

<i>a</i>	<i>b</i>	<i>c</i>	-	<i>a</i>
<i>a</i>	<i>b</i>	<i>a</i>	<i>b</i>	<i>a</i>
<i>a</i>	<i>c</i>	<i>c</i>	<i>b</i>	-
<i>c</i>	<i>b</i>	-	<i>b</i>	<i>c</i>

Figure 1: A multiple alignment that will be used to generate a profile.

	<i>C1</i>	<i>C2</i>	<i>C3</i>	<i>C4</i>	<i>C5</i>
<i>a</i>	.75		.25		.50
<i>b</i>		.75		.75	
<i>c</i>	.25	.25	.50		.25
-			.25	.25	.25

Figure 2: The profile extracted from the previous multiple alignment.

### 0.0.1 Family representations and alignments with profiles

**Definition** Given a multiple alignment of a set of strings, a *profile* for that multiple alignment specifies for each column the *frequency* that each character appears in the column. A profile is sometimes also called a *weight matrix* in the biological literature. Profiles were initially developed and exploited in molecular biology by Gribskov, Eisenberg and co-workers [?, ?, ?].

For example, consider the multiple alignment of four strings shown in Figure 1. The profile derived from it has five columns shown in Figure 2.

Often the values in the profile are converted to *log-odds ratios*. That is, if  $p(y, j)$  is the frequency that character  $y$  appears in column  $j$ , and  $p(y)$  is the frequency that character  $y$  appears anywhere in the multiply aligned sequences, then  $\log p(y, j)/p(y)$  is commonly used as the  $y, j$  profile (or weight matrix) entry. The meaningful part is the ratio. The logarithm is for primarily for convenience and any particular base can be used. For our purposes, when we use the term “profile” we will not need to specify whether the value is a raw frequency or a log-odds ratio.

#### 0.0.1.1 Aligning a string to a profile

Given a profile  $\mathcal{P}$  and a new string  $S$ , one often asks how well  $S$ , or some substring of  $S$ , fits the profile  $\mathcal{P}$ . Since a space is a legal character of a profile, a fit of  $S$  to  $\mathcal{P}$  should also allow the insertion of spaces into  $S$ , and hence the question of how well  $S$  fits  $\mathcal{P}$  is naturally formalized as an easy generalization of pure string alignment. Consider a string  $C$  of profile column positions,

<i>a</i>	<i>a</i>	<i>b</i>		<i>b</i>	<i>c</i>
1		2	3	4	5

Figure 3: An alignment of string *aabbc* to the column positions of the previous alignment.

and then align *S* to *C* by inserting spaces into *S* and *C* etc., as in pure string alignment. For example, an alignment of the string *aabbc* to the previous profile is shown in Figure 3. The key issues are how to score an alignment of a string to a profile, and given a scoring scheme, how to compute an optimal alignment.

### How to score a string/profile alignment

Assume an alphabet-weight scoring scheme for *pure* string alignment is known. Using that pure string-scoring scheme, the common approach to scoring an alignment of a character in *S* to a column in *C* is to consider the string character aligned to every character in the column and then compute a weighted sum of scores based on the frequency of the characters in the column. The score for the full alignment is the sum of the column scores. For example, suppose the alphabet-weight scheme gives a score of 2 to an *a,a* alignment, a score of -1 to an *a,b* or an alignment of *a* with a space, and a score of -3 to an *a,c* alignment. Then, the first column of the previous alignment of string *aabbc* to *C* contributes a score of  $0.75 \times 2 - 0.25 \times 3$  to the overall alignment score. The second column contributes a score of -1, since it aligns character *a* to a space.

### How to optimally align a string to a profile

We view alignment in terms of maximizing similarity and compute the optimal alignment by dynamic programming. The details are very similar to the way pure string similarity is computed (see Section ??). However, the notation is a little more involved. Recall that for two characters *x* and *y*,  $s(x, y)$  denotes the alphabet-weight value assigned to aligning *x* with *y* in the pure string alignment problem.

**Definition** For a character *y* and column *j*, let  $p(y, j)$  be the frequency that character *y* appears in column *j* of the profile, and let  $S(x, j)$  denote  $\sum_y [s(x, y) \times p(y, j)]$ , the score for aligning *x* with column *j*.

**Definition** Let  $V(i, j)$  denote the value of the optimal alignment of substring  $S[1..i]$  with the first *j* columns of *C*.

With that notation, the recurrences for computing optimal string to profile alignment are:

$$V(0, j) = \sum_{k \leq j} s(-, k),$$

and

$$V(i, 0) = \sum_{k \leq i} s(S_1(k), -).$$

For  $i$  and  $j$  both strictly positive, the general recurrence is:

$$V(i, j) = \max[V(i-1, j-1) + S(S_1(i), j), V(i-1, j) + s(S_1(i), -), V(i, j-1) + S(-, j)].$$

The correctness of these recurrences is immediate and left to the reader. As usual in dynamic programming, the actual alignment is obtained during a backtrack phase, after the recurrences have been evaluated. The time to evaluate the recurrences is  $O(\sigma nm)$ , where  $n$  is the length of  $S$  and  $\sigma$  is the size of the alphabet. Verification of the time bound is also left to the reader. We see from this that aligning a string to a profile is not much more costly than aligning two strings. A specific use for aligning strings to profiles is discussed in Section ???. See [?] for a more in-depth discussion of profiles in computational biology.

### Profile to profile alignment

Another way that profiles are used is to compare one protein set to another. In that case, the profile for one set is compared to the profile of the other. Alignments of this kind are sometimes used as a subtask in certain multiple alignment methods (see Section ??). It is straightforward to formalize optimal profile to profile alignment and to obtain the recurrences to compute it. That is again left as an exercise.