PATTERNS, PROFILES, AND MULTIPLE ALIGNMENTS

THEORY CHAPTER

When you have read Chapter 6, you should be able to:

Explain how to produce position-specific scoring matrices (PSSMs).

Describe the methods to overcome a lack of data.

Compare and contrast scoring methods for alignments and profiles.

Explain the graphical illustration of sequence profiles and patterns.

Explain how to produce profile HMMs.

Describe the alignment of profiles with sequences and other profiles.

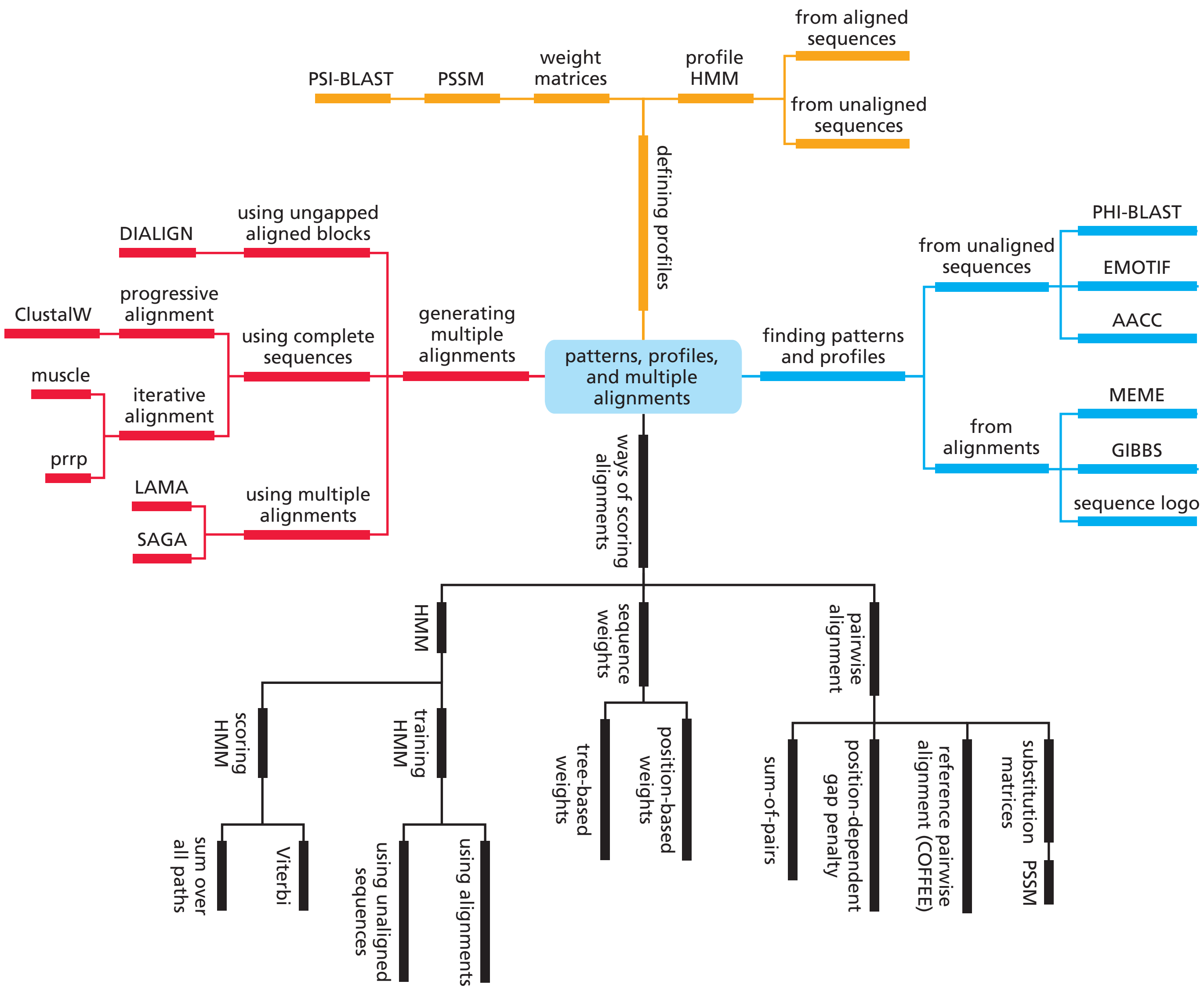
Obtain multiple alignments.

Discover sequence patterns.

Discuss the statistical scoring of patterns.

The great majority of protein sequences share significant similarity with others in the databases, as can be determined by the techniques described in Chapters 4 and 5. When these sequences are carefully compared they can reveal important information about the special role played by specific residues at particular sequence locations. The degree and kind of residue conservation found can improve our understanding of the interplay of protein function and structure. Furthermore, sequences that are all related by a common ancestral sequence hold the key to uncovering the evolutionary history since that ancestor. Techniques that are able to recover this information are presented in Chapters 7 and 8. All of these facets of protein sequence analysis are dependent at some stage on having a multiple alignment or profile constructed from all the available sequences. The main focus of this chapter is the description of the scientific bases of the many different techniques that have been proposed to achieve this crucial task.

In addition, many cases have been identified of short sequences, usually showing high levels of conservation, that have been found to correlate with specific functional properties, such as serine protease activity. These patterns are now used for function prediction purely on the basis of sequence. The last part of this chapter will explore the ways in which such patterns can be identified, including the use of methods that do not require alignments.



Mind Map 6.1
A mind map illustrating the topics covered in patterns, profiles, and multiple alignments. A large part is devoted to the all-important subject of scoring schemes.

Because of their general use in multiple alignment methods, in the first part of the chapter we describe methods for producing profiles, which at one level can be regarded as representations of alignments. Multiple alignments can be used in various ways to generate special scoring schemes for searching for other similar sequences. Position-specific scoring matrices (PSSMs), which take into account the position in the alignment when scoring matches, have been very successful in this context. Programs that use PSSMs include PSI-BLAST, for searching sequence databases, and LAMA, for searching a database of alignments with a query alignment.

We then discuss the use of hidden Markov models (HMMs), especially profile HMMs to define a sequence profile of a protein family. Sequence profiles can then be used to search for other family members. HMMs are based on a sound probabilistic theory, and are used in many multiple alignment and sequence-profile programs. Unlike PSSMs, HMMs do not require an alignment, and can produce a description of a family of related sequences without any prior alignment. Relationships between sequence families can be discovered by aligning profiles, which can be the most sensitive way to detect homology.

Most methods of multiple alignment are based on modifications of the pairwise dynamic programming techniques described in Section 5.2. In these methods the alignment is built up by adding sequences one at a time. As with pairwise alignments, the goodness-of-fit of the sequence to the alignment is tested by giving it a score, and the overall quality of the multiple alignment can also be tested quantitatively. The scoring schemes used have many similarities to those used for pairwise alignments but, as we shall see, additional scoring schemes are available, and different approaches can be taken to the practical task of building up the alignment.

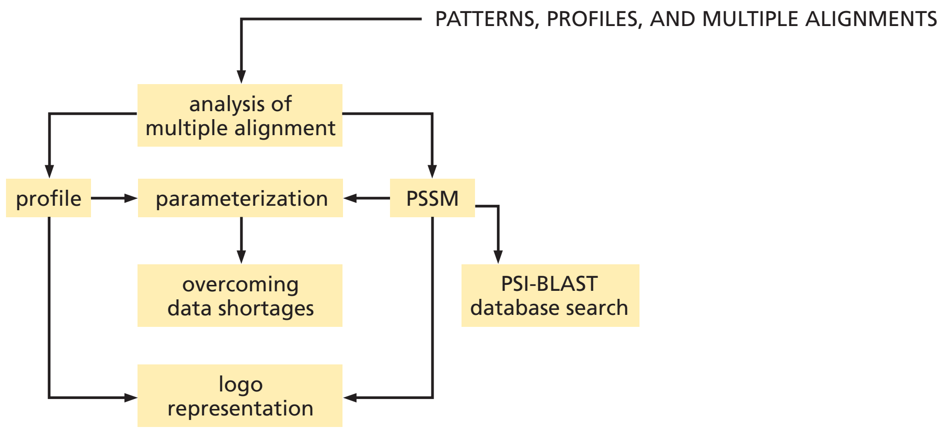
There are several other ways of constructing multiple alignments that differ in important respects from the pairwise dynamic programming approach. In a later part of the chapter we look at methods that construct alignments using all the sequences simultaneously.

The final part of the chapter deals with methods that can detect common sequence patterns in a multiple alignment, and those that can identify patterns in unaligned sequences. These have been used to create pattern databases such as PROSITE, which can be used to help predict a protein's function on the basis of the presence of specific sequences.

6.1 Profiles and Sequence Logos

In Chapter 5 we looked at some techniques for aligning one sequence to another. Very often many similar sequences are known, all presumably descendants of a common ancestral sequence. From experience these sequences will be expected to have similar properties at equivalent regions, leading for example to protein sequences sharing a common protein fold and patterns of residue conservation. A frequently encountered problem is the alignment of a sequence to such a set of similar sequences. Rather than align the new sequence to a single member of the set, it would be better to find a way to represent the general properties of the set of sequences such that the new sequence can be aligned to this representation. Such a representation is called a profile, and in this section we will consider the ways in which profiles can be constructed and used in pairwise alignment of new sequences (see Flow Diagram 6.1).

Profiles can be constructed in a form that makes it possible to use the dynamic programming techniques of Section 5.2 to align sequences to the profile. An important aspect of generating these profiles is the frequent lack of sufficient data for parameterization. A number of techniques are presented to overcome this problem. Following this we will briefly look at how the PSI-BLAST program uses these techniques. Finally, a method will be presented that gives a graphical display of the residue preferences of profile positions.


PATTERNS, PROFILES, AND MULTIPLE ALIGNMENTS
analysis of multiple alignment
profile
parameterization
PSSM
overcoming data shortages
PSI-BLAST database search
logo representation

Flow Diagram 6.1
The key concept introduced in this section is that analysis of multiple alignments can identify the specific preferences of each alignment position, which can be used to define a profile. The profile can be used to define a scoring scheme such as a position-specific scoring matrix (PSSM), which can be used to search for further examples of the profile, and can also be illustrated graphically as a logo.

Position-specific scoring matrices are an extension of substitution scoring matrices

All the alignment methods discussed in Chapter 5 apply a substitution score matrix to an alignment of residue type *a* with residue type *b* without regard for their environment; that is, the score $s_{a,b}$ for aligning these two residue types is always the same. Note that in Section 5.2 this was written $s(x_i,y_j)$ where x_i is the residue at the *i*th position of sequence $\boldsymbol{x}$. Similarly, the gap penalties discussed in Section 5.1 are the same regardless of where the gaps occur along the sequence. One can view database searches with these general scoring schemes as using the query sequence without any added information beyond the general features of evolution used to generate the substitution score matrix.

One of the common uses of database searches is to discover all known sequences that belong to the same sequence family as the query sequence, namely those sequences that align well over the whole of a specific region (often the entire length) of the query. Sequence variability within the family will usually prevent all members from being detected with a search based on a single query sequence. One method to try to identify potentially missed sequences is to perform several searches with different family members in the hope that all members will be identified by at least one of the searches. When the family examples used for further searches have been detected in the initial query-sequence search, and each newly discovered sequence is used in a further query, this method is referred to as an **iterated sequence search (ISS)**.

An alternative and usually more efficient way of finding all family members takes account of known residue preferences at each alignment position. This information is obtained from the alignment of an initial set of family members, such as those discovered from the first database search. Inclusion of these position-specific preferences in the scoring scheme is achieved with the use of a scoring profile in which each alignment position has its own substitution scores. Such constructions are often referred to as position-specific scoring matrices (PSSMs), although usually this terminology is reserved for cases without the position-dependent gap penalties that will also be described. The alignment of a sequence to a PSSM can proceed using dynamic programming in exactly the same way as described in Section 5.2 for aligning two sequences, except that the scores differ for each column. We will now discuss some of the ways of obtaining profiles, including methods that are designed to overcome the problems that may arise as a result of a lack of data.

In order to generate a PSSM, a set of sequences is required, all of which are aligned to a common reference. Two sources of such alignments exist: the results of a database search or a multiple alignment. In the first case, the common reference is the query sequence with which each database entry has been aligned. The common reference for multiple alignments is harder to define, but all the sequences are aligned with reference to all the other sequences. Either source can be used to define the preferences for each alignment position. The scores assigned must represent the residue preference found at a given position. If the alignment only contains a few sequences it will not be possible to determine an accurate residue preference at all alignment positions. This problem can be tackled in several related ways. A further aspect of determining PSSMs is that the data usually require weighting.

Suppose we have an alignment of N_{seq} sequences with L_{aln} positions; that is, L_{aln} alignment columns. The PSSM for this alignment will also have L_{aln} columns, each of which will (for protein sequences) have 20 rows. Each row corresponds to an amino acid type. An example of a PSSM can be seen in the top part of Figure 11.24, where the alignment is used to improve secondary structure prediction. However, note that in that example the PSSM has been transposed, so that the 20 rows have become 20 columns. It is possible to use one or two extra rows with parameters that relate to position-dependent gap penalties. Such matrices are usually referred to as profiles rather than PSSMs. (A PSSM, with only 20 rows, will use identical gap penalties at all sequence positions.) The values assigned to the PSSM are a weighted function of the values of a standard substitution matrix of the form discussed in Section 5.1, for example BLOSUM-45. We will write a substitution score matrix element $s_{a,b}$ for the alignment of residue types a and b, and will label the elements of the PSSM $m_{u,a}$ for column u and row (residue type) a.

One possible derivation of PSSM values uses the average of the scores of the residue types found at each alignment position. If a particular alignment column contains a perfectly conserved tyrosine residue, for example, the score on aligning a residue from another sequence to that position is taken to be the same as if we were dealing with just a single tyrosine residue. The elements of that column in the PSSM are the $s_{a,b}$ elements that relate to tyrosine; that is, for row (residue type) a they are $s_{a,\mathrm{Y}}$, using the one-letter amino acid code for the subscript. If instead the residue preference of that column was exactly shared by tyrosine and tryptophan, then each row a of the PSSM column will have the score $(s_{a,\mathrm{Y}} + s_{a,\mathrm{W}})/2$. Generalizing this, if there are $n_{u,b}$ residues of type b at column u, comprising a fraction $f_{u,b}$ of the column residues, i.e.,

$$f_{u,b} = \frac{n_{u,b}}{N_{seq}} \tag{EQ6.1}$$

then the score associated with row a and column u will be

$$m_{u,a} = \sum_{\substack{\text{residue}\\ \text{types } b}} f_{u,b} s_{a,b} \tag{EQ6.2}$$

If the residue type in an alignment column is found to be highly conserved, it seems sensible to give the preferred residues extra support, because the residue types rarely found are probably highly disfavored at that position. Rather than using the fraction $f_{u,b}$, the following logarithmic form of weighting the substitution scores has been proposed:

$$m_{u,a} = \sum_{\substack{\text{residue}\\ \text{types } b}} \frac{\ln\left(1 - f'_{u,b}\right)}{\ln\left(1/\left(N_{seq} + 1\right)\right)} s_{a,b} \tag{EQ6.3}$$

where $f'_{u,b}$ differs from $f_{u,b}$ as defined by Equation EQ6.1, in that the denominator is $(N_{seq} + 1)$ instead of N_{seq}. (This is necessary to avoid the numerator becoming $-\infty$ for alignment columns only containing a single residue type.) The value of the ratio of the logs varies between 0 and 1 as does $f_{u,b}$, but residues present in a smaller fraction of the sequences are relatively under-weighted.

Although the two methods just described have been applied successfully to create useful PSSMs, neither is of the log-odds ratio form that was shown in Section 5.1 to be particularly appropriate for alignment scoring. We will define the probability of residue type a occurring in column u of the PSSM as $q_{u,a}$, and the probability of residue type a occurring at any position in any sequence, including those not related to the PSSM sequence family (i.e., the background frequency), as p_a. The $q_{u,a}$ are the PSSM equivalent of the $q_{a,b}$ of Section 5.1, which are the probability of aligning two residues of types *a* and *b* when they are part of a meaningful alignment. The log-odds form for a PSSM element can then be written

$$m_{u,a} = \log \frac{q_{u,a}}{p_a} \qquad \text{(EQ6.4)}$$

If there are sufficient sequence data available, $q_{u,a}$ can be identified with $f_{u,a}$ as given by Equation EQ6.1. The p_a are readily obtained from the analysis of database composition.

Because PSSMs are often used in database searches for other members of the sequence family, it is important to understand the scoring statistics, as was the case for BLAST and FASTA searches as discussed in Section 5.4. As was explained then, each substitution matrix has a value λ associated with it that has a strong influence on the statistics. The following general formula applies to log-odds substitution matrices relating the alignment of two residues of types *a* and *b*:

$$q_{a,b} = p_a p_b e^{\lambda s_{a,b}} \qquad \text{(EQ6.5)}$$

(Compare this equation with Equation EQ5.31, which was encountered in the discussion of the significance of alignment scores.) Equation EQ6.5 can be rearranged to give

$$s_{a,b} = \frac{\ln\left(q_{a,b} / p_a p_b\right)}{\lambda} \qquad \text{(EQ6.6)}$$

By analogy, an alternative to Equation EQ6.4 is

$$m_{u,a} = \frac{\log\left(q_{u,a} / p_a\right)}{\lambda} \qquad \text{(EQ6.7)}$$

where the value of λ can be specified to control the scoring statistics. It is a simple scaling factor for the scores, and so could be omitted. However, this method of obtaining PSSM values has been applied in the PSI-BLAST method described below, where the λ parameter is used to selectively scale the scores as desired.

If position-dependent gap penalties are included, these can be assigned manually on the basis of the location of secondary structural elements to give smaller penalties for creating alignment gaps between these elements. An alternative approach is to have a position-specific multiplier of the gap penalty at column *u*, g'_u, such that an affine gap penalty for a gap of length n_{gap} extending across this column has a penalty

$$g_u\left(n_{\text{gap}}\right) = g'_u\left[-I - \left(n_{\text{gap}} - 1\right)E\right] \qquad \text{(EQ6.8)}$$

where *I* and *E* are the gap opening and extension penalties, respectively (see Equation EQ5.14). One proposed assignment of values to g'_u starts by identifying the length of the longest gap that occurs that includes column *u* and the highest possible score in the substitution matrix used. These values are used to scale g'_u so that the penalty applied to the longest observed gap is of the same magnitude as the highest possible score in the substitution matrix.

Thus far, we have treated each sequence in the alignment equally. The best PSSM will represent the full range of diversity within the sequence family in an unbiased fashion. However, a partial set of family sequences will most probably be biased toward a certain subgroup. Hence, we must weight the different sequences, and the weighting should be reduced for very similar sequences. We will look at two sequence weighting schemes, one of which applies the weight to all residues of a sequence, the other specifying different weights for each alignment column.

Some PSSMs have been derived using a sequence weighting scheme proposed by Peter Sibbald and Patrick Argos. In this scheme, weights are assigned to the sequences on the basis of an iterative procedure. The sequence weights are first initialized to zero. Random aligned sequences are generated where the residue at each position is chosen at random from the residues (including any gap) that occur at that particular position in the aligned sequences. The closest sequence or equally close sequences to each random sequence are identified, and a weight of 1 is evenly distributed between them. The sequence weights are normalized to sum to 1. Further random sequences are generated until the sequence weights are seen to have converged.

The position-based sequence-weight method assigns weights based on a multiple alignment. The basic unit used to assign weights is not the whole sequence, but the individual alignment columns. For each column, the number of different residues present is counted. If there are m different residues, each is assigned a weight of $1/m$. Then for each residue type, the number of sequences that have this residue at this position is counted. If there are n sequences with this residue, the weight is equally divided amongst them; that is, the weight becomes $1/mn$. These weights can be used as sequence weights for the individual column, and the total for all aligned residues is always 1. If desired, an overall weight for the whole sequence can be defined by adding the individual column weights and then normalizing by the number of columns, so that the sequence weights also sum to 1. This weighting scheme is illustrated in Figure 6.1 for a simple example alignment.

12345	Weight
HSAPL	0.280
HTADV	0.171
HTAEV	0.171
HTGLI	0.188
HTGVI	0.188

Figure 6.1
Illustration of the position-based sequence weight scheme of Henikoff and Henikoff. There are five sequences, each of them is five residues long. Column 1 contains only one residue type (H) but there are five occurrences, so each histidine has a weight of ⅕. Column 4 contains five different residue types, each of which will be weighted as ⅕. Note that this scheme does not distinguish between these two situations, which can be understood by observing that neither can be used to prefer any sequence over any other. In column 2, there are two residue types (S and T), each of which will be assigned an initial weight of ½. As there is only one instance of S, it is assigned a weight of ½. The four Ts will each have a weight of ⅛. After assigning weights to each residue, the sequence weights are obtained by adding the residue weights together and dividing by the number of columns. The resulting sequence weights are given in the right-hand column.

If the weights are labeled $w_u^{\boldsymbol{x}}$ for column u of sequence $\boldsymbol{x}$ (including the possibility that there may be no variation across the alignment columns) the above formulae can be modified by replacing $f_{u,b}$ with

$$f''_{u,b} = \frac{\sum_{\text{sequences } x} w_u^x \delta_{u,b}^x}{\sum_{\text{sequences } x} w_u^x} \qquad \text{(EQ6.9)}$$

where $\delta_{u,b}^{\boldsymbol{x}}$ is 1 if sequence $\boldsymbol{x}$ has residue b in alignment column u, and 0 otherwise. Note that both weighting methods discussed above do not require the denominator sum of Equation EQ6.9, as they both produce weights that sum to 1 for a column.

Methods for overcoming a lack of data in deriving the values for a PSSM

The log-odds schemes for calculating the PSSM elements $m_{u,a}$ using the formulae given above all have the property that if any residue type is not observed in the column u, the score for aligning that residue type in that column will always have the value $-\infty$. As a consequence, no sequence being aligned to the PSSM will be able to align that residue type at that location. While this might be appropriate for perfectly conserved and functionally vital positions, even in extreme cases it is almost certainly too restrictive. (We most probably want to be able to use the PSSM to align any sequences that code for related but dysfunctional proteins.) Furthermore, the absence of a particular residue type in a particular column of the alignment used to derive the PSSM is more likely to indicate a lack of sequence alignment data rather than the true residue preferences. Such situations are common in fitting profile models, and a variety of possible solutions have been proposed that will now be described.

The p_a of Equations EQ6.4 and EQ6.7 does not cause the $-\infty$ values mentioned above, nor does their estimation suffer from a lack of data. The problem rests entirely with the estimation of the $q_{u,a}$ and is almost always due to a lack of data. (The exception to this would be a rare case of a truly perfectly conserved residue type—an extremely rare occurrence.) All the methods described to overcome the problem can include sequence weights but they make the formulae more complex, so for simplicity we will not use them here. The starting point for estimating $q_{u,a}$ is to use $f_{u,a}$ as given by Equation EQ6.1.

A simple way of trying to overcome the lack of data would be to assume at least one occurrence of each residue type at each alignment position. It is preferable to treat all residues and all columns the same way, so in this case Equation EQ6.1 is modified to

$$q_{u,a} = \frac{n_{u,a}+1}{N_{seq}+20} \qquad \text{(EQ6.10)}$$

in the case of a PSSM with 20 rows (i.e., no position-dependent gap parameters). Note that the denominator becomes ($N_{seq}+20$) as this is the sum of the numerators for the 20 different terms. The sum of the $q_{u,a}$ for all possible a must always equal 1 as each sequence must be represented at column u. The inclusion of the extra observation means that $q_{u,a}$ will never be 0 nor ever reach 1. It is as if we have increased the amount of data available by 20 residues in each column, and such additional data are usually called **pseudocounts.**

It is easy to see that there are more sophisticated ways of adding pseudocounts that take advantage of the knowledge we have of the properties of sequences. We know that the amino acid composition of proteins is not uniform and, as discussed in Section 5.1, the frequency of occurrence p_a of residue type a. This knowledge can readily be incorporated into the formula to obtain

$$q_{u,a} = \frac{n_{u,a}+\beta p_a}{N_{seq}+\beta} \qquad \text{(EQ6.11)}$$

where the parameter β is a simple scaling parameter that determines the total number of pseudocounts in an alignment column. The advantage of introducing the β parameter is that we can easily adjust the relative weighting of the pseudocounts and real data. When there are a lot of data (i.e., N_{seq} is large) there is little if any need for pseudocounts, and β should be much smaller than N_{seq}, whereas when there are less data, β should be larger relative to N_{seq}. A simple formula that has been found useful is to make β equal to $\sqrt{N_{seq}}$, although this can result in β being too small for small values of N_{seq}. At large values of N_{seq} this formula approaches $q_{u,a} = f_{u,a}$ as desired. Often an additional parameter α is used to weight the observed data, giving a formula more easily expressed in terms of the $f_{u,a}$ than the $n_{u,a}$:

$$q_{u,a} = \frac{\alpha f_{u,a}+\beta p_a}{\alpha+\beta} \qquad \text{(EQ6.12)}$$

In the absence of any data, the pseudocounts would completely determine the PSSM values. In Bayesian analysis terms, the pseudocounts represent the **prior distribution**, which expresses our prior knowledge of the system before we introduce the data. (They can also be seen as an expression of our prejudices and bias!) See Appendix A for a discussion of Bayesian analysis.

We can improve on the pseudocount distribution suggested by Equations EQ6.11 and EQ6.12, because the substitution matrices contain a more accurate distribution based on the data from which they were derived. By definition, log-odds substitution matrices have embedded in them the information about the relative probabilities of residues aligning because their sequences are related as opposed to purely random alignment. This is expressed by the terms $q_{a,b}/p_a p_b$ for residue types a and b. Rearranging Equation EQ6.5 we find

$$\frac{q_{a,b}}{p_a p_b} = e^{\lambda s_{a,b}} \qquad \text{(EQ6.13)}$$

If a column u contains a fraction $f_{u,b}$ of type b residues, the probability of finding a residue of type a aligned with these is proportional to $f_{u,b} q_{a,b}/p_a p_b$. Adding these terms for all residue types b gives the overall probability of finding a residue of type a aligned at column u on the basis of existing residues in that column. Because residue type a occurs at a background frequency p_a, this sum needs to be multiplied by p_a to obtain a suitable pseudocount for residue type a. Thus the formula for the number of pseudocounts of residue type a is

$$g_{u,a} = p_a \sum_{\substack{\text{residue} \\ \text{types } b}} \frac{f_{u,b} q_{a,b}}{p_a p_b} = \sum_{\substack{\text{residue} \\ \text{types } b}} \frac{f_{u,b}}{p_b} q_{a,b} \qquad \text{(EQ6.14)}$$

Note that because of the summation, the $g_{u,a}$ can never be 0. If the intermediate results are available that were used to derive the substitution matrix, the values of the terms $q_{a,b}/p_a p_b$ may be available. If this is not the case, they can be recovered from the substitution scores $s_{a,b}$ by applying Equation EQ6.13, although the data are likely to be less accurate. Replacing p_a with this in Equation EQ6.12 we obtain the improved estimate of $q_{u,a}$:

$$q_{u,a} = \frac{\alpha f_{u,a} + \beta g_{u,a}}{\alpha + \beta} \qquad \text{(EQ6.15)}$$

One possible value for α is $N_{seq} - 1$. If the PSSM elements $m_{u,a}$ are calculated using Equation EQ6.7, then with only one sequence this particular α will result in the $m_{u,a}$ being the substitution matrix values $s_{a,b}$ (see Equation EQ6.13). The value of β controls how much the PSSM is biased to the substitution matrix when there are few data available. In PSI-BLAST (see below) a value of 10 is used by default.

The derivation of pseudocounts from substitution scoring matrices goes some way toward using a more realistic prior distribution. However, that method as well as all the others described so far have qualitatively inaccurate features. Only the relative frequency of different amino acids in an alignment column affects the PSSM values. All PSSM columns based on the observation of a single fully conserved amino acid will have the same values regardless of whether the alignment contained just three or a thousand sequences. Clearly the latter case, with no exceptions in a thousand observations, should be treated differently, with far greater penalties for the presence of a different residue than when the PSSM has been derived from only three sequences. The second incorrect feature is that no distinction is made between the different possible environments of a residue. The probabilities associated in these methods with any two aligned residue types are the same in all circumstances. However, it is well known that alignment columns often show a clear preference for a small set of residues with similar properties, such as charge, size, or polarity. In reality the alignment probabilities will differ for a given pair of residue types according to these preferences. The probability of aligning a leucine to a column will differ according to whether that column has a perfectly conserved leucine, conserved residues that are small and hydrophobic, or no apparent residue preference.

Figure 6.2
The nine-component Dirichlet mixture derived from the BLOCKS alignment database. The components are labeled Blocks9.1 to Blocks9.9, and their mixture coefficients c_i are given in the first row. Subsequent rows give the distributions (unnormalized) for the individual amino acids in each component. Those amino acids of a component that are present at more than double the background frequency are highlighted in yellow, and summarized in the last row. Thus positively and negatively charged residues dominate components Blocks9.4 and Blocks9.7, respectively. The overall amino acid composition predicted by this mixture is obtained by weighting these component distributions by their coefficients c_i. (Data K. Sjölander, K. Karplus, M. Brown et al. Dirichlet mixtures: a method for improved detection of weak but significant protein sequence homology. *Comput. Appl. Biosci.* 12:327–345, 1996.)

It is possible to avoid the shortcomings of the methods described so far by using a **Dirichlet mixture**, more accurately described as a mixture of **Dirichlet distribution densities**. Dirichlet distributions are often used as the prior distribution in Bayesian analysis of data where each observation is one of a limited set of possibilities. In this case, the 20 amino acids constitute the set, and each Dirichlet density describes a particular residue composition that might occur at an alignment column. The Dirichlet mixture is a linear combination of these densities with multiplying coefficients that are the probability of their occurrence. Therefore these coefficients sum to 1. A set of densities and mixture coefficients has been fitted by Kimmen Sjölander and co-workers to the BLOCKS database of protein multiple alignments. The details of the fitting are beyond the scope of this book, but involve maximum likelihood and the EM (expectation maximization) method. For further information on these methods see Further Reading.

Figure 6.2 gives the values for a nine-component Dirichlet mixture derived from the BLOCKS database. The *i*th Dirichlet density component is labeled Blocks9.*i* and has a linear mixture coefficient (probability of occurrence in the BLOCKS data) c_i. The amino acid distribution of each density component is defined by the numbers in each column, labeled $\beta_{i,a}$ for residue type a. Note that these do not add to 1, their sum β_i being given for each density, so the frequency for residue a in density i is given by $\beta_{i,a}/\beta_i$. The density components can be analyzed to identify their residue preferences. As the BLOCKS database like all real protein sequences has a nonuniform residue composition—equivalent to the background composition p_a of the previous formulae—the component residue preferences are best measured as the component frequency relative to the background frequency. These preferences are shown on the bottom row of Figure 6.2, where all residues present at greater than twice the background frequency are listed. For example, the first component favors

	Blocks9.1	Blocks9.2	Blocks9.3	Blocks9.4	Blocks9.5	Blocks9.6	Blocks9.7	Blocks9.8	Blocks9.9
c_i	0.182962	0.057607	0.089823	0.079297	0.083183	0.091122	0.115962	0.066040	0.234006
A	0.270671	0.021465	0.561459	0.070143	0.041103	0.115607	0.093461	0.452171	0.005193
C	0.039848	0.010300	0.045448	0.011140	0.014794	0.037381	0.004737	0.114613	0.004039
D	0.017576	0.011741	0.438366	0.019479	0.005610	0.012414	0.387252	0.062460	0.006722
E	0.016415	0.010883	0.764167	0.094657	0.010216	0.018179	0.347841	0.115702	0.006121
F	0.014268	0.385651	0.087364	0.013162	0.153602	0.051778	0.010822	0.284246	0.003468
G	0.131916	0.016416	0.259114	0.048038	0.007797	0.017255	0.105877	0.140204	0.016931
H	0.012391	0.076196	0.214940	0.077000	0.007175	0.004911	0.049776	0.100358	0.003647
I	0.022599	0.035329	0.145928	0.032939	0.299635	0.796882	0.014963	0.550230	0.002184
K	0.020358	0.013921	0.762204	0.576639	0.010849	0.017074	0.094276	0.143995	0.005019
L	0.030727	0.093517	0.247320	0.072293	0.999446	0.285858	0.027761	0.700649	0.005990
M	0.015315	0.022034	0.118662	0.028240	0.210189	0.075811	0.010040	0.276580	0.001473
N	0.048298	0.028593	0.441564	0.080372	0.006127	0.014548	0.187869	0.118569	0.004158
P	0.053803	0.013086	0.174822	0.037661	0.013021	0.015092	0.050018	0.097470	0.009055
Q	0.020662	0.023011	0.530840	0.185037	0.019798	0.011382	0.110039	0.126673	0.003630
R	0.023612	0.018866	0.465529	0.506783	0.014509	0.012696	0.038668	0.143634	0.006583
S	0.216147	0.029156	0.583402	0.073732	0.012049	0.027535	0.119471	0.278983	0.003172
T	0.147226	0.018153	0.445586	0.071587	0.035799	0.088333	0.065802	0.358482	0.003690
V	0.065438	0.036100	0.227050	0.042532	0.180085	0.944340	0.025430	0.661750	0.002967
W	0.003758	0.071770	0.029510	0.011254	0.012744	0.004373	0.003215	0.061533	0.002772
Y	0.009621	0.419641	0.121090	0.028723	0.026466	0.016741	0.018742	0.199373	0.002686
β_i	1.180650	1.355830	6.664360	2.081410	2.081010	2.568190	1.766060	4.987680	0.099500
	AST	FHWY	EQ	KQR	ILM	IV	DEN	M	GPW

small neutral residues; the second component favors aromatic residues. The ninth component has the largest mixture coefficient, and is the most common component in the BLOCKS database. It is the component with the closest composition to the background frequency, and favors highly conserved individual residues such as tryptophan and proline. Other Dirichlet mixtures have been derived to represent amino acid alignments, including one with twenty components that is regarded as an improvement over the Blocks9 mixture.

The application of Dirichlet mixtures is more complex than using pseudocounts, because for each alignment column the weights have to be assigned to each of the components. The equations for residue type *a* in alignment column *u* is given by

$$q_{u,a} = \sum_{\substack{\text{Dirichlet}\\ \text{components } i}} P\left(\vec{\beta}_i \middle| \vec{n}_u, \theta\right) \frac{n_{u,a} + \beta_{i,a}}{N_{seq} + \beta_i} \qquad \text{(EQ6.16)}$$

where the term $P\left(\vec{\beta}_i \middle| \vec{n}_u, \theta\right)$ means the posterior probability (see Appendix A for a definition of this term) of the *i*th Dirichlet component with a vector of residue coefficients $\vec{\beta}_i$, given the vector of observed residue occurrences $\vec{n}_u$ in alignment column *u*. The symbol θ stands for all the Dirichlet mixture parameters. This term is the weight of the *i*th Dirichlet component. This equation should be compared with Equation EQ6.11.

After considerable algebraic manipulation, it is possible to derive a useful expression for terms $q'_{u,a}$, which can then be normalized to sum to 1 to obtain the $q_{u,a}$. The formula is

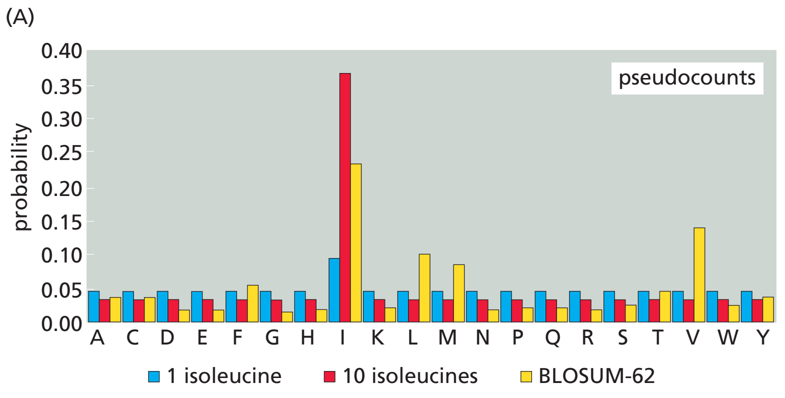
$$q'_{u,a} = \sum_{\substack{\text{Dirichlet}\\ \text{components } i}} c_i \frac{B\left(\vec{n}_u + \vec{\beta}_i\right)}{B\left(\vec{\beta}_i\right)} \left(\frac{n_{u,a} + \beta_{i,a}}{N_{seq} + \beta_i}\right) \qquad \text{(EQ6.17)}$$

where the mixture coefficients c_i are now clearly involved, and the function *B* is the Beta function, which is defined in terms of the more familiar mathematical Gamma function (Γ) by

$$B\left(\vec{n}_u + \vec{\beta}_i\right) = \frac{\prod_{\substack{\text{residue}\\ \text{type } b}} \Gamma\left(n_{u,b} + \beta_{i,b}\right)}{\Gamma\left(N_{seq} + \beta_i\right)} \qquad \text{(EQ6.18)}$$

Thus the observed residue counts in a column can readily be combined with the coefficients of the Dirichlet mixture components to obtain an improved estimate of the $q_{u,a}$.

Figure 6.3 compares some of the alternative methods discussed to overcome a lack of data, using as an example two cases. In one case, only a single sequence is present in the data, with a leucine residue at the column of interest. In the other case, 10 sequences all have a leucine at that position. The column composition is identical in the two cases, but clearly there is more information when the data consist of 10 sequences instead of just one. The figure shows how different methods can incorporate that information, with the Dirichlet mixture method clearly superior to the others.



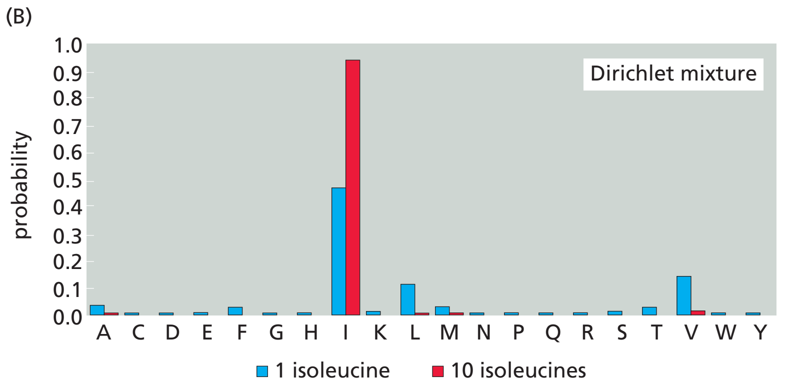


Figure 6.3
The estimated probabilities for amino acids according to some methods for dealing with a lack of data. The results shown are those given when the data indicate a conserved site with only isoleucine residues. Two cases are considered, observing either 1 or 10 isoleucine residues at the site. (A) The estimates given by using the BLOSUM-62 matrix with Equation EQ6.3 compared with those obtained using a pseudocount method of adding 1 to each observed amino acid frequency (Equation EQ6.10). Note that Equation EQ6.3 is only sensitive to observed composition, so that the probabilities are the same. (B) The estimates given by the nine-component Dirichlet mixture of Sjölander et al. (see Figure 6.2).

PSI-BLAST is a sequence database searching program

When a PSSM is carefully constructed it can be a very sensitive tool in database searches, finding many distant members of a protein sequence family not easily found by a standard sequence search. For this reason Steven Altschul and co-workers decided to enhance their BLAST database searching method to incorporate PSSMs. They created the program PSI-BLAST (Position-Specific Iterated BLAST) which has proved extremely successful.

A PSI-BLAST database search incorporates a number of steps. The first step usually involves performing a standard BLAST search of a database using the BLOSUM-62 substitution matrix with a single query sequence. This results in an initial set of related sequences. These sequences will be determined as those whose BLAST score gives an E-value smaller than a predetermined cut-off, often quite stringent to ensure that only truly significant matches are identified (see Section 5.4). The alignments of these significant matches with the query sequence are used to create a PSSM. The PSSM is then scanned against the database using a variant of the BLAST program to identify new sequences with suitably small E-values. If this second search finds some newly identified related sequences, these are used to update the PSSM. Successive cycles of PSSM refinement and database searching can be carried out until no new sequences are found.

Clearly, the cut-off threshold of the E-value is a critical parameter. If it is too large, as well as homologs, false positive sequences (i.e., unrelated to the initial query sequence) will be collected. These will result in PSSM corruption, which in further cycles will most likely give rise to a substantial fraction of false positive sequences. If the cut-off is too small, the run may fail to identify some of the more distantly related sequences, or will require many search cycles to locate them. A commonly used E-value cut-off is 0.001, although this may still result in a corrupted PSSM in some cases.

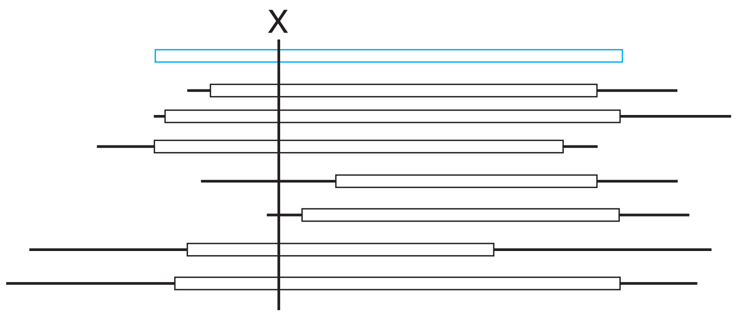


Figure 6.4
Illustration of how PSI-BLAST uses the pairwise sequence alignments from a database search in PSSM construction. The BLAST local alignments to the query sequence (the top blue sequence) are shown as rectangles. At each residue position of the query, a PSSM is constructed using only those sequences whose BLAST alignments involve that position. Thus at position X only six residues are considered in order to derive the PSSM, as only six of the sequences (including the query sequence) have local alignments including this position.

We will concentrate here on how the PSSM is constructed in PSI-BLAST. The PSSM is restricted to those residues that have been aligned to a residue in the query sequence. This removes from consideration any residues of database sequences that align with insertions in the query sequence. Thus the PSSM will have the same length as the query sequence. The PSSM constructed does not explicitly consider gaps, the usual gap penalties being used even when searching with the PSSM. Thus the gaps are treated exactly as in BLAST, i.e., not position-specific.

For any given alignment column, only those sequences that actually have a residue aligned are considered, so that the number of sequences in the alignment changes from column to column (see Figure 6.4). Each sequence is weighted using the position-based sequence weight scheme (see Figure 6.1), slightly modified to include gaps as another residue type, and to ignore fully conserved residues. The resulting weighted frequencies are used in Equations EQ6.14 and EQ6.15, which are then substituted into Equation EQ6.7 to obtain the PSSM parameters. The value of α used is not $N_{seq} - 1$, where N_{seq} is the number of sequences, but $N' - 1$, where N' is the number of different residue types observed in the column, including gaps, and thus varies from 1 to 21. The initial version of PSI-BLAST used standard substitution score matrices and Equation EQ6.13, but recent versions use the values of the terms $q_{a,b}/p_a p_b$.

The scaling λ used is that given by the chosen substitution matrix. The gap penalties used are those applied in a standard BLAST run in combination with this substitution matrix. This results in the scoring statistics for the PSSM being the same as for standard BLAST with the same matrix. This was confirmed using alignment data for real sequences, as the theory has not been derived for these alignments. Consequently, the measure of significance for PSI-BLAST scores is readily available, and one can readily ascertain which new sequences should be included in the recalculation of the PSSM. The technique has proved very successful, as shown by a comparison with some other sequence-search methods (see Table 6.1). A further refinement of λ, resulting in improved accuracy, has been applied in recent versions of PSI-BLAST. The value is modified to take account of deviations from the scoring statistics that result from compositional bias in the sequences.

Representing a profile as a logo

The score parameters of a PSSM are useful for obtaining alignments, but do not easily show the residue preferences or conservation at particular positions. This residue information is of interest because it is suggestive of the key functional sites of the protein family. A suitable graphical representation would make the identification of these key residues easier. One solution to this problem uses information theory, and produces diagrams that are called **logos**.

In any PSSM column u residue type a will occur with a frequency $f_{u,a}$. The entropy (uncertainty—see Appendix A) in that position is defined by

$$H_u = -\sum_a f_{u,a} \log_2 f_{u,a} \tag{EQ6.19}$$

Protein family	SWISS-PROT id	Smith–Waterman	Gapped BLAST	PSI-BLAST
Interferon α	P05013	53	53	53
Serine protease	P00762	275	275	286
Serine protease inhibitor	P01008	108	108	111
Glutathione transferase	P14942	83	81	142
Ras	P01111	255	252	375
Globin	P02232	28	28	623

A protein sequence from each of the example families (designated by a protein i.d. in the SWISS-PROT sequence database) was submitted to a search using one of the three methods. For each search algorithm the number of sequences found for that family is reported. The Smith–Waterman method uses full matrix dynamic programming as described in Section 5.2 to determine the local alignment of optimal score. Gapped BLAST uses a method described in Section 5.3 that is faster but in principle cannot guarantee to find the optimal alignment. In all cases except the Ras family it is as successful as Smith–Waterman. The PSI-BLAST method is notably more successful than the other two, especially in three cases shown here, indicating that it is considerably more sensitive in homology searches. In the first step a gapped BLAST search is run, so it always reports at least as many similar sequences. Only in the case of interferon α does it fail to find any more sequences. (Data from Table 3 of Altschul *et al.*, 1997.)

Table 6.1
An illustration of the greater effectiveness of the PSI-BLAST method as compared with some other algorithms for detecting significantly similar sequences.

where the summation is over all the residue types, and the entropy units are bits of information. If only one residue is found at that position, all terms are zero and H_u is zero; that is, there is no uncertainty. The maximum value of H_u occurs if all residues are present with equal frequency, in which case H_u takes the value $\log_2 20$ for proteins. The information present in the pattern at that position I_u is given by

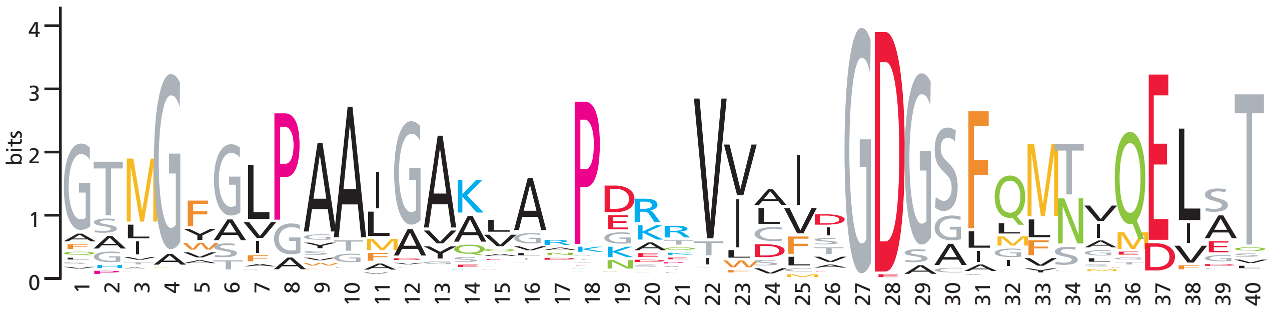
$$I_u = \log_2 20 - H_u \qquad \text{(EQ6.20)}$$

Thus a position with a perfectly conserved residue will have the maximum amount of information.

In practice this equation must be slightly modified. Firstly, the average amino acid composition will not be uniform, and the entropy of that composition can be used instead of $\log_2 20$. Also, account must be taken of the small sample sizes, which potentially underestimate the information content. (If there are only two sequences, for example, H_u can never exceed $\log_2 2$.) For details of this correction see Further Reading.

Thomas Schneider and Michael Stephens used the information measure to produce a simple graphic that shows not only the amount of information present at each position but also the residues and their contributions. The contribution of a residue is simply defined as $f_{u,a}I_u$. At every position the residues are represented by their one-letter code, with each letter having a height proportional to their contribution. An example is given in Figure 6.5. These logos are a good visual summary of the profile, although they cannot easily cope with variable insertions. The same method can be equally successfully applied to nucleotide sequences, as will be seen in Chapter 10.

Figure 6.5
Example of a protein sequence alignment logo, taken from the BLOCKS database. This is block IPB000399E, constructed from an alignment of 186 sequences of TDP (thymine diphosphate)- binding enzymes. This region of sequence is also present in PROSITE (entry PS00187), which reports a TPP (thiamine pyrophosphate)- binding pattern [LIVMF]-[GSA]-x(5)-P-x(4)-[LIVMFYW]-x-[LIVMF]-x-G-D-[GSA]-[GSAC]. (See Table 4.2 for the notation used in PROSITE patterns.) Note that this pattern is only found in a subset of 44 of the 186 sequences whose alignment is shown here. The conserved proline of this pattern is in column 18, and the G-D-[GSA] at columns 27–29. Produced at http://weblogo.berkeley.edu



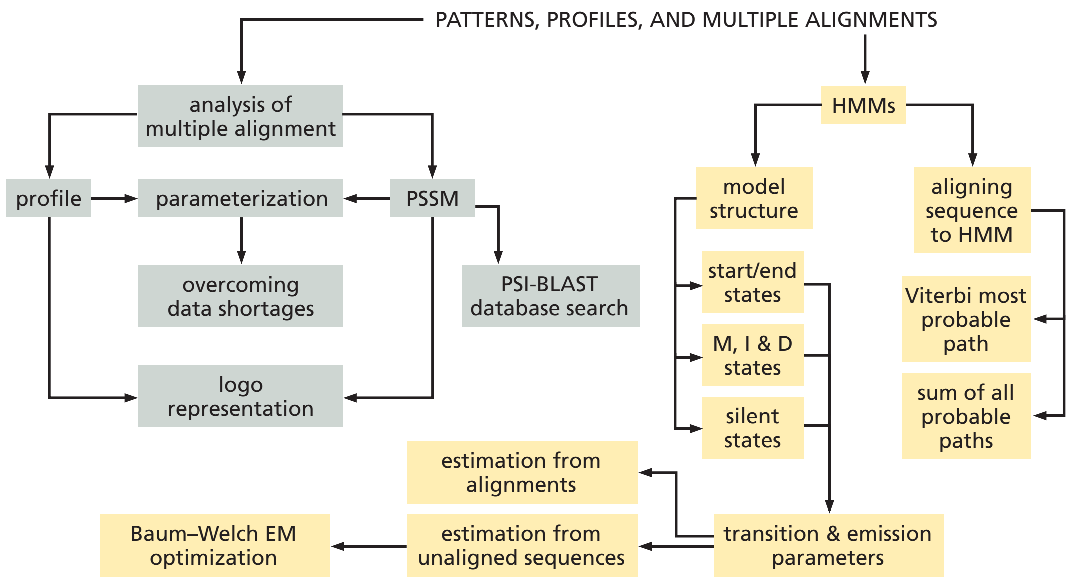
6.2 Profile Hidden Markov Models

In Section 5.3 we saw how finite-state automata can be used to find a specified subsequence or string within a sequence (see Figure 5.23). The automaton model has to be specially constructed for the desired string. The model consists of several states and it can only be in one type of state at any one time. There is a limited set of transitions allowed between states, and the output of the model (the sequence) is referred to as the emission of, in this case, one residue at a time. This idea can be extended to make models that can be used in many other sequence-recognition applications. An important aspect of some of these more complex models is the introduction of probabilities for transitions between states and emissions from states. The class of models that will be presented here are generally called hidden Markov models (HMMs) (see Flow Diagram 6.2). HMMs can be designed for many different tasks, of which sequence alignment is just one. In this section we will study HMMs designed to align a sequence to a profile. Other applications of HMMs will be encountered elsewhere in this book, for example for gene identification in Chapter 10 and for transmembrane helix prediction in Section 12.5, as it is a versatile way of constructing a variety of models of relevance to bioinformatics.

An HMM is defined by having a set of **states**, each of which has a limited number of **transitions** to other states and a limited number of **emissions** from a given state. Each transition between states will have an assigned probability, the value of which is independent of the history of previous states encountered. It is this property that makes these a class of **Markov models**. Each of the models considered here has a **start state** and an **end state**, and any **path** through the model from the start to the end state will produce a sequence. In these HMMs the state emission is of a residue of the sequence. As will be shown below, there are many alternative paths through the model that can produce the same sequence, and the *i*th residue of the sequence may have been emitted from any of a number of alternative states. The sequence alone has no information about the state from which each residue arose. This is the hidden nature of these models.

It is possible to define an HMM that contains the information present in a multiple alignment so that it becomes an alternative representation of a PSSM. The resultant model is usually referred to as a **profile HMM**. Profile HMMs can be seen as a more sophisticated version of a PSSM, especially because of the position-dependent way in which insertions and deletions are treated. Many HMM practitioners emphasize the fact that HMMs have a sound theoretical foundation,

Flow Diagram 6.2
The key concept introduced in this section is that a class of hidden Markov models (HMMs) can be constructed to represent a sequence profile, and methods are defined for generating such models for use in searching for the profile in sequences.


PATTERNS, PROFILES, AND MULTIPLE ALIGNMENTS
analysis of multiple alignment
profile
parameterization
PSSM
overcoming data shortages
PSI-BLAST database search
logo representation
HMMs
model structure
aligning sequence to HMM
start/end states
M, I & D states
silent states
Viterbi most probable path
sum of all probable paths
estimation from alignments
Baum–Welch EM optimization
estimation from unaligned sequences
transition & emission parameters

permitting accurate estimation of the likelihood of observing the alignment produced. This can be especially useful when searching for new members of a sequence family. It is always preferable to be able to take decisions about the inclusion of a distantly related member based on precise statistical analysis. (This is why the PSI-BLAST PSSM parameters were scaled so carefully.)

This section starts with a look at the structure of profile HMMs. Each alignment position has a standard structure, consisting of several different states with a limited set of allowed transitions. Some additional model architectures are required to permit the alignment of sequences that either do not fully or do not only cover the profile. After this, the model parameters will be defined and a method presented for deriving their values when a multiple alignment is available. (This is the exact analogy of fitting the PSSM values and, as will be seen, involves a similar concern for the potential lack of sufficient data.) Once a profile HMM has been defined, sequences can be aligned to it. Any path through the model and the associated emitted sequence can be assigned a probability based on the transition and emission probabilities. As is the case for pairwise sequence alignment, there are many possible alignments, paths, of which we are most likely only to be interested in the optimal alignments. These will be shown to be obtainable for profile HMMs in ways related to the dynamic programming techniques shown in Section 5.2. However, profile HMMs can produce the same alignment by many different paths through different states. This is not the case with the dynamic programming methods mentioned, for which a particular alignment is associated with only one path. A way to identify the most probable sequence will be discussed, as will be the ability to estimate the most likely state from which a given residue was emitted. The final part of this section will explore how a profile HMM can be parameterized using unaligned sequences.

The basic structure of HMMs used in sequence alignment to profiles

All the HMMs discussed here consist of a number of states that are linked together by transitions. An example of a state is one that matches (i.e., aligns) residues at a specific position (column) of an alignment. This state may be linked to the matching state for the next alignment position by a transition that specifies the direction of the connection. All transitions have an associated probability, such that for every state that has transitions to other states, the sum of all such probabilities must sum to exactly 1. Thus, it is certain that a transition will occur from any such state, although it is possible for the transition to be directly back to the same state. The only state present in the HMMs discussed here that does not have this property is the end state, because on reaching the end state the HMM terminates, so that state has no transitions. The sum of probabilities for all transitions to a given state need not add to 1, resulting in certain states being more likely to be visited.

A state of a profile HMM may emit a residue, so that a path through the model via a series of states will define the alignment of a sequence. The order of the residues emitted along the path must correspond to that of the query sequence. In general, any state that emits residues can emit any type and has defined probabilities for each possible type, with the total emission probability summing to 1. Not all states in a model are required to emit residues, and those that do not are sometimes called **silent states**.

It should be noted that there are some significant differences between the HMM and the finite-state automaton of Figure 5.23. In the latter, the sequence is emitted on transition from one state to another, and the transitions are labeled with these residues. In HMMs, the emission occurs when the state has been entered, and the transitions themselves are not directly connected to the query sequence. In addition, the costs associated with transitions in finite-state automata are not necessarily

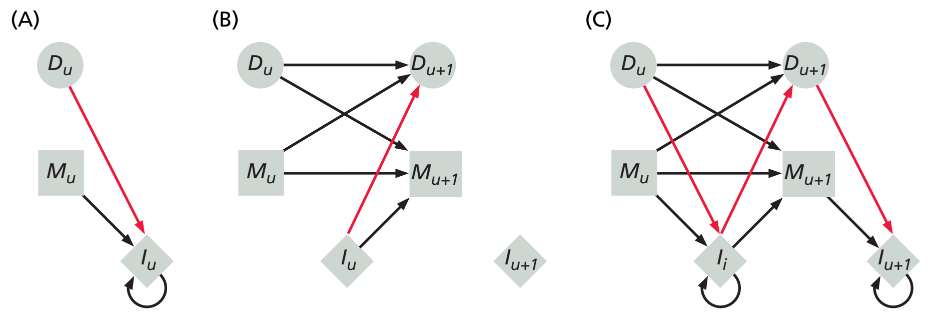


Figure 6.6
The repeating architecture of a profile HMM. (A) The transitions between the three states of a profile HMM associated with position u. The match, insert, and delete states are labeled M_u, I_u, and D_u, respectively. Note the red transition, which is often not included in models. (B) The transitions from the three states of a profile HMM associated with position u and those states associated with position $u + 1$. Again, a red transition is often omitted from models. (C) The full model, being the two sets of transitions combined.

probabilities, and paths through such models are unlikely to have a probabilistic interpretation.

There are three possible outcomes when considering aligning the next residue of a query sequence to a reference sequence alignment. First, the query sequence residue may align (or match) with the next residue of the reference. Second, it may correspond to an insertion relative to the reference. Third, the query sequence residue may correspond to a later position in the reference, indicating a gap (or deletion) in the query. Profile HMMs contain a set of three states associated with each alignment position to model these alternatives. For the uth alignment position the match, delete, and insert states will be referred to as M_u, D_u, and I_u, respectively, and will be represented on diagrams by a square, circle, and diamond, respectively (see Figure 6.6). The delete state is an example of a silent state, as it does not emit any residues of the sequence.

A few transitions are allowed between M_u, D_u, and I_u in all profile HMMs (see Figure 6.6A). As the insert state emissions correspond to residues inserted after the uth position, there is a transition from M_u to I_u. In many models there is also a transition from D_u to I_u, although a number of models omit this because when such transitions are present they tend to have low probability, often resulting in low-scoring paths. Furthermore, the alignments they represent can be obtained by alternative paths through the HMM involving the match state. In addition, there is a transition from the I_u state to itself to allow insertions of more than one residue before alignment position $u + 1$. (The alignment is strictly only of those residues emitted by the match states, so that sequence residues emitted by the insert states are not regarded as occupying an alignment position.)

Several transitions connect the uth position states to states at position $u + 1$, all in the forwards direction from u to $u + 1$ (see Figure 6.6B). The M_{u+1} and D_{u+1} states can be reached from all three of the uth position states. This means that there is a path connecting successive delete states, corresponding to a large deletion. The insert states are only accessed via transitions from the match and delete states at the same position. Transitions between insert states I_u and I_{u+1} are not allowed because they would imply the absence of a match state between them, and hence delete state D_{u+1} should be involved. However, note that the transition from I_u to D_{u+1} is often omitted, for the same reasons given above in the case of D_u to I_u. The full set of states for positions u and $u + 1$ are shown in Figure 6.6C, together with all the transitions involving just this set of states.

As discussed above, in addition to transitions between states, HMMs also have emissions from states. Match and insert states emit a residue, while delete states do not. Recalling the discussion of substitution matrices in Section 5.1, there are two basic models of sequence alignment to consider: the random and nonrandom (that is, related) models. If there is an insertion in the query sequence relative to the HMM, the residues involved are, by definition, not related to residue information in the HMM. The emission probabilities for insert states are often assigned according to the basic residue frequencies p_a as described in Section 5.1. In contrast, the residues emitted by match states are related, and so the emission probabilities depend on the multiple alignment. They are most closely related to the PSSMs discussed above, as they vary with position in the alignment.

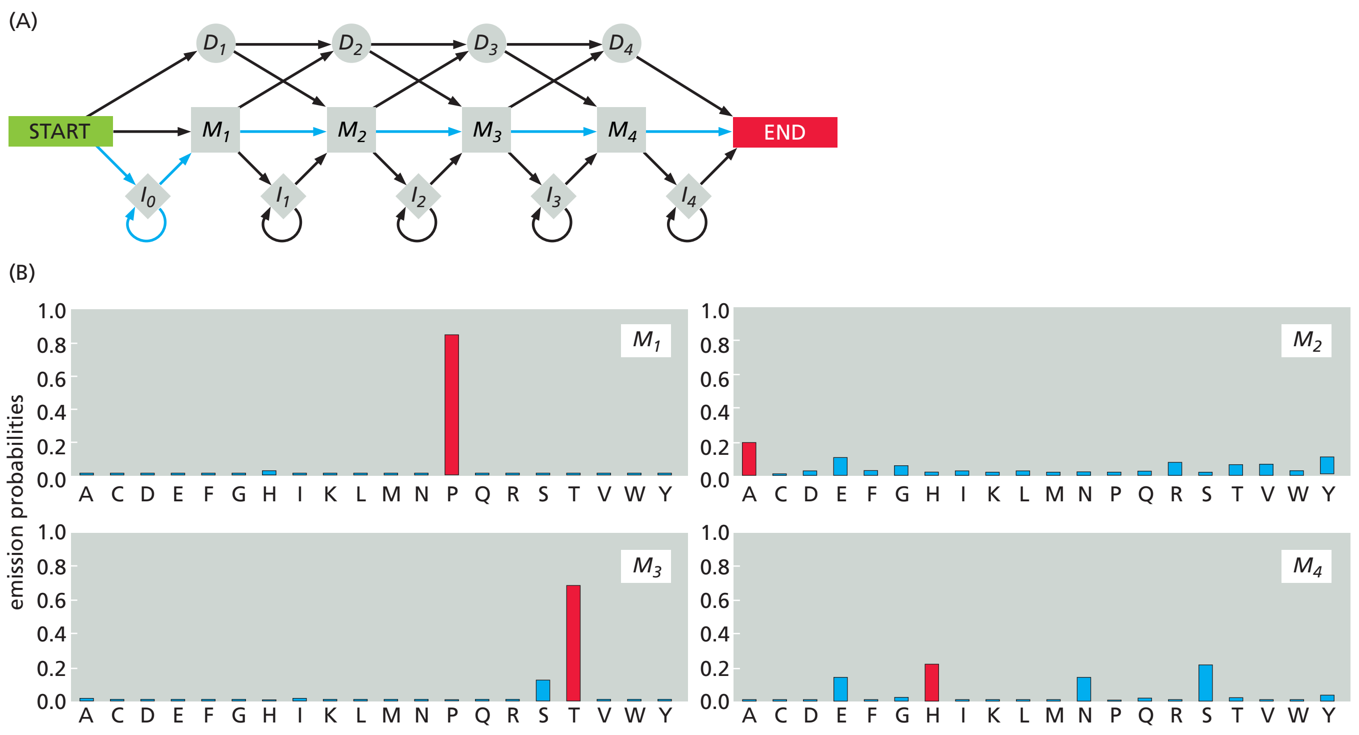
Unlike the PSSMs discussed above, which were constructed with as many positions as in the query sequence, the structure of the equivalent profile HMM is not quite as straightforward. Firstly, in contrast to PSSMs, profile HMMs can be obtained from a set of unaligned sequences. In such cases the number of match states is often set to the average length of the sequences that define the profile, since this will hopefully reduce the need for insert and delete states. However, it is often useful to try some alternative models, as they can lead to better results. The HMM program SAM examines the existing HMM to identify match states that are used by relatively few sequences, or insert states used by many sequences, and then respectively removes or inserts positions in the HMM. The process of exploring alternative model structures is sometimes called **model surgery**. When fitting a profile HMM to a multiple alignment a different approach is often taken, in which those alignment columns that have more gaps than some specified fraction are assumed to be insertions, other columns being taken as indicating a match state.

Each transition and emission in the model is a parameter that needs to be defined. The basic HMM structure for an alignment position normally has either nine or seven transitions (see Figure 6.6C). If all insert states are assigned identical emission probabilities, these can be precalculated from the residue composition or refined as 20 parameters. The match-state emission probabilities will be different for each alignment position, each requiring 20 parameters. Ignoring the insert-state emission, this adds up to as many as 29 parameters for each alignment position, so that a profile HMM with 100 match positions and associated states, transitions, and emissions will have several thousand parameters. This has important consequences, as almost certainly a lot of sequence data will be required to derive a good parameter set for an HMM of this size.

The path taken through the model, resulting in the emission of a sequence, can be interpreted as assigning each of the residues to be either aligned to a particular HMM position (corresponding to the particular match state) or else inserted at a particular position (corresponding to the particular insert state). Note that when comparing the assignments for several sequences, those residues from different sequences that were emitted by the same match state are regarded as aligned with each other. However, those emitted by the same insert state are not regarded as aligned, even though they may be shown in the same column of a summary alignment.

So far we have only discussed the states required in the section of the profile HMM that aligns the relevant region of the query sequence (for example, the domain that the profile HMM represents). The HMM must also contain special sections for starting and ending the alignment. In addition, some HMMs contain features designed to cope with query sequences containing unrelated regions of sequence as well as the relevant region, for example other domains not modeled in this HMM. Such HMMs model local as opposed to global alignments. Other HMMs are designed to allow for repetition of segments of sequence, even a complete domain. We will now examine these other features of profile HMMs as used in some of the commonly available packages.

All the profile HMMs described here require states at the start and end of the model. Figure 6.7A shows a profile HMM configuration including such start and end states. The start state does not emit any residues and has no transitions to itself, but has three transitions to other states. All paths through the HMM start at this state. The transition to M_1 will occur when the query sequence amino-terminal residue matches the first position in the profile HMM. If the query sequence has extra residues at its amino terminus, these will be emitted by the I_0 state. Finally, if the



first-position residue is missing from the query the transition used will be from start to D_1. The end state shown in Figure 6.7A has transitions to it from the three states associated with the final profile position and does not emit a residue, nor have any transitions from itself. All paths through the HMM terminate at this state. In the model shown, any extra residues in the query sequence after the profile will have to be emitted by the last insert state. Figure 6.7B shows example emission probabilities from the four match states. These sum to 1 for each state, and show that each alignment position has particular residue preferences. For example, state M_1 represents a highly conserved proline, and M_3 a conserved small polar residue. The insert state emission probabilities are not shown, but would normally be the same for all insertion states and related to the overall amino acid composition.

All the models considered here represent a profile against which a query sequence is to be aligned. We must allow for the query sequence to be wholly unrelated to the profile. In the model discussed so far, such an unrelated sequence would be emitted by paths that visited many delete and insert states. It should be recalled that only the sequence emitted by the match states can be regarded as aligned to the profile HMM, and any residues emitted by insert states are unaligned by definition. In addition, an unrelated sequence will align to a profile HMM with a very low probability or log-odds score. As in the case of database searches, unrelated sequences will usually be identified on these quantitative scores rather than a qualitative assessment of the alignment itself.

Another possible situation arises if the query sequence is longer than the profile, so that there are segments of sequence that are unrelated. The initial and final insert states can represent any unrelated flanking sequences, allowing for a local alignment of the query sequence. However, the model of Figure 6.7 is not the full equivalent of Smith–Waterman local alignment (see Section 5.2) because the path taken through the HMM by the sequence will still include all profile positions. Alignment of only a part of the profile will involve a path through many delete states, which will almost certainly occur with very low probability if the model is parameterized for sequences with the full-length profile. A more realistic local alignment model is shown in Figure 6.8A. This involves two new silent states that are directly connected

Figure 6.7
A complete profile HMM model, with a start and end state and four match states. The existence of states I_0 and I_4 allows the profile to occur anywhere within a larger sequence. (A) The organization of the states and allowed transitions. Note that each of the transitions in the model will in general have a different value for the transition probabilities. One of the possible paths through the model is marked by blue transition arrows, and produces a sequence that has two or more residues at the amino terminus, followed by the profile with no insertions, immediately followed by the carboxyl terminus. (B) The emission probabilities from the four match states. Ignoring the residues emitted from state I_0, the sequence PATH has the greatest probability of being emitted, but many other sequences are almost as likely to be emitted, for example, PETS. Note that the emissions for each state sum to 1.

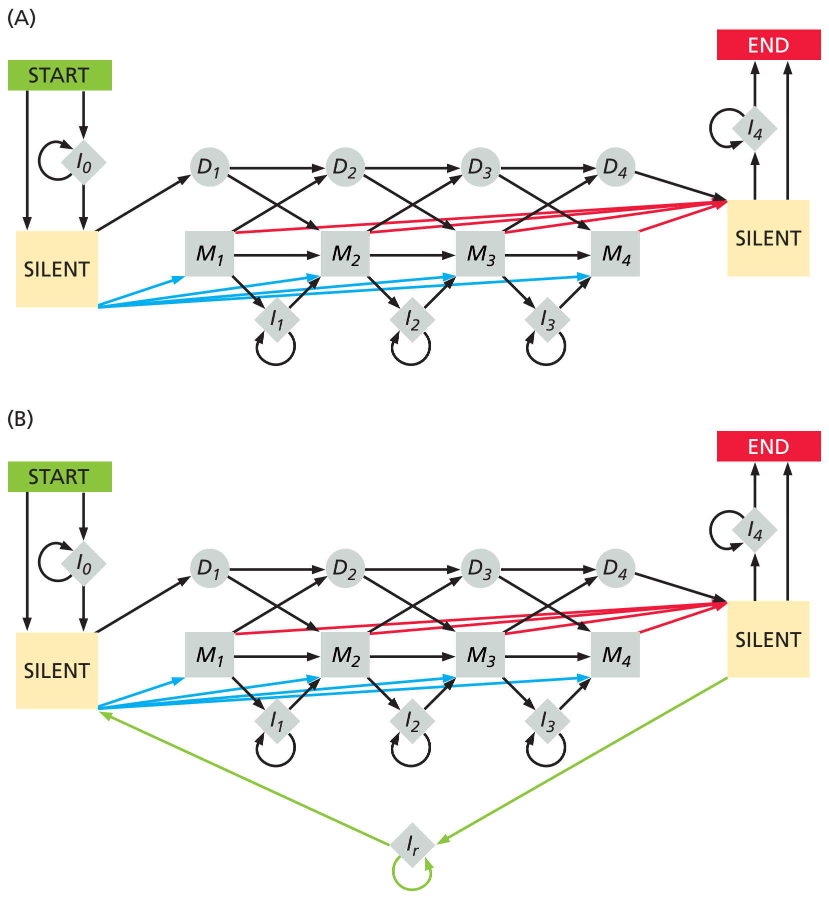


Figure 6.8
Complete profile HMM models for local and repeat local profile alignment. (A) Complete profile HMM model allowing for local alignment, including the presence of only part of the profile in the query sequence. Note the two silent states, and the change in architecture in the region of the I_0 and I_4 states, as compared with Figure 6.7. The I_0 state models all the sequence prior to the profile, and the I_4 state all the sequence after it. The blue and red transitions connect the silent states to all match states, allowing only a part of the profile to be included in the path. (B) The same HMM augmented by another insert state (I_r) linking the two silent states, and allowing the profile to be repeated on the path.

to every match state of the model, as well as to two insert states that model the flanking sequences of the query. These silent states are not necessary, but they reduce the number of transitions required in the model, making parameterization easier. Without them, the start and end states would also need to be connected to all match states to allow for the absence of flanking regions. Note that the red transitions of Figure 6.6C are missing in this model.

Although the insert states I_0 and I_4 of Figures 6.7 and 6.8 can model any flanking sequence segments that are not part of the profile, further design is needed to appropriately model such regions. For example, if simple insert states are used, at least one residue must be emitted at each end of the sequence, and these terminal residues will not be part of the profile alignment. This can be avoided by a more complex model structure. In the SAM program such structures are referred to as **free insertion modules** (FIMs) and, in addition to the insert state, include a delete state. The FIM is always entered via this delete state and can be left via either state. This allows for the possibility of no residues in a flanking sequence. The delete state has a transition to the FIM insert state, the insert state has a transition back to itself, and both have transitions that leave the FIM.

As described so far, with probabilities constrained to sum to 1 for every state, all transitions and emissions in a profile HMM will affect the overall probability or score for the sequence. This is undesirable in the case of flanking sequences, as it is only the region related to the profile that is of interest. The contribution of these flanking regions can be controlled by relaxing the probability constraints to set the transition from the FIM delete to the FIM insert state and all the transitions leaving a FIM a probability of 1. The only remaining influence on scoring will be from the emitted residues, but these are usually assigned the probabilities of the null, i.e., random model, and as a result have no influence.

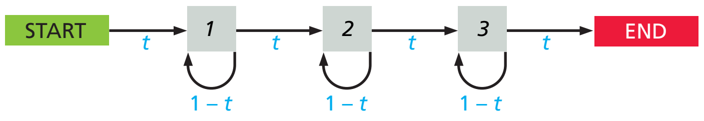


Figure 6.9
A simple HMM used in the discussion of the length dependence of HMM models. There are three match states between the start and end states. As there are no insert or delete states, all the match states must be included at least once in each path.

The HMM can be further enhanced to permit repeats of the profile by connecting the silent states of Figure 6.8A together. The connecting transition goes via an insert state (shown in Figure 6.8B) or more correctly via a FIM to model any intervening sequence, and must lead back to the state that has transitions to all the profile match states. This is the model used in the program HMMER2 but is closely related to those used in several other HMM packages.

The final issue relating to the basic HMM structure that we will consider is that the number of model states and the transition structure have an important influence on the distribution of path lengths through the HMM. This is easiest to understand by looking at some simpler HMMs, such as the one shown in Figure 6.9. All paths through the model shown in Figure 6.9 must pass at least once through each of the emitting states 1, 2, and 3, each of which emits a residue. Therefore, the minimal sequence length this HMM can model is three residues long. Furthermore, states 1, 2, and 3 have identical transition probabilities. Thus, any path corresponding to a sequence of L residues will involve three transitions of probability t (to move from start to end states) and $(L-3)$ transitions of probability $(1-t)$. Thus the probability of any path emitting L residues is $t^3(1-t)^{L-3}$. However, there are many different equally probable paths that emit L residues. We need to know how many in order to calculate the probability of this model producing an L-residue sequence. First, note that every path must begin with a transfer from the start state into match state 1. Thus, the paths can only differ in their remaining $(L-1)$ transitions, of which there will be $(L-3)$ transitions returning to the same match state (1 to 1, 2 to 2, or 3 to 3). The paths can only differ in the ordering of these $(L-3)$ transitions within the remaining $(L-1)$. Combinatorial analysis tells us that there are

$$^{L-1}C_{L-3} = \frac{(L-1)!}{(L-3)!2!} \qquad \text{(EQ6.21)}$$

different orderings and hence different paths. As these are all of equal probability, the probability of a path in this model having length L is $^{L-1}C_{L-3}\, t^3(1-t)^{L-3}$. For a general model with N match states, the probability of a path of length L is

$$^{L-1}C_{L-N}\, t^N(1-t)^{L-N} = \frac{(L-1)!}{(L-N)!(N-1)!}t^N(1-t)^{L-N} \qquad \text{(EQ6.22)}$$

This distribution is shown in Figure 6.10 for several different values of t and numbers of match states N. It can have an important influence on the success of the HMM, because the model should ideally have the same distribution of lengths as the set of sequences it is trying to represent. If t is made sufficiently small, the most likely path lengths can become very long, but such models are not encountered in realistic protein-sequence problems.

Estimating HMM parameters using aligned sequences

The structure and parameters of a profile HMM can be derived from a set of protein sequences. These sequences may already be in a multiple alignment or they may as yet be unaligned, waiting for the HMM to do the work. Here we will consider parameterization using alignments, leaving the unaligned case until later.

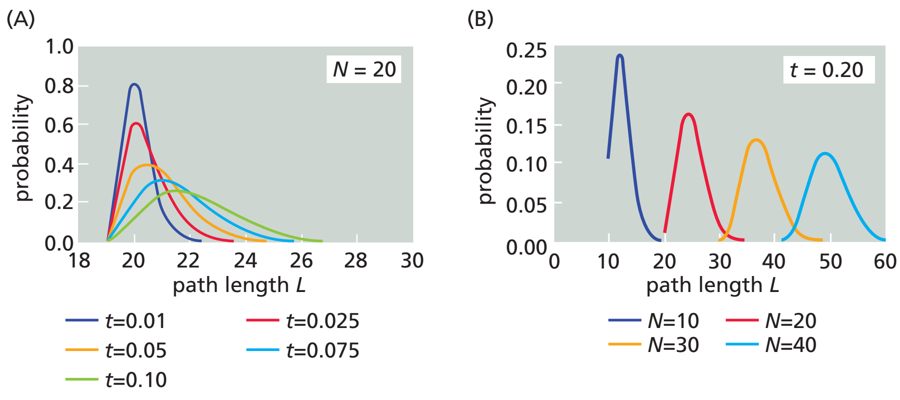


Figure 6.10
Length distributions for the simple HMMs shown in Figure 6.9. (A) For a model with 20 match states, the probability of paths of length *L* for different values of the probability of a transition back to the same match state *t*. (B) The probability of paths of length *L* for models with different numbers of match states when the transition probability *t* is 0.20.

To use a multiple sequence alignment for profile HMM parameterization we must first decide which alignment positions correspond to which HMM match states, and which are to be modeled as insert states. An alignment column that contains no gaps should be assigned to a match state, and one with a majority of gaps to an insert state, but intermediate situations need careful consideration. Usually, a threshold proportion of gaps is selected to determine the assignments. Once these assignments have been made, the path that each sequence follows through the HMM can be deduced. In this way, the alignment data can be translated into the frequencies of transitions between particular states along the path and the frequencies of emission of individual residues from particular states. Note that more sophisticated parameterization methods have been proposed that simultaneously try to find the best column assignment during the parameterization (see Further Reading).

Given a set of sequences with known paths through the profile HMM, parameterization is in principle straightforward. All states in the HMM, except the end state, have some transitions from that state to others (in some cases including looping transitions back to itself). The total probability for these transitions must sum to 1 for each state, so that there is certainty that one of these transitions will be taken. (Exceptions such as states in FIMs will be ignored here.) For the state *u* the transition to another state *v* will be written $t(u,v)$. The summation over all states *w* that are connected to state *u* by transitions (including possibly state *u* itself) gives

$$\sum_w t(u,w) = 1 \tag{EQ6.23}$$

If the paths taken by the sequences used for parameterization contain a total of $m_{u,v}$ transitions from *u* to state *v*, then the model transition probabilities can be estimated by

$$t(u,v) = \frac{m_{u,v}}{\sum_w m_{u,w}} \tag{EQ6.24}$$

Emission probabilities for match states can be derived in an analogous way. The emission probabilities $e_{M_u}(a)$ for a residue *a* from the *u*th match state M_u add up to 1 where all possible residue types are considered. If the parameterization data contain $n_{M_u,a}$ emissions of residue *a* from match state M_u (which would result from observing that number of occurrences of residue *a* in the alignment column corresponding to match state M_u), then the model emission probability can be estimated by

$$e_{M_u}(a) = \frac{n_{M_u,a}}{\sum_b n_{M_u,b}} \qquad \text{(EQ6.25)}$$

where the summation is over all the residue types b. The insert state emission probabilities $e_{I_u}(a)$ are not calculated in this way. The model for insert states is the random model, so that the probabilities are usually taken from the overall amino acid composition of a selected data set. Thus

$$e_{I_u}(a) = p_a \qquad \text{(EQ6.26)}$$

Unfortunately, parameterization is more difficult in practice because there is almost always a lack of data. If there are insufficient sequences for parameterization, many of the counts $m_{u,v}$ and especially $n_{M_u,a}$ will be zero or very small, resulting in very poor estimates of the probabilities. This is especially the case for emission parameters. For protein sequence HMMs a minimum of 20 sequences is required simply to observe each possible match state emission, but many more are required to obtain realistic estimates of the emission probabilities. If no transitions or emissions have been observed, this will lead to $t(u,v)$ or $e_{M_u}(a)$ being set to zero for these events. Using zero probabilities will result in the HMM being unable to accurately align sequences that require these events as they will be impossible in the model. An HMM that has zero probability for emission of a particular residue from a given match state cannot align a sequence with that residue at that position. It can only make the residue an insertion before or after the position, in which case it does not count as aligned. Even when a relatively large quantity of sequence data is available, parameterization problems often occur for positions with very highly conserved residues or sequence segments.

This problem is identical to that discussed above in the context of PSSM parameterization. In the case of emission probabilities the solution is very similar. The formulae presented earlier for $q_{u,a}$ can be applied immediately, so that for example the equivalent of Equation EQ6.10 is

$$e_{M_u}(a) = \frac{n_{M_u,a} + 1}{\sum_b n_{M_u,b} + 20} \qquad \text{(EQ6.27)}$$

Other pseudocount methods can be applied, deriving expressions by reference to the previous formulae for PSSM parameters.

Scoring a sequence against a profile HMM: The most probable path and the sum over all paths

Given a parameterized profile HMM, any given path through the model will emit a sequence with an associated probability. This path probability will be the product of all the transition and emission probabilities along the path. In addition, the path defines how the emitted sequence is aligned to the model. Residues emitted from match states will be aligned to that position in the profile. Residues emitted from insert states should not be regarded as aligned, but the position of the insertion is clear. Normally, the sequence is specified and we wish to find the alignment and score against the profile HMM.

There can be many paths through the HMM that result in the same emitted sequence, each with a different alignment to the profile. This is analogous to the many possible alignments of one sequence to another. Of these many paths, some or one will be the most probable, analogous to the best-scoring alignment in dynamic programming. However, the probability of the sequence being emitted by the HMM is given by the sum of the probabilities for all possible paths. This has no equivalent in dynamic programming.

Methods exist for calculating the most probable path and the total probability, both of which are quite similar to the dynamic programming methods discussed in Section 5.2. Using probabilities is computationally impractical, because in realistic situations it leads to very small numbers. This technical problem can be avoided by taking logarithms. However, log-odds scores (the log of the ratio of probabilities of alternative models) are preferable, using a random model as in Section 5.1, as this is easier to analyze to provide an estimate of the significance of the score. The random model of an L residue sequence is the product of all the L terms p_a (the residue composition frequency). If each of the L emission probabilities, each for a particular state u emitting $e_u(a)$, is divided by its associated p_a then the emission terms will include all the terms of the random model and the transition terms $t(u,v)$ can remain unchanged. Therefore the equations that follow will contain terms such as $\ln(e_u(a)/p_a)$ and $\ln(t(u,v))$ where u and v are model states and a is the emitted residue type.

We will deal first with calculating the most probable path, which can be found using the **Viterbi algorithm**. This will be applied to a profile HMM with a structure very similar to that shown in Figure 6.7A, except that it will have an arbitrary number of sets of match M, delete D, and insert I states. The states are labeled as was described earlier when presenting the basic profile HMM structure. At a profile position u, these states will be called M_u, I_u, and D_u. Hence the transition probability from M_u to M_{u+1} will be written $t(M_u,M_{u+1})$. The start and end states will here be labeled *"Start"* and *"End,"* respectively. However, the formulae will only be shown for the regular repeating structure. The formulae involving the *Start* and *End* states are a modification of those presented here. If the query or emitted sequence $\boldsymbol{x}$ is L residues long and is labeled $x_1x_2x_3...x_L$, then the emission probability for residue x_i from insert state I_u is written $e_{I_u}(x_i)$.

During the Viterbi algorithm, a record must be kept of the highest probability or best log-odds score up to that point in the model and for a given amount of emitted sequence. For example, at the state D_u, when the sequence up to and including residue x_i has been emitted, the highest probability will be written $v_{D_u}(x_i)$ and the best log-odds score will be written $V_{D_u}(x_i)$. Note that this calculation is equivalent to a dynamic programming matrix with one side of length equal to the number of states in the HMM, and the other side being the length of the query or emitted sequence

The Viterbi algorithm, like dynamic programming, starts with very short emitted sequences involving paths of only very few HMM states, so that the best path probabilities and log-odds scores are readily calculated, and then builds on these. The calculation usually uses initialization values that start the path at a particular state with no sequence emitted. For example, in probability calculations the initialization for the model in Figure 6.7A might be $v_{Start}(0) = 1$, $v_u(0) = 0$ for any or other state u. The log-odds equivalents are $V_{Start}(0) = 0$, $V_u(0) = -\infty$ for any or other state u. This results in all paths beginning at the *Start* state.

The key stage in the algorithm is to derive values for the states in the uth profile position based on values at states which have transitions to them, wholly or mostly from the u–1th profile position. The following equations deal only with the basic profile HMM model shown in the central part of Figure 6.7A. We will also assume that the insert state emission probabilities are given by the amino acid composition p_a for residue type a, written p_{x_i} in the following equations where they represent emission of the residue type present at position x_i.

The formulae for the probability calculation are

$$
\begin{aligned}
v_{M_u}(x_i) &= e_{M_u}(x_i)\max\begin{cases} v_{M_{u-1}}(x_{i-1})\,t(M_{u-1},M_u) \\ v_{I_{u-1}}(x_{i-1})\,t(I_{u-1},M_u) \\ v_{D_{u-1}}(x_{i-1})\,t(D_{u-1},M_u) \end{cases} \\
v_{I_u}(x_i) &= p_{x_i}\max\begin{cases} v_{M_u}(x_{i-1})\,t(M_u,I_u) \\ v_{I_u}(x_{i-1})\,t(I_u,I_u) \end{cases} \\
v_{D_u}(x_i) &= \max\begin{cases} v_{M_{u-1}}(x_i)\,t(M_{u-1},D_u) \\ v_{D_{u-1}}(x_i)\,t(D_{u-1},D_u) \end{cases}
\end{aligned}
\qquad \text{(EQ6.28)}
$$

Note that other transitions, such as those shown in red in Figure 6.6C, may also occur in some other model profile HMMs. These would lead to extra terms corresponding to these extra transitions. For traceback to determine the path, a record must be kept of which term was selected as the maximum in each formula. For computational reasons the logarithms of probabilities are always used, which replaces the multiplication with addition. For example,

$$
\begin{aligned}
\log v_{M_u}(x_i) &= \log e_{M_u}(x_i) + \max\begin{cases} \log v_{M_{u-1}}(x_{i-1}) + \log t(M_{u-1},M_u) \\ \log v_{I_{u-1}}(x_{i-1}) + \log t(I_{u-1},M_u) \\ \log v_{D_{u-1}}(x_{i-1}) + \log t(D_{u-1},M_u) \end{cases} \\
\log v_{I_u}(x_i) &= \log p_{x_i} + \max\begin{cases} \log v_{M_u}(x_{i-1}) + \log t(M_u,I_u) \\ \log v_{I_u}(x_{i-1}) + \log t(I_u,I_u) \end{cases} \\
\log v_{D_u}(x_i) &= \max\begin{cases} \log v_{M_{u-1}}(x_i) + \log t(M_{u-1},D_u) \\ \log v_{D_{u-1}}(x_i) + \log t(D_{u-1},D_u) \end{cases}
\end{aligned}
\qquad \text{(EQ6.29)}
$$

Using a **null model** the same as the model used for insert emissions (Equation EQ6.26), we can obtain log-odds scores. Using the HMM model of Figure 6.6C, we can derive log-odds scores as

$$
\begin{aligned}
V_{M_u}(x_i) &= \log\left(\frac{e_{M_u}(x_i)}{p_{x_i}}\right) + \max\begin{cases} V_{M_{u-1}}(x_{i-1}) + \log t(M_{u-1},M_u) \\ V_{I_{u-1}}(x_{i-1}) + \log t(I_{u-1},M_u) \\ V_{D_{u-1}}(x_{i-1}) + \log t(D_{u-1},M_u) \end{cases} \\
V_{I_u}(x_i) &= \max\begin{cases} V_{M_u}(x_{i-1}) + \log t(M_u,I_u) \\ V_{I_u}(x_{i-1}) + \log t(I_u,I_u) \\ V_{D_u}(x_{i-1}) + \log t(D_u,I_u) \end{cases} \\
V_{D_u}(x_i) &= \max\begin{cases} V_{M_{u-1}}(x_i) + \log t(M_{u-1},D_u) \\ V_{I_{u-1}}(x_i) + \log t(I_{u-1},D_u) \\ V_{D_{u-1}}(x_i) + \log t(D_{u-1},D_u) \end{cases}
\end{aligned}
\qquad \text{(EQ6.30)}
$$

For comparison with the previous equations, these are for the basic profile HMM unit as shown in Figure 6.6C, with all nine possible transitions given in these equations. Only the match state M_u equations include an emission term because the delete state D_u does not emit, and the insert state I_u emission term becomes 0 on dividing by the null model. Note that the emitted residue is dictated by the query sequence, and must be the the next residue in the sequence $\boldsymbol{x}$.

The termination of this calculation will depend on the precise details of the model in the region of the end state. If the end state is called *End* the value of $V_{End}(x_L)$ will be the log-odds score of this best path. Because this state does not emit a residue, this last formula will lack emission terms.

Algorithms quite similar to these can calculate the total probability or total log-odds score for all paths that emit the query sequence. The main difference is that instead of determining the maximum of three alternatives, the three probabilities are summed. However, there is a significant practical difference, because actual probabilities must be calculated, not log-odds scores. The method is known as the **forward algorithm** because it progresses from the start state to the end state of the HMM. At the state M_u, when the sequence up to and including residue x_i has been emitted, the total probability is written $f_{M_u}(x_i)$ and the total log-odds score is written as $F_{M_u}(x_i)$. The initialization is performed as was discussed previously. By analogy with Equation EQ6.28 we have

$$\begin{aligned}
f_{M_u}(x_i) &= e_{M_u}(x_i)\left[f_{M_{u-1}}(x_{i-1})t(M_{u-1},M_u) + f_{I_{u-1}}(x_{i-1})t(I_{u-1},M_u) + f_{D_{u-1}}(x_{i-1})t(D_{u-1},M_u)\right] \\
f_{I_u}(x_i) &= p_{x_i}\left[f_{M_u}(x_{i-1})\,t(M_u,I_u) + f_{I_{u-1}}(x_{i-1})\,t(I_{u-1},I_u)\right] \\
f_{D_u}(x_i) &= \left[f_{M_{u-1}}(x_i)t(M_{u-1},D_u) + f_{D_{u-1}}(x_i)t(D_{u-1},D_u)\right]
\end{aligned} \qquad \text{(EQ6.31)}$$

The log-odds score version of this, by analogy to Equation EQ6.30 is

$$\begin{aligned}
F_{M_u}(x_i) &= \log\left(\frac{e_{M_u}(x_i)}{p_{x_i}}\right) + \log\left[t(M_{u-1},M_u)e^{F_{M_{u-1}}(x_{i-1})} + t(I_{u-1},M_u)e^{F_{I_{u-1}}(x_{i-1})} + t(D_{u-1},M_u)e^{F_{D_{u-1}}(x_{i-1})}\right] \\
F_{I_u}(x_i) &= \log\left[t(M_u,I_u)e^{F_{M_u}(x_{i-1})} + t(I_u,I_u)e^{F_{I_u}(x_{i-1})} + t(D_u,I_u)e^{F_{D_u}(x_{i-1})}\right] \\
F_{D_u}(x_i) &= \log\left[t(M_{u-1},D_u)e^{F_{M_{u-1}}(x_i)} + t(I_{u-1},D_u)e^{F_{I_{u-1}}(x_i)} + t(D_{u-1},D_u)e^{F_{D_{u-1}}(x_i)}\right]
\end{aligned}$$

$$\text{(EQ6.32)}$$

As for the Viterbi algorithm, $F_{End}(x_L)$ will be the total log-odds score for obtaining the query sequence via all possible paths with this HMM. Again the *End* state does not emit a residue, so the formula to calculate it will lack this emission term.

An equivalent algorithm has also been proposed going backwards from the *End* state to the *Start* state. This is called the **backward algorithm**, and defines the terms b and B that are exactly analogous to the terms f and F. We are only interested here in the probabilities b as they will be used later. The probability of the sequence x_{i+1} to x_L being emitted, starting from state M_u when it emitted residue x_i, is written $b_{M_u}(x_i)$. The initialization is $b_u(x_L) = t(u,End)$ for all states u, which is only non-zero for those states with a direct transition to the *End* state. By analogy with Equations EQ6.28 and EQ6.31, using the HMM model of Figure 6.7A we have

$$\begin{aligned}
b_{M_u}(x_i) &= t(M_u,M_{u+1})e_{M_{u+1}}(x_{i+1})b_{M_{u+1}}(x_{i+1}) + t(M_u,I_u)e_{I_u}(x_{i+1})b_{I_u}(x_{i+1}) + t(M_u,D_{u+1})b_{D_{u+1}}(x_i) \\
b_{I_u}(x_i) &= t(I_u,M_{u+1})e_{M_{u+1}}(x_{i+1})b_{M_{u+1}}(x_{i+1}) + t(I_u,I_u)e_{I_u}(x_{i+1})b_{I_u}(x_{i+1}) \\
b_{D_u}(x_i) &= t(D_u,M_{u+1})e_{M_{u+1}}(x_{i+1})b_{M_{u+1}}(x_{i+1}) + t(D_u,D_{u+1})b_{D_{u+1}}(x_i)
\end{aligned} \qquad \text{(EQ6.33)}$$

Note that $f_u(x_i)$ is the probability of obtaining the sequence $\boldsymbol{x}$ from the start up to the ith residue in a path that ends at the uth model state. Similarly, $b_u(x_i)$ is the probability of obtaining the sequence $\boldsymbol{x}$ from the $(i + 1)$th residue to the end having emitted the ith residue by the time state u was left. Also note that $f_{End}(x_L) = b_{Start}(0) = P(\boldsymbol{x})$, the probability of observing the whole sequence $\boldsymbol{x}$ from the HMM.

Estimating HMM parameters using unaligned sequences

Returning now to the case where the sequences to be used for parameterization are not aligned, we must begin by deciding on the number of match states. These sequences (called the training sequences) should have been selected on the basis of having a common sequence region, possibly only a small part of the complete sequence. The number of match states should be fixed according to the anticipated features of the common region; that is, how many alignment columns are expected to contain residues as opposed to gaps in a majority of the sequences.

A starting set of parameters is required, which will be improved by iteration as described below. This set should be as good as possible to increase the chances of determining a good parameter set. For this reason, Dirichlet priors are often used to obtain a first estimate of the parameters. Furthermore, because there are general problems in locating the optimal parameter set, it is often advisable to generate several starting parameter sets and try each of them.

Several algorithms can be used to obtain HMM parameters. Pierre Baldi has proposed a technique involving maximum likelihood estimation methods where the optimization occurs via gradient descent. For more detail see Further Reading, although some of the key concepts are discussed in Appendix C.

The method we will discuss here is the **Baum–Welch expectation maximization** algorithm. The essential idea is to try to estimate the number of times each transition and emission occurs in the model when using the training sequences. These values are obtained by summation over all possible paths that produce the sequences, using the HMM with the current parameter set. The method uses the forward and backward algorithms. (A faster but less accurate variant uses only the highest-scoring paths; that is, Viterbi paths.) The estimated transition and emission frequencies are used to derive an improved parameter set. The training sequences are then rerun with the HMM parameters now set at their new values. Several cycles are run until the parameters have converged. This method is an example of the expectation maximization (EM) method (see Further Reading), and involves iteratively improving the values of the parameters, such that the likelihood of observing the sequences with the HMM should increase with each iteration.

First, note that $f_u(x_i)t(u,v)e_v(x_{i+1})b_v(x_{i+1})$ is the probability of emitting the sequence $\boldsymbol{x}$ summed over all paths that pass through states u and v, emitting x_{i+1} at state v. From this, the probability that the transition from state u to state v is taken with the emission of x_{i+1}, is given by

$$\frac{f_u(x_i)t(u,v)e_v(x_{i+1})b_v(x_{i+1})}{P(\boldsymbol{x})} \qquad \text{(EQ6.34)}$$

where $P(\boldsymbol{x})$ is the probability of observing sequence $\boldsymbol{x}$ by any path. (As noted above, the value of $P(\boldsymbol{x})$ can be obtained by the forward or backward algorithm.) By summing this formula over all values of sequence position i and over all training sequences, one can obtain the expected number of transitions $m_{u,v}$ for the training set. For the model illustrated in Figure 6.6C, with extra transitions allowed, we obtain the nine equations

$$
\begin{aligned}
m_{M_u,M_{u+1}} &= \sum_{seqs\,\boldsymbol{x}} \left[\frac{1}{P(\boldsymbol{x})} \sum_i f_{M_u}(x_i)\, t(M_u,M_{u+1})\, e_{M_{u+1}}(x_{i+1})\, b_{M_{u+1}}(x_{i+1}) \right] \\
m_{I_u,M_{u+1}} &= \sum_{seqs\,\boldsymbol{x}} \left[\frac{1}{P(\boldsymbol{x})} \sum_i f_{I_u}(x_i)\, t(I_u,M_{u+1})\, e_{M_{u+1}}(x_{i+1})\, b_{M_{u+1}}(x_{i+1}) \right] \\
m_{D_u,M_{u+1}} &= \sum_{seqs\,\boldsymbol{x}} \left[\frac{1}{P(\boldsymbol{x})} \sum_i f_{D_u}(x_i)\, t(D_u,M_{u+1})\, e_{M_{u+1}}(x_{i+1})\, b_{M_{u+1}}(x_{i+1}) \right] \\
m_{M_u,I_u} &= \sum_{seqs\,\boldsymbol{x}} \left[\frac{1}{P(\boldsymbol{x})} \sum_i f_{M_u}(x_i)\, t(M_u,I_u)\, q_{x_{i+1}}\, b_{I_u}(x_{i+1}) \right] \\
m_{I_u,I_u} &= \sum_{seqs\,\boldsymbol{x}} \left[\frac{1}{P(\boldsymbol{x})} \sum_i f_{I_u}(x_i)\, t(I_u,I_u)\, q_{x_{i+1}}\, b_{I_u}(x_{i+1}) \right] \\
m_{D_u,I_u} &= \sum_{seqs\,\boldsymbol{x}} \left[\frac{1}{P(\boldsymbol{x})} \sum_i f_{D_u}(x_i)\, t(D_u,I_u)\, q_{x_{i+1}}\, b_{I_u}(x_{i+1}) \right] \\
m_{M_u,D_{u+1}} &= \sum_{seqs\,\boldsymbol{x}} \left[\frac{1}{P(\boldsymbol{x})} \sum_i f_{M_u}(x_{i+1})\, t(M_u,D_{u+1})\, b_{D_{u+1}}(x_{i+1}) \right] \\
m_{I_u,D_{u+1}} &= \sum_{seqs\,\boldsymbol{x}} \left[\frac{1}{P(\boldsymbol{x})} \sum_i f_{I_u}(x_{i+1})\, t(I_u,D_{u+1})\, b_{D_{u+1}}(x_{i+1}) \right] \\
m_{D_u,D_{u+1}} &= \sum_{seqs\,\boldsymbol{x}} \left[\frac{1}{P(\boldsymbol{x})} \sum_i f_{D_u}(x_{i+1})\, t(D_u,D_{u+1})\, b_{D_{u+1}}(x_{i+1}) \right]
\end{aligned}
\qquad \text{(EQ6.35)}
$$

Using similar logic, and this time restricting paths to those that emit a residue of type a at state M_u (that is, the residue emitted can be any residue a in the sequence $\boldsymbol{x}$), we obtain

$$
n_{M_u,a} = \sum_{\text{seqs}\,\boldsymbol{x}} \left[\frac{1}{P(\boldsymbol{x})} \sum_{i|x_i=a} f_{M_u}(x_i) b_{M_u}(x_i) \right] \qquad \text{(EQ6.36)}
$$

where the summation is over all residues of type a in the whole sequence. The insert state emission probabilities can also be fitted with an equivalent equation for I_u.

The process starts with rough estimates of the transition and emission parameters. These values are used to obtain a new set of parameters, using Equations EQ6.35 and EQ6.36 with EQ6.27 or other methods to take account of a lack of data as described in Section 6.1. An improved parameter set should make the emission of the training sequences more likely. The likelihood of observing all the training sequences is given by the product of the $P(\boldsymbol{x})$ for all sequences $\boldsymbol{x}$. Normally the log likelihood is used, which is the sum of the $F_{End}\,(x_L)$ for each sequence. The method is repeatedly used to (hopefully) improve the HMM parameters, until the log likelihood of the training data does not improve any further. After repeating this procedure from several alternative starting points, the best parameters are kept, and the HMM is regarded as parameterized.

Thus far we have not considered weighting the training sequences, even though we might expect some to be more helpful in the parameterization than others. The sequence weighting methods discussed in Section 6.1 are only applicable to aligned sequences and so do not help here, unless the poorly parameterized HMM is used for their "alignment". (Aligning each sequence to the profile HMM cannot be said to create a true multiple alignment, despite the results appearing in a similar form. However, the output could be treated in the same way to obtain sequence weights.) Most profile HMM programs weight training sequences according to their log-odds score. Typically the worst-scoring sequence is assigned a weight of 1, and the others are weighted on a scale that assigns a weight of 0.01–0.001 to the best-scoring sequence. These weights are typically updated during each iteration of the parameterization.

6.3 Aligning Profiles

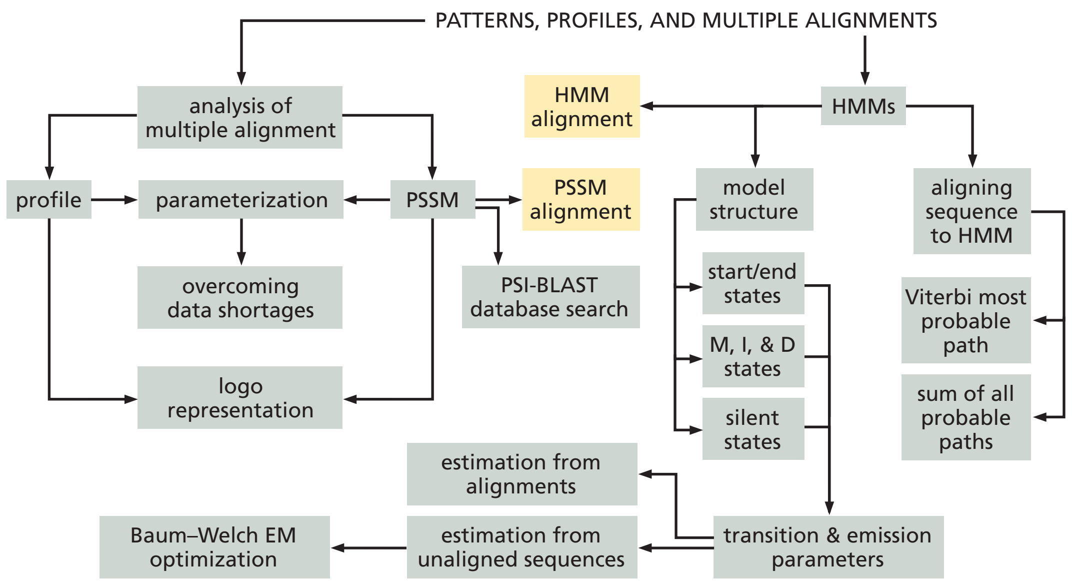
The methods of generating PSSMs and profile HMMs from sequences are now well established, and several databases exist that provide such models for different protein families and subfamilies. It has become common practice to use these models in searching for remote homologs, during which sequences are aligned to them. More recently it has been appreciated that aligning two PSSMs or profile HMMs can be even more effective at identifying remote homologs and evolutionary links between protein families (see Flow Diagram 6.3). (Generally families are thought to have no evolutionary relationship to each other.) As a result several methods have been proposed that can perform such analysis.

Comparing two PSSMs by alignment

The alignment of two PSSMs cannot proceed by a standard alignment technique. Consider the alignment of two columns, one from each PSSM. As neither represents a residue or even a mixture of residues, but both in fact are scores, there is no straightforward way of using them together to generate a score for use in an alignment algorithm. The solution to this problem is to use measures of the similarity between the scores in the two columns.

The program LAMA (Local Alignment of Multiple Alignments) solves one of the easiest formulations of this problem, not allowing any gaps in the alignment of the PSSMs. Suppose the two PSSMs A and B consist of elements $m_{u,a}^{A}$ and $m_{v,a}^{B}$ for residue type a in columns u and v, respectively. Various methods can be used to try

Flow Diagram 6.3
The key concept introduced in this section is that methods have been designed that align two profiles. There is a hierarchy of related protein sequences, so that often two profiles can be aligned to identify features of their common ancestral sequence.


PATTERNS, PROFILES, AND MULTIPLE ALIGNMENTS
analysis of multiple alignment
HMM alignment
HMMs
profile
parameterization
PSSM
PSSM alignment
model structure
aligning sequence to HMM
overcoming data shortages
PSI-BLAST database search
start/end states
Viterbi most probable path
M, I, & D states
logo representation
silent states
sum of all probable paths
estimation from alignments
Baum–Welch EM optimization
estimation from unaligned sequences
transition & emission parameters