f_i\, x^w ij \qquad (7)$$

To calculate the overall probabilities we raise equation 7 to the power J and then multiply the result by the number of different gaps allowed.

Inverted repeats in nucleic acid sequences. This problem is equivalent to finding direct repeats but using a score matrix that gives a positive score to complementary characters and zero to others. Hence we use the equations for direct repeats with the matrix shown in Figure 4.

Exact matches to strings. These probabilities can be calculated using the equation for percentage matches and taking the value for a 100% match.

Membership of a set. This method of defining motifs leads to a much simpler probability calculation. The motif is defined by giving a list of the allowed characters at each position along its length. Clearly the probability at each position is simply the sum of the frequencies of the individual permitted characters. That is $p_j = \Sigma f_j$ and the probability for the whole motif is the product of these values for each position $\Pi\Sigma f_j$.

Algorithms

The 'membership of a set' calculation is trivial and will not be considered further here. For the other motif classes three algorithms are required: one for weight matrices, one for repeats and one for strings. In addition these three algorithms need to multiply polynomials and so we also show an algorithm for this purpose. Only the polynomial coefficients are multiplied—not the powers of x. Notice that it is the cumulative probability we need, i.e. the probability of getting at least score N, and so algorithms B, C and D return this.

Algorithm A: polynomial multiplication

We use three arrays to contain the polynomial coefficients: polya, polyb and polyc. The algorithm assumes that the coefficients for polynomial A are in array polya and those for polynomial B are in polyb. Elements 0 to a of array polya are multiplied with elements 0 to b of array polyb and placed in array polyc. Then the elements of polyc that are greater than some small value s are copied to polya; other elements of polya are set to zero. This is to avoid underflow.

Algorithm B: weight matrices

The weights are usually real values and so must be scaled to positive integers before performing the operations given below. The only new variables used are a, b and c which contain the highest element numbers used in arrays polya, polyb and polyc respectively. The value of b remains constant (=T) but a and c increase by T with each round of multiplication.

Algorithm C: repeats

We employ the same variables as for weight matrices and we use algorithm A to raise the polynomial to the power J.

Algorithm D: strings

Again this is similar to the weight matrix algorithm but here we use variable g to contain the index of the jth character in the string. For example, referring to Figures 2 or 3, the index for character A is 3 and for D is 15.

By reversing the scaling of the scores or weights the probability of getting at least any score can be found by looking up the appropriate element of array polya.

Worked example

Suppose we have the weight matrix shown in Figure 5, and we wish to apply it to sequence of composition f_i, where i = A,C,G,T.

There are only three scores possible for the first position in the weight matrix, namely 9, 1 and 0, with probabilities f_A, f_G and $f_C + f_T$ respectively. Referring to equation 1:

$$\begin{aligned} g_0 &= f_C + f_T \\ g_1 &= f_G \\ g_9 &= f_A \\ \text{else} &= 0 \end{aligned}$$

So in equation 3 (omitting terms with zero coefficients) we have

$$G_1(x) = (f_C + f_T)\,x^0 + f_G x^1 + f_A x^9$$

For colulmns 2 and 3 we get:

$$G_2(x) = (f_C + f_T)\,x^0 + f_A x^1 + f_G x^9$$

$$G_3(x) = (f_A + f_G + f_T)\,x^1 + f_C x^7$$

Similarly we substitute for the other columns and then evaluate equation 4 by muliplying together the five polynomials.

A	9	1	1	10	7
C	0	0	7	0	0
G	1	9	1	0	0
T	0	0	1	0	3

Fig. 5. An example of a weight matrix.

In order to make the method clear and to demonstrate that it gives the required results, let us restrict ourselves to the first two columns and compare the values we would expect to occur with those obtained from algorithm A. First, what is the probability of getting score 0 from these two columns? Inspecting the weight matrix shows we can get it from the sequences CC, CT, TT, and TC so the probability is

$$f_C f_C + f_C f_T + f_T f_C + f_T f_T$$

The other possible scores and their probabilities found from inspecting the matrix are:

$$\begin{aligned} \text{score } 1 &\quad f_C f_A + f_G f_C + f_G f_T + f_T f_A \\ \text{score } 2 &\quad f_G f_A \\ \text{score } 9 &\quad f_A f_C + f_A f_T + f_C f_G + f_T f_G \\ \text{score } 10 &\quad f_A f_A + f_G f_G \\ \text{score } 18 &\quad f_A f_G \end{aligned}$$

Now let us evaluate the first two columns of the table by stepping through the two main loops of algorithm A for values of i from 0 to 9 and values of j from 0 to 9. Assume we have put the coefficients for columns 1 and 2 in the appropriate elements of polya and polyb. Note that only values of i and j that have non-zero coefficients have been shown below.

i	j	$i+j$	polya(i)	polyb(j)	polyc($i+j$)	
0	0	0	$f_C + f_T$	$f_C + f_T$	$(f_C + f_T) * (f_C + f_T)$	S0
0	1	1	$f_C + f_T$	f_A	$(f_C + f_T) * f_A$	
0	9	9	$f_C + f_T$	f_G	$(f_C + f_T) * f_G$	
1	0	1	f_G	$f_C + f_T$	$(f_C + f_T) * f_A + f_G * (f_C + f_T)$	S1
1	1	2	f_G	f_A	$f_G f_A$	S2
1	9	10	f_G	f_G	$f_G f_G$	
9	0	9	f_A	$f_C + f_T$	$(f_C + f_T) * f_A + f_G * (f_C + f_T)$	S9
9	1	10	f_A	f_A	$f_G f_G + f_A f_A$	S10
9	9	18	f_A	f_G	$f_A f_G$	S18

The final sums for each possible score are in the rows marked with a letter S; all other sums are zero. Comparison with the expected values shows that the method and algorithm give the desired results.

Combining probabilities for motifs in a pattern

As is described in Staden (1988) patterns are lists of motifs. For each motif a range of allowed positions is defined relative to a motif further up the list. (The first motif in the list is an exception to this and its range is the length of the sequence being analysed). The chance of finding a match to a pattern is therefore a function of the probabilities and ranges of the individual

motifs. Note that a motif can only be found if all the motifs further up the list have already matched. Motifs are included into a pattern using one of the logical operators AND, OR and NOT. Below we examine the three operators in turn. Remember that we are assuming that the probability of finding a motif at position r in a sequence is independent of the probability of finding it at $r-1$ (see System and methods and Discussion).

First consider the AND operator. Suppose we have three motifs A, B and C with probabilities P_A, P_B and P_C, and ranges R_A, R_B and R_C. Then the expected number of matches to motif A is given by $P_A R_A$. For each of these matches the expected number of matches to B is $P_B R_B$. For each of these matches the expected number of matches to C is $P_C R_C$. That is

$$\begin{aligned} E_A &= P_A R_A \\ E_B &= E_A P_B R_B \\ E_C &= E_B P_C R_C = P_A R_A\ P_B R_B\ P_C R_C \\ &= P_A P_B P_C\ R_A R_B R_C \\ &= \text{Expected number of matches for the whole pattern} \end{aligned}$$

In general, if we have n motifs and P_m is the probability for motif m and R_m the range for motif m then the overall, or pattern, probability is

$$P = \Pi P_m \qquad m = 1,n \tag{8}$$

and the expected number of matches is

$$E = \Pi P_m\ R_m \qquad m = 1,n \tag{9}$$

In order to calculate the overall maximum probability for a pattern we need only multiply together the individual motif probabilities. To calculate the expected number of matches we multiply this value by the product of the ranges.

Were OR to be used probabilities should be added rather than multiplied. So for each set of motifs combined using the OR operator we have

$$E = \Sigma P_m\ R_m \qquad m = 1,n \tag{10}$$

If NOT is employed we must use $1 - P$ for the probability of such a condition being satisfied, and raise it to the power Range. That is

$$E = (1 - P_m)^{R_m} \tag{11}$$

In general patterns will comprise motifs included using all three operators so equations 8–11 are used as is appropriate.

Implementation

The above calculations have been employed in the following ways. As motifs are defined by the user of the programs a routine is called that returns the probability of finding the cut-off score. This value is multiplied into a running product, and the range is multiplied into a further value. Before a search is started the overall pattern probability is displayed and the user is given the opportunity to set a maximum overall probability for the pattern. In addition the program displays the expected number of matches calculated from equations 9, 10 and 11. If the expected number of matches is higher than desired the user may decide to set the maximum probability lower. If the user elects to obtain graphical output (only available for searches of individual sequences, not libraries), probabilities are always calculated to enable scaling of plots in the y direction.

T	6	49	1	56	6	22	6	20
C	14	6	0	0	3	0	1	2
A	8	4	58	4	51	38	53	30
G	32	1	1	0	0	0	0	8

Fig. 6. A weight matrix for TATAA boxes.

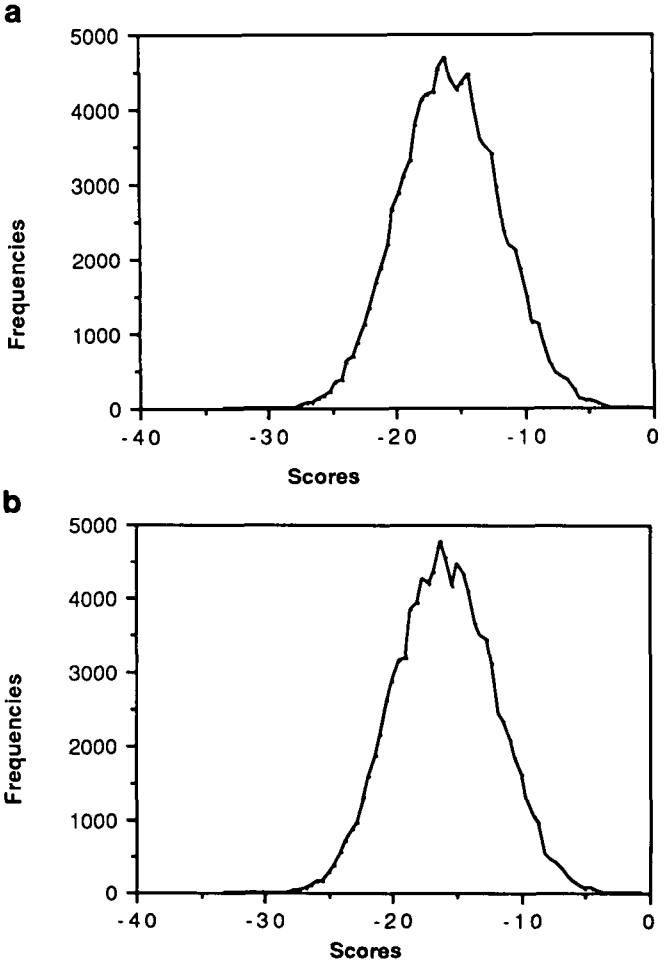

Fig. 7. a. A histogram of expected scores for the TATAA box weight matrix. b. A histogram of observed scores for the TATAA box weight matrix.

Comparison between observed and expected scores

The calculations and algorithms were tested in two ways: by comparing the shapes of expected and observed score histograms, and by comparing counts of observed and expected numbers of matches for randomly chosen motifs. A random

Table I. Comparison of expected and observed numbers of matches

Class	Motif	Score	Expected	Observed
1	AAAATTTT	8	14	6
1	ATATATAT	8	14	14
1	CGCG	4	84	88
1	GGCC	4	84	91
2	AAAATTTT	7	242	242
2	ATATATAT	7	242	240
3	TGACACGT	252	38	37
4	HEATSHOCK	−21.0	136	129
4	SP1	−7.1	1	1
4	TGGCA	−51.8	5	4
6	Length = 10 GAP = 10−15	16	735	725
8	Length = 10 GAP = 0−5	9	41	48

sequence of composition A = 33%, C = 17%, G = 17% and T = 33% and length = 100 000 characters was generated. Particular motifs of each class were randomly chosen and the equations given above used to calculate the number of matches we would expect to find in such a random sequence. Then the sequence was searched for the motifs and the observed and expected numbers of matches compared.

Below are shown graphs for a TATAA box (defined using the weight matrix show in Figure 6). The first graph (Figure 7a) shows the expected values for obtaining exactly each possible score, and the next (Figure 7b) the observed values found by searching the random sequence. Note the similarity of the shapes.

Table I shows some examples of comparisons between expected and observed values for several motif classes. These are for the same random sequence and are the counts for getting at least some particular score. Note that the weight matrices (not shown) were calculated from alignments of known examples of the heatshock (Pelham, 1985), SP1 (Kadonga *et al.*, 1986) and TGGCA (Nowock *et al.*, 1985) motifs. The class 6 example is for a stem of 10, score 16 and gap ranging from 10 to 15 bases; the class 8 example is for a repeat of length 10 with nine bases matching and gap varying from 0 to 5 bases. As can be seen there is good agreement between the observed and predicted numbers of matches.

Discussion

We have described the use of probability-generating functions for calculating the probabilities of finding motifs in sequences. Also equations have been given for calculating expected numbers of matches to patterns of motifs. The examples shown have all been for nucleic acids, but only the matrices change for proteins. The calculations can be performed quickly and the algorithms are simple. This should be compared with the alternative of randomizing sequences and repeated searching to find expected numbers of matches. Library searches have not been mentioned above, but as they are relatively slow, the current versions of the programs do not re-calculate probabilities for each individual sequence. Rather they assume even frequencies for nucleic acid sequences, and an average amino acid composition (Dayhoff *et al.*, 1978) for all the protein sequences. Should accurate values be required for individual sequences they can be calculated using the programs that operate on single sequences.

In the previous paper describing these pattern searching methods (Staden, 1988) we showed an example of searches of the human beta globin gene (Lawn *et al.*, 1980). The point of the example was to show that while searches for the individual components of the promoter (namely a CCAAT box, a TATA box and a capsite sequence) gave many false positive matches, a search for a pattern comprising all three together, gave only one match above a particular score (−15.0). The TATA box and capsite were defined using weight matrices and the CCAAT box as a five out of five match to the sequence CCAAT. To emphasise the point about the reduction in the background that can be achieved by searching for patterns rather than individual motifs, the cutoff score was chosen so that only one match would be found. Of course, in general, we will not get sensible results by using cutoff scores for patterns comprising motifs of different classes. But now, by employing the calculations presented here, we are able to give an overall cutoff, by defining it, not as a score, but as a probability. So repeating the search of the beta globin gene with the same pattern, but allowing a four out of five match to CCAAT and applying an overall probability cutoff of 0.7×10^{-6} we obtain the following results. The match shown in the previous paper is indeed the one with the lowest probability of being found on a random basis (0.16×10^{-9}), the matching positions and sequences being 28 CCAAT, 71 GCATAAAA, 99 TGCTTACATTTG; but we also find two other matches: 1070 CCAAA, 1100 TTAAAAAA, 1134 TGTTTATCTTAT (probability = 0.44×10^{-6}), and 1251 GCAAT, 1286 ATATAAAT, 1311 GTTTCATATTGC (probability = 0.6×10^{-6}).

Users of the programs should be aware of the assumptions made. It is assumed at several levels that the probabilities involved are independent. The probability of finding any residue is assumed to be independent of its neighbouring residues. In DNA sequences nearest neighbour frequencies are not random, but in proteins the assumption may be more realistic. Also if we are searching for an exact match to the motif AAAAAA, and we find a match at postion r we can still find it at postion $r+1$. But if we are searching for AAATTTT and find a match at r we cannot find a match at $r+1$. As these two assumptions are common to all motif classes, they do not detract from the major goal of allowing the application of overall cut-off scores to patterns of motifs defined using different methods. The problem of the lack of independence of 'hits' in dot-matrix sequence comparisons is discussed and given solutions by McLachlan and Boswell (1985) and Reich and Meiske (1987). A further level where the assumption of independence is not

always valid is when motifs are combined into a pattern, and their ranges overlap. To give an absurb example we might have two motifs, one an exact match to A and the other an exact match to T. We combine them using the AND operator and say they must both be in the same position. The probability calculation will give a value of $f_A f_T$ but clearly it should be zero.

It is worth pointing out that, since they were described in Staden (1984, 1985, 1988), several changes have occurred in the way that we define and use weight matrices. Originally weights were chosen by making an alignment from all known occurrences of the motif we wished to define. The weights were simply the logarithms of the observed base frequencies for each position in the alignment. To calculate cut-off scores to employ when using the matrix to search new sequences, we would first apply it to the aligned set of known motifs. This would give a range of expected scores, plus their mean and standard deviation, and cut-offs could be chosen accordingly. All of this (except the choice of score) was done automatically by the software. Now users can also choose their own frequency values, but still apply the resulting matrix to the original sequences, in order to select cut-off scores. Although only a trivial change, it allows users to apply other information about the relative importance of individual elements in the matrix and weigh them accordingly, yet still be able to estimate expected scores. The methods have been generalized to operate on protein sequences as well as nucleic acids. The way of correcting for zero elements in the frequency tables is now simply to add 1 to every element. A further addition is the facility to add the weight matrix values instead of multiplying them by summing their logs. The final addition is the provision of a masking facility, the application of which is explained below. Sequences often have superimposed functions and we may want to mask out those residue positions that are not essential to the structure we wish to define. Also some parts of a sequence may be of general structural importance and give rise to an overall framework, while other parts give specificity and so are not useful for defining a general structure. We may want to use a set of aligned sequences to define a motif, but want to use only the framework positions. The masking facility permits the user to specify which positions in a set of aligned sequences should be used when constructing a weight matrix. All of the operations to set up weight matrices ready for use by the search programs are handled by a program called WEIGHTS.

As stated in the Introduction, the purpose of the work presented here is to use probabilities to normalize scores for different ways of defining motifs, so that they can be combined together into scores for patterns. However, being able to calculate the probability of finding patterns in sequences also allows us to ask questions about their relative levels of specificity and the numbers of them we are likely to find in particular sequences. Above we described how cut-off scores for weight matrices are chosen by applying the matrix to known examples of the motif they represent. Suppose we use this method for defining regulatory sequences in nucleic acids. Then, for a genome of any given composition and length, we could use the equations given above, to estimate the numbers of sites we are likely to find that achieve scores at least as high as known functional sequences.

Questions concerning specificity have been addressed using information theory by Schneider *et al.* (1986). Also Berg and von Hippel (1987) have applied statistical mechanical theory to the problem of selection of DNA binding sites by regulatory proteins. One of their results shows how sequence information, as defined by Schneider *et al.* (1986) can be used to estimate the number of protein molecules that are bound to random sites within the genome. A useful survey of motif searching methods is contained in Stormo (1987).

Acknowledgements

I thank A.Klug, A.D.MacLachlan and P.J.G.Butler for critical reading of the manuscript.

References

Berg,O.G. and von Hippel,P.H. (1987) Selection of binding sites by regulatory proteins. *J. Mol. Biol.*, **193**, 723–750.

Dayhoff,M.O., Hunt,L.T .and Hurst-Calderone,S. (1978) Composition of proteins. In Dayhoff,M.O, (ed.), *Atlas of Protein Sequence and Structure*. National Biomedical Research Foundation, Georgetown University Medical Center, Washington, DC.

International Union of Biochemistry (1985) Nomenclature for incompletely specified bases in nucleic acids. *Eur. J. Biochem.*, **150**, 1–5.

Kadonga,J.T., Jones,K.A. and Tijan,R. (1986) Promoter-specific activation of RNA polymerase II transcription by Sp1. *Trends Biochem. Sci.*, **11**, 20–23.

Lawn,R.M., Efstradiatis,A., O'Connell,C. and Maniatis,T. (1980) The nucleotide sequence of the human beta-globin gene. *Cell*, **21**, 647–651.

McLachlan,A.D. (1971) Tests for comparing related amino-acid sequences. *J. Mol. Biol.*, **61**, 409–424.

McLachlan,A.D. and Boswell,D.R. (1985) Confidence limits for homology in protein or gene sequences. *J. Mol. Biol.*, **185**, 39–49.

Nowock,J., Borgmeyer,U., Puschel,A.W., Rupp,A.W. and Sippel,A.E. (1985) The TGGCA protein binds to the MMTV-LTR, the adenovirus origin of replication and the BK virus enhancer. *Nucleic Acids Res.*, **13**, 2045–2061.

Pelham,H. (1985) Activation of heat-shock genes in eukaryotes. *Trends Genet.*, **1**, 31–35.

Reich,J.G. and Meiske,W. (1987) A simple statistical significance test of window scores in large dot matrices obtained from protein or nucleic acid sequences. *Comput. Applic. Biosci.*, **3**, 25–30.

Schneider,T.D., Stormo,G.D., Gold,L. and Ehrenfeucht,A. (1986) Information content of binding sites on nucleotide sequences. *J. Mol. Biol.*, **188**, 415–431.

Schwartz,R.M. and Dayhoff,M.O. (1978) Matrices for detecting distant relationships. In Dayhoff,M.O. (ed.), *Atlas of Protein Sequences and Structure*. National Biomedical Research Foundation, Georgetown University Medical Center, Washington, DC.

Staden,R. (1984) Computer methods to locate signals in nucleic acid sequences. *Nucleic Acids Res.*, **12**, 521–538.

Staden,R. (1985) Computer methods to locate genes and signals in nucleic acid sequences. In Setlow,J. and Hollaender,A. (eds), *Genetic Engineering: Principles and Methods*. Plenum Publishing Corporation, Vol. 7, pp. 67–114.

Staden,R. (1988) Methods to define and locate patterns of motifs in sequences. *Comput. Applic. Biosci.*, **4**, 53–60.

Stormo,G.D. (1987) Identifying coding sequences. In Bishop,M.J. and Rawlings, C.J. (eds), *Nucleic Acid and Protien Sequence Analysis–A Practical Approach*. IRL Press, Oxford, pp. 231–258.

Received on March 1, 1988; accepted on September 30, 1988

Algorithm A

```
Zero array polyc
For i = 0 to a Do
    For j = 0 to b Do
        polyc(i+j) = polyc(i+j) + polya(i) * polyb(j)
    End do
For i = 0 to c
    z = polyc(i)
    If z < s then z = 0
    polya(i) = z
End do
```

Algorithm B

```
! Set-up and multiply together the polynomials:
Zero array polya.
For i = 1 to K Do
    polya (w_i1) = polya (w_i1) + f_i
End do
a = T
b = T
For j = 2 to J Do
    Zero array polyb
    For i = 1 to K Do
        polyb (w_ij) = polyb (w_ij) + f_i
    End do
    c = a + b
    polya = polya * polyb          ! use algorithm A
    a = c
End do
! Put the cumulative probability in polya:
For i = 1 to c Do
    j = c - i
    poly(j) = poly(j) + polya(j+1)
End do
```

Algorithm C

```
! set-up and multiply together the polynomials:
Zero array polya.
Zero array polyb.
For i = 1 to K Do
    For j = 1 to K Do
        polya (w_ij) = polya (w_ij) + f_i * h_j
        polyb (w_ij) = polya (w_ij)
    End do
End
a = T
b = T
For j = 2 to J Do
    c = a + b
    polya = polya * polyb          ! use algorithm A
    a = c
End do
! Put the cumulative probability in polya
For i = 1 to c Do
    j = c - i
    polya (j) = polya(j) + polya(j+1)
End do
```

Algorithm D

```
! set-up and multiply together the polynomials: Zero array polya.
g = character index for position 1 of string
For i = 1 to K Do
    polya (w_ig) = polya (w_ig) + f_i
End do
a = T
b = T
For j = 2 to J Do
    zero array polyb
    g = character index for position j of string
    For i = 1 to K Do
        polyb (w_ig) = polyb (w_ig) + f_i
    End do
    c = a + b
    polya = polya * polyb      ! use algorithm A
    a = c
End do
! Put the cumulative probability in polya
For i = 1 to c Do
    j = c - i
    polya (j) = polya(j) + polya (j+1)
End do
```