Chapter 2

Theoretical Background

In this chapter we provide a concise introduction to the two pillars of annotated alignments: the concept of *profiles* for representing and searching transcription factor binding sites (TFBSs) and the concept of *alignments* for extracting conserved regions. Indeed, we view annotated alignments simply as *alignments* with parts assigned to putative *conserved TFBSs*. Section 2.1 deals with the first aspect wherein we formalize the path from experimental data to position-specific scoring matrices (PSSMs) and describe their applicability for searching putative TFBSs (Section 2.1.3). In Section 2.2, we review standard alignment concepts including algorithms and choice of scoring schemes. At the onset though, we begin with notations that will be used throughout the thesis.

Terms and notations Let Σ be a finite alphabet, in our context consisting of the DNA nucleotides $\mathrm{A}, \mathrm{C}, \mathrm{G}$, and T. Hence, $|\Sigma| = 4$. A string $\mathbf{u}$ of length p from this alphabet is represented as $\mathbf{u} = (u_1 u_2 \dots u_p)$ where each $u_i \in \Sigma$. A *substring* $\mathbf{u}_{ij} = (u_i \dots u_j)$ denotes the contiguous string of letters from i to j of $\mathbf{u}$, where $1 \leq i \leq j \leq p$. Hence for $i = 1$, the resulting substring is $\mathbf{u}_{1j}$ consisting of the first j letters of $\mathbf{u}$ and referred to as the j^{th} *prefix* of $\mathbf{u}$.

We describe a *background* probability distribution on Σ given by the row vector $\pi = \pi(j)$, $j \in \Sigma$, with non-negative entries that sum to one. This distribution represents the genomic properties, like GC-content, of the sequences under consideration. Unless otherwise stated it is simply the uniform distribution with $\pi(j) = 0.25$ for all j.

For the sake of simplicity, we restrict the upcoming discussions on profiles to one transcription factor $\mathbf{F}$, assumed to bind DNA segments of length l.

2.1 Binding site profiles

In Section 1.2.2 we mentioned that a profile is a probabilistic model for representing a collection of binding sites. In the following, we formally address the two objectives of a binding site model:

1. to sufficiently represent the properties of the experimentally verified sites for both *comparative* and *generative* purposes and,

2. to be straightforwardly applicable in *predictive* methods.

Starting with the estimation of TFBS profiles using experimental sites, we proceed then to a discussion on the first objective. Following that we describe the construction of scores from probabilities and use them to address the second objective. Much of what follows is on the lines of well-established methods and we refer the interested reader to articles by Staden [185], McLachlan [129], Tatusov *et al.* [192], Hertz and Stormo [80] and Rahmann *et al.* [158] for details.

2.1.1 From sites to probabilities

The experimental sites of the factor **F** can be aligned to generate counts of observed nucleotides at each position. These are used to formulate a *position-specific count matrix* PSCM, with entries $\texttt{PSCM}_{ij}$ $(i = 1, \ldots, l; j \in \Sigma)$ representing the number of occurrences of nucleotide j at position i in the alignment. To prevent rejecting the possibility of observing a previously un-observed nucleotide at a position, the count matrix is regularized: a pseudocount is added to each entry. This also prevents singularities in the scores calculated later.

The probability of observing a nucleotide j at position i is then estimated by taking the ratio of the regularized $\texttt{PSCM}_{ij}$ to the total number of experimental sites, modified according to the regularization. This gives rise to the *signal profile* P with $\mathrm{P}^i(j)$ being the probability of observing nucleotide j at position i. To maintain consistency in notation, we name the matrix composed of these probabilities as the *position-specific probability matrix* PSPM with $\texttt{PSPM}_{ij} = \mathrm{P}^i(j)$.

Generative Use Equipped with a probabilistic description of the binding sites, it is now possible to sample new instances. A random instance of the signal profile can thus be generated simply by sampling letters at each position i using P^i. For a string $\mathbf{u}$, the probability that it is generated by the signal profile P is:

$$\mathbb{P}_{\mathrm{P}}(\mathbf{u}) = \prod_{i=1}^{l} \mathrm{P}^i(\mathbf{u}_i) \tag{2.1}$$

Note the underlying assumption here that each position is independent of the others. That is, observing a nucleotide at one position does not influence the probability of observing one at another position yielding the product in Equation(2.1). For computational tractability where precision is often a limiting criterion, usually it is preferable to work with integral scores instead of this product of probabilities. This leads us to the concept of *position-specific scoring matrix* (PSSM) or *position-specific weight matrix* (PWM).

2.1.2 From probabilities to scores

Our goal is to decide whether a window W of length l is a site (ie. sampled from P) or a non-site. To this end, a background model is required for the specification of "non-site" properties. Although Markov models from zero to third order have been employed for background modeling, the uniform distribution defined in Section 2 suffices for our purpose here. We represent the background profile by $\Pi = (\Pi_i) = \pi$, $(i = 1, \ldots, l)$; a non-site is assumed to be sampled from this profile.

pos	A	C	G	T
1	0	0	0	16
2	16	0	0	0
3	0	0	0	16
4	0	16	0	0
5	4	0	3	9
6	7	6	0	3

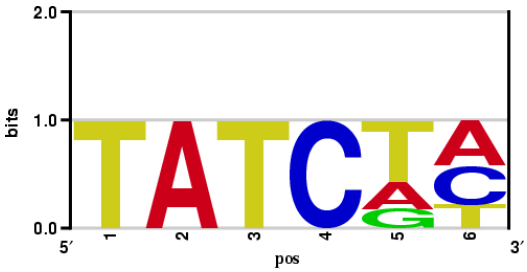
(a)

pos	A	C	G	T
1	0.0040	0.0032	0.0004	0.9922
2	0.9932	0.0021	0.0002	0.0043
3	0.0040	0.0032	0.0004	0.9922
4	0.0023	0.9933	0.0002	0.0038
5	0.2511	0.0083	0.1818	0.5587
6	0.4216	0.3601	0.0037	0.2150

(b)

pos	A	C	G	T
1	-82	-87	-126	28
2	28	-95	-135	-81
3	-82	-87	-126	28
4	-93	28	-137	-84
5	0	-68	-6	16
6	10	7	-87	-3

(c)



(d)

Figure 2.1: *Formulation of the scoring matrix.* **a)** An example position-specific count matrix retrieved from TRANSFAC (Identifier (Id) M00142). The experimentally verified sites are aligned to retrieve the frequency or count matrix depicted on left. **b)** Regularized position-specific probability matrix or profile for the above given count matrix. The regularization techniques to remove zero entries follow Rahmann *et al.* [158]. **c)** Finally, the position-specific scoring matrix for the same profile. Care is taken to round and scale the scores to yield integral entries. Note the difference between the initial counts and the final scores is the consideration of a background distribution. **d)** Sequence logo of the consensus motif for this profile. Vertical height of the individual letters represent the information content, while horizontal axis corresponds to the position in a motif.

The *score* of a string $\mathbf{u}$ is calculated as a scaled log-odds ratio of the probabilities of observing $\mathbf{u}$ under the signal profile P and the background profile Π:

$$\texttt{Score}(\mathbf{u}) = \mathrm{c} \log \left(\frac{\mathbb{P}_{\mathrm{P}}(\mathbf{u})}{\mathbb{P}_{\Pi}(\mathbf{u})} \right) = \mathrm{c} \sum_{i=1}^{l} \log \left(\frac{\mathrm{P}^i(u_i)}{\pi(u_i)} \right) \tag{2.2}$$

where c is the scaling constant. This implies that comparing the probabilities of observing a nucleotide j at a position i under both the models gives rise to the elements of the scoring matrix, or $\texttt{PSSM}_{ij} = \mathrm{c} \log(\mathrm{P}^i(j))/(\pi(j))$. In this way, each nucleotide in the string is scored yielding the Equation 2.2. A string with a strongly positive score is assumed to be an instance of the signal profile. This can be seen from Equation 2.2 which then implies that the probability that the string is a signal instance is higher. Figure 2.1 illustrates the process of arriving at a position specific score matrix from the experimental data using an example.

We are now in a position to address the usability of the above-derived PSSM in predictive applications. In the following, a TFBS *hit* is a window that is predicted as a putative binding site. Consider a sequence of length $n + l - 1$ with n overlapping windows of length l each. Our objective is to identify putative binding sites of the factor on this sequence.

Predictive Use Using the PSSM as a scoring tool, each l-length window in the sequence is scored to check for putative hits. More formally, we can define the binding site searching problem as:

> *Using a given* PSSM*, find all positions j in a sequence where a starting window $W(j)$ of length l scores greater than a pre-defined threshold t.*

The choice of the threshold is crucial: it should be high enough to limit false predictions (background sites that score higher than t) and still not too high to be attainable by true sites. Thus, a derivation of t that **a)** takes background sequence composition into account, and **b)** is independent of the considered TFBS profile, is required. To this end, it is preferred to use a desired *p-value* or *power* setting to calculate the score cutoff. Although numerous other techniques for threshold derivation have been proposed, using a p-value or power cutoff is by far the most statistically well-founded ([158, 98], etc.).

For use of profiles for predictive purposes, we therefore need to describe how we arrive at a score threshold. We outline the procedure to derive the cutoff in a *testing* framework and use it to re-phrase the binding site search problem. Again for computational ease, we assume that all calculations take place in the integer set. To this end, original PSSM scores are assumed to be appropriately scaled and rounded to obtain an integer range. For further insight into issues involved in regularization, scaling and rounding as well as for details in the upcoming discussion, we refer the reader to the work of Rahmann *et al.* [158].

2.1.3 Searching for binding sites - a statistical testing framework

Let $X = \texttt{Score}(W)$, the score of a window W, be a discrete random variable which takes values from a finite set of integers Γ. Let $\mathbb{P}_{\mathrm{P}}$ and $\mathbb{P}_{\Pi}$ represent the two probability distributions associated with X. The former gives the probability that the window W is generated

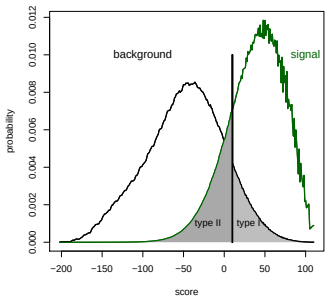


Figure 2.2: *An example of score distribution curves.* Under the signal (green) and background (background) model, for the binding site profile of the factor HOXA3 (TRANSFAC ID M00395). The cutoff can be varied and the corresponding type I and type II errors measured.

by the signal profile P and the latter the probability that W is generated by the background profile Π. We wish to distinguish between the two using statistical inference.

Let H_0 be the *null hypothesis* that W is generated by the background profile. That is, X is distributed according to Π. The *alternative hypothesis* H_1 is that it is generated by the signal profile, implying X is distributed according to P. Taking the likelihood ratio $\mathrm{LLR}_{\mathrm{P},\Pi}(W)$ of observing W under both the models as the *test statistic*, H_0 is rejected iff $\mathrm{LLR}_{\mathrm{P},\Pi}(W) \geq t$, for a pre-determined threshold t. That is,

$$\mathrm{LLR}_{\mathrm{P},\Pi}(W) := \log\left(\frac{\mathbb{P}_{\mathrm{P}}(W)}{\mathbb{P}_{\Pi}(W)}\right) = \sum_{i=1}^{l} \log\left(\frac{\mathrm{P}^i(W_i)}{\pi(W_i)}\right) \geq t$$

Two types of errors are possible here. If H_0 is rejected when it is true, then it gives rise to a *type I error* or a *false positive* (FP). This implies a background window that scores greater than t and hence is wrongly predicted as signal window. On the other hand, if H_0 is accepted when it is false, then we have a *type II error* or *false negative* (FN). This is the case when a known site scores below the threshold and is missed. Correct predictions in each case are referred to as *true positives* (TP) and *true negatives* (TN), respectively. In Fig 2.2, we show two overlapping distributions, presumably representing the background and the signal. It can be seen that the above-defined terms correspond to different areas under the curves depending on the threshold, as highlighted.

For our problem of searching for putative binding sites in a sequence of length $n + l - 1$, this test is performed for n overlapping windows. Since the windows are not independent, calculating the exact type I and type II errors for the *whole* sequence is not trivial; approximations are required. In their work, Rahmann *et al* [158] provide recipes to calculate these sequence level errors using independence assumptions. We present their results next.

Adopting the notations of the authors we call:

- $\alpha(t)$: as the type I *window* error probability. It represents the probability of observing a score of at least t, given that a window is generated by the background model:

$$\alpha(t) := \mathbb{P}_{\Pi}(X \geq t) \tag{2.3}$$

  In other words, the area under the background distribution curve where the scores exceed the threshold t.

- $\alpha_n(t)$: as the type I *sequence* error probability. It is the probability that *at least* one out of n consecutive overlapping windows scores greater than a pre-defined threshold t, given that the sequence is generated by the background model. If $X(i)$ is the random variable describing the score of the i^{th} window $W(i)$, this means:

$$\alpha_n(t) := \mathbb{P}_\Pi(\max_{i\in[1,n]} X(i) \geq t) \tag{2.4}$$

  There are two points worth mentioning here. First, a sequence of n windows has a higher probability of error than a single window, and so $\alpha_n(t)$ is much higher than $\alpha(t)$. And second, the type I sequence error probability is the same as the *sequence p-value*, the likelihood of exceeding the threshold on a background generated sequence. Hence, if a desired p-value level α is given, it should be possible to calculate the score cutoff t such that $\alpha_n(t) < \alpha$.

- $\beta(t)$: as the type II *window* error probability. This is the probability that a score below t is observed given that a window is generated by the signal model:

$$\beta(t) := \mathbb{P}_\mathrm{P}(X < t) \tag{2.5}$$

  That is, the area under the signal distribution curve where the scores are below the threshold t.

- $\beta_m(t)$: as the *m-instance* type II error probability. It is impractical to say that the whole sequence is generated by the same signal profile. Hence, we consider m independent signal instances and $\beta_m(t)$ represents the probability that *at least* one out of m instances scores less than t, given that all are generated by the signal model. That is,

$$\beta_m(t) := \mathbb{P}_\mathrm{P}(X(i) < t \text{ for at least one } i) \tag{2.6}$$

  Note that the *power* of the m-instance test is then defined as $1-\beta_m(t)$, the probability of recovering all instances given that they are generated by the signal model. Again, given a desired type II error level β the score cutoff t should be such that $\beta_m(t) < \beta$.

By assuming that consecutive windows are independent of each other, the sequence level type I error can be calculated using the window level type I error and is simply given by:

$$\begin{aligned}\alpha_n(t) &\approx 1 - (1 - \alpha(t))^n \\ &\approx 1 - \exp(-n\alpha(t))\end{aligned} \tag{2.7}$$

Similarly, assuming that each of the true instances are independent of the others, $\beta_m(t)$ can be calculated as:

$$\begin{aligned}\beta_m(t) &= 1 - (1 - \beta(t))^m \\ &\approx 1 - \exp(-m\beta(t)) &(2.8)\\ &\approx m\beta(t), \qquad \text{if} \quad m\beta(t) \ll 1 &(2.9)\end{aligned}$$

Hence, both the sequence level errors can be approximately calculated using their window level counterparts. Each of the window errors themselves can be calculated using the score distributions under the respective models. Therefore, if the score distributions are known then it is possible to calculate the score cutoff that corresponds to the desired error levels. The choices are the ones corresponding to a fixed type I error level (*type I cutoff*), a fixed

type II error level (*type II cutoff*) and when both errors are equal (*balanced cutoff*)([158]). Sometimes, minimizing the sum of the type I and type II errors is also considered ([98]).

At this stage, it is now essential to address the issue of efficiently calculating the score distributions under both the models. Initial works of McLachlan [129], Staden [185] and Tatusov *et al.* [192] describe methods to compute recursively the probability of observing a score given a model. Wu *et al.* [220], Rahmann *et al.* [158], Beckstette *et al.* [16] and Malde *et al.* [118] have since then presented variations of the basic recursive approach to calculate exact or approximate score distributions. As in our other expositions, we stick to the method prescribed by Rahmann *et al.* [158] who describe an approach for calculating exact distributions in $O(l|\Gamma||\Sigma|)$ time. We restrain from providing detailed formalism (and refer the interested reader to the article [158]), instead after a succinct overview of the idea, we use a simple hypothetical example to walk through the procedure.

For the following, we assume that the scores are properly discretized with the integral range of scores given by Γ and that the score distribution is being calculated for the signal profile P (though also valid for the background profile Π). Let us again use the random variable $X = \texttt{Score}(W)$ for the score of a window. The random variable X_i represents the score *at* the i^{th} position and X^i represents the score *till* the i^{th} position in the window.

Calculation of Score distributions To start with, there are a few observations about the above-defined scores that should be highlighted here. First, since the positions in a window W are assumed independent, the total score of the window is the sum of the scores at the individual positions X_i. Second, using the position-specific nucleotide distribution given by P, it is straightforward to calculate the probability of observing a score at a position i:

$$\mathbb{P}(X_i = x) = \sum_{\substack{a \in \Sigma, \\ \texttt{PSSM}_{ia} = x}} \mathrm{P}^i(a) \qquad \forall \quad x \in \Gamma \tag{2.10}$$

Using P, the probabilities for possible scores at the first position ($f^1(x)$) can be calculated, leading to the initialization step. Hence, it is possible to calculate the probability of observing a score till the position $(i+1)$ recursively by combining the accumulated values till the i^{th} position and the value at the position $(i+1)$. In other words, by taking the convolution of the two values. If we denote $f^{i+1}(x)$ as the probability to observe a score x till the position $(i+1)$, then this means:

$$\begin{aligned} f^{i+1}(x) &= \sum_{x' \in \Gamma} \mathbb{P}(X^i = x') \cdot \mathbb{P}(X_{i+1} = x - x') \\ &= \sum_{x' \in \Gamma} f^i(x') \cdot \sum_{\substack{a \in \Sigma, \\ \texttt{PSSM}_{i+1,a} = x - x'}} \mathrm{P}^{i+1}(a) \end{aligned} \tag{2.11}$$

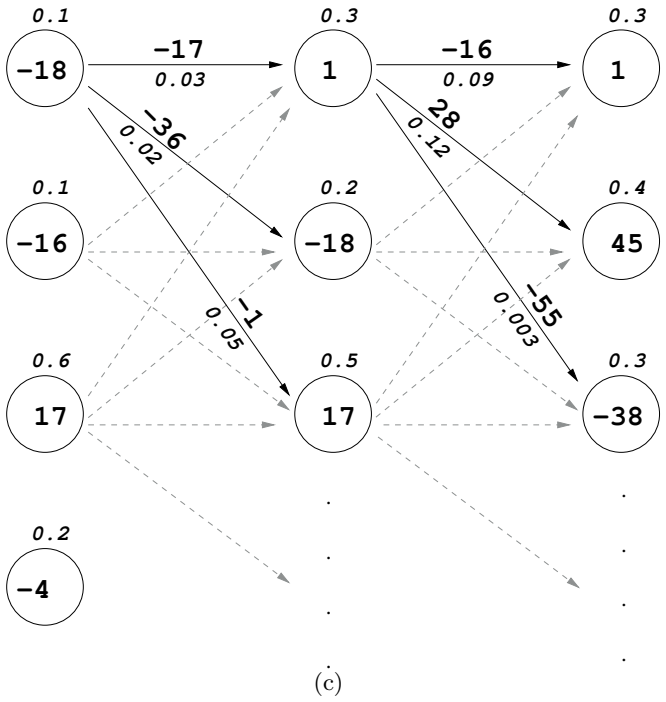
Finally, the score distribution for the window length is simply $f^l(x)$. Hence, the basic idea involves two main steps: consider all scores that can be generated using the PSSM and use the profile (background or signal) to find the probability of observing the scores under the respective model as described above. This can be viewed as tracing all possible paths in the scoring matrix. The probability of a score depends solely on the probabilities of observing the nucleotides that contribute that score, irrespective of the nucleotides themselves. In Figure 2.3, a simple hypothetical example is presented to elucidate the method.

	A	C	G	T
1	0.1	0.1	0.6	0.2
2	0.3	0.1	0.5	0.1
3	0.4	0.3	0.1	0.2
4	0.2	0.2	0.5	0.1
5	0.2	0.3	0.3	0.2

(a)

	A	C	G	T
1	-18	-16	17	-4
2	1	-18	17	-18
3	45	1	-38	-38
4	-16	-16	27	-16
5	-18	27	2	-18

(b)



(c)

Figure 2.3: *Example.* For the score distribution calculation under the signal profile given by **a)** with the scoring matrix shown in **b)**. Each column of circles in **c)** represents a position in the scoring matrix, the values inside the circles representing the scores of observing each nucleotide at this position in the profile. The values on top of the circle correspond to the probability of observing the corresponding nucleotide at the position ($\mathrm{A}, \mathrm{C}, \mathrm{G}$ and T from top to bottom). Each arrow has an associated score (above) that is calculated as the sum of the scores of nucleotide at position i to another at position j and an associated probability (below) obtained as a product of the individual probabilities.

2.2 Standard pairwise alignments

In Section 1.4 (Chapter 1) we broadly discussed sequence comparison strategies, focusing on dynamic programming based alignment approaches. We provided a succinct description of the alignment model and associated scoring scheme which forms the basis of the present work. In this section, we narrow our focus to formally discuss the ingredients of standard alignments – the dynamic programming algorithm and the standard parameter choice. In Section 2.2.3, we introduce three probabilistic models for modeling evolutionary processes such as nucleotide substitutions. It is through such models that the values of the scores introduced in the first half of the section are calculated.

2.2.1 Dynamic Programming Algorithm

The dynamic programming approach consists of three elements: recursion rule with proper initialization conditions, tabular computation and finally a traceback procedure to generate the alignment. The idea is to break a bigger problem into several smaller component problems and use the results of a predecessor problem for successive calculations. For alignments, this involves calculating the best alignment up to a certain point in the two sequences and extending it to get the overall optimal. An intuitive way of representing an alignment is as a path through an $(m+1) \times (n+1)$ dynamic programming matrix $\mathtt{M}$ with the row and column indices given by x and y. From each point in the matrix three kinds of edges can be drawn – a diagonal edge implying an aligned pair, a horizontal edge a deletion in x and a vertical edge an insertion in x. The matrix is initialized here as following:

$$\mathtt{M}(i,0) = \mathtt{M}(0,j) = 0,\ \forall\ 0 \leq i \leq m, 0 \leq j \leq n \tag{2.12}$$

Linear Gap Costs At every point (i,j), the algorithm chooses the optimal alignment between the substrings $x_{1i} = (x_1 \dots x_i)$ and $y_{1j} = (y_1 \dots y_j)$. There are three possible scenarios for the optimal alignment ending at x_i, y_j:

- either x_i is aligned to y_j, that is the alignment ends with a substitution. Here, the substitution score $\mathtt{s}(x_i, y_j)$ needs to be added to the optimal score $\mathtt{M}(i-1, j-1)$ to get the new score at (i,j).
- or, x_i is aligned to a gap in y (a deletion), in which case a gap cost is subtracted from $\mathtt{M}(i-1,j)$ to yield the score at (i,j).
- or y_j is aligned to a gap in x (an insertion) which is symmetric to the previous case.

Comparing between the three possibilities gives rise to the recursion rule for global alignments (in the NW algorithm, dynamic programming matrix shown in Figure 2.4). Extending to local alignments requires allowing the additional possibility to take the value 0 if the maximum score less than 0, which leads to:

$$\mathtt{M}(i,j) = \max \begin{cases} 0, \\ \mathtt{M}(i-1,j-1) + \mathtt{s}(x_i, y_j), \\ \mathtt{M}(i-1,j) - \mathrm{g}, \\ \mathtt{M}(i,j-1) - \mathrm{g} \end{cases} \tag{2.13}$$

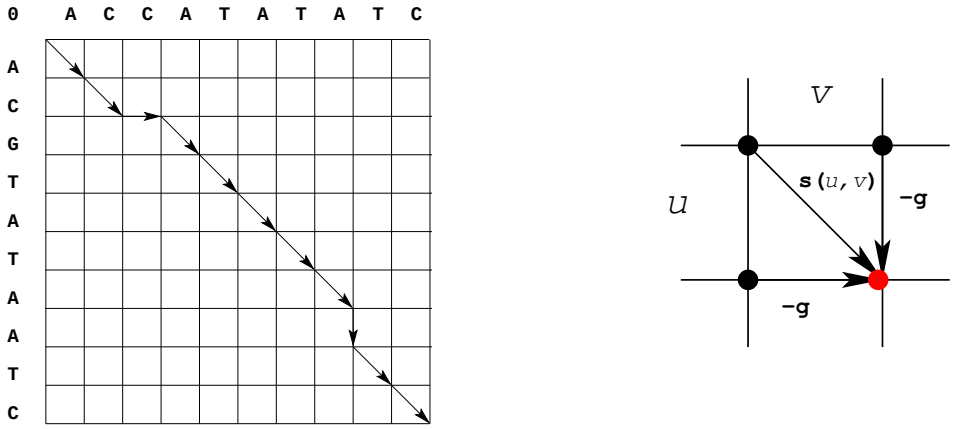


Figure 2.4: *Dynamic Programming Matrix.* In the left figure, the standard dynamic programming matrix M for global alignment algorithm for the two sequences considered in the example of Figure 1.3 is shown. At each point in the matrix, the scores of three predecessor points are compared (right). A diagonal arrow corresponds to a substitution, a vertical arrow to a deletion and the horizontal arrow to an insertion. Note, that the recursion in the Smith-Waterman local alignment algorithm still compares the three predecessor points *and* zero to give alignments that need not cover the whole length of the sequences.

That is, $\mathtt{M}(i,j)$ is the maximum of 0 or the best score of an alignment ending at x_i and y_j. If $\mathtt{M}(i,j)$ is derived from $(i-1,j-1)$, then it is called a substitution (S). If it is derived from $(i,j-1)$, then it is called an insertion (I) and it is a deletion (D), if $\mathtt{M}(i,j)$ is derived from $(i-1,j)$. Also, whenever the value 0 is taken, a new alignment is started. Thus, the score of a grid point can be calculated using the scores of three predecessor points (Fig. 2.4).

Finally, the alignment is found by tracing back from the point (i,j) with the maximum score $\mathtt{M}(i,j) = \mathtt{M}_{\max}$, where $\mathtt{M}_{\max} = \max_{k,l} \mathtt{M}(k,l)$, $\forall\ 0 \le k \le m$, $0 \le l \le n$.

One way to compute the optimal alignment is to store back-pointers at each step during the tabular computation. This implies storing the whole matrix leading to a space requirement of $O(n^2)$ for a pair of sequences of length n each. Again, at each step constant number of operations need to be performed yielding an $O(n^2)$ time complexity too.

Affine Gap Costs In our discussion about scoring schemes in Section 1.4.2, we mentioned that in reality it is less likely that each gap is a separate mutational event. Usually gaps are introduced in spurts, and hence a gap cost system that considers longer gaps and penalizes them less severely as compared to the linear gap cost presented before, is required. For global alignments, a first effort in this direction was made by Waterman *et al.* [213]. They considered multiple contiguous indel events and introduced a *general* gap cost function $\nu(k)$ dependent on the length k of the gap. Now a prefix in one sequence can be aligned to an extended gap in the second, and hence all possible prefixes need to be compared to extend the optimal alignment at a certain point:

$$\mathtt{M}(i,j) = \max \begin{cases} \mathtt{M}(i-1,j-1) + \mathtt{s}(x_i, y_j), & \\ \mathtt{M}(i-k,j) - \nu(k) & k = 0,\ldots,(i-1) \\ \mathtt{M}(i,j-k) - \nu(k) & k = 0,\ldots,(j-1) \end{cases} \tag{2.14}$$

As is usual, *allowing flexibility increases complexity.* Since at each step we have to look at $(i+j+1)$ positions, the complexity now increases to $O(n^3)$, making the approach infeasible

for realistic computations. A compromise between biological reality and computational feasibility was reached when Gotoh [70] proposed an $O(n^2)$ algorithm for a special case of the above-mentioned general gap cost system, namely the *affine gaps*.

The affine gap case considers two gap costs: a *gap open* cost g_o that penalizes opening a gap and is generally high, and a *gap extension* cost g_e which penalizes extending an already existing gap by a constant value which is usually lower than g_o. A gap of length k is penalized as:

$$g_{\text{total}} = g_o + k \times g_e \tag{2.15}$$

Hence, in the affine gap case the algorithm is encouraged to find continuous stretches of matches and avoids opening a gap often. Once the gap is opened though, it is easier to extend it rather than start a new one. This also prevents the algorithm from generating an unrealistic alignment with all possible "align-able" pairs aligned with small gaps filled in-between.

The local alignment version of the algorithm presented by Gotoh [70] is as follows. Instead of calculating the two maxima directly in $\mathtt{M}$, two new matrices D and I which store the scores of the best alignments ending with a gap in either sequence, are introduced. Finally, the three possibilities are compared to 0 to stop an alignment when the score is less than 0. The modified recursion rules are:

$$\mathtt{M}(i,j) = \max \begin{cases} 0, \\ \mathtt{M}(i-1,j-1) + \mathtt{s}(x_i, y_j), \\ \mathrm{D}(i-1,j-1) + \mathtt{s}(x_i, y_j), \\ \mathrm{I}(i-1,j-1) + \mathtt{s}(x_i, y_j) \end{cases} \tag{2.16}$$

where the D and I entries are calculated as:

$$\mathrm{D}(i,j) = \max \begin{cases} \mathtt{M}(i-1,j) - g_o, \\ \mathrm{D}(i-1,j) - g_e \end{cases} \tag{2.17}$$

$$\mathrm{I}(i,j) = \max \begin{cases} \mathtt{M}(i,j-1) - g_o, \\ \mathrm{I}(i,j-1) - g_e \end{cases} \tag{2.18}$$

Finally, analogous to the linear case, the alignment is found by tracing back from the point (i,j) with the maximum score at $\max_{k,l} \mathtt{M}(k,l)$. For details on variations of the basic dynamic programming algorithm, gap cost functions and other issues involved in sequence comparisons, the reader is referred to introductory textbooks by Waterman [210] and Durbin *et al.* [52].

2.2.2 Choice of Score Parameters

The quality of an alignment hinges on the appropriate choice of score parameters. As mentioned previously (Section 1.4.2), a useful scoring scheme is one that brings the biologically correct alignment to the forefront by scoring it extremally. In the following, we will describe how substitution scores are usually derived and their interpretation in terms of the underlying evolutionary model. We will also describe issues involved in choosing the correct gap penalties and the common scoring matrix-gap penalty combinations used in standard applications.

Substitution Scoring Matrix The choice of the substitution scores should allow one to distinguish between nucleotide pairs that arise just by chance in an alignment and those that are truly *related*. Let $\phi = (\phi(u, v))$, $u, v \in \Sigma$, be a matrix giving the probability that an aligned pair (u, v) is related. If we have at our disposal a set of pairs of homologous sequences whose true alignments are known, then ϕ can be derived empirically by considering the frequencies of observing (u, v) as an aligned pair in the true alignments (as is done in the famous BLOSUM matrices [78]). The other (and older) approach is to approximate evolutionary rates and probabilities of particular amino acid mutations to yield ϕ (as is the case with the original PAM matrices [47]). Simply put, the idea is to relate ϕ to an evolutionary process via a suitable probabilistic model and then compare it to a background model, which we define next.

The background model corresponds to the other possibility that (u, v) are at an aligned position only by chance – an independent observation of u and v in the respective sequences. Using the background distribution introduced in Section 2, this yields the value $\pi(u)\pi(v)$ as the probability of their being observed as an aligned pair.

The substitution scores are now interpreted simply as log-likelihood ratios comparing the two possibilities. Keeping all score calculations in the set of integers, the substitution scoring matrix $\mathtt{s}$ can be defined as:

$$\mathtt{s}(u, v) := \mathrm{c}_{\mathtt{s}} \log \left(\frac{\phi(u, v)}{\pi(u)\pi(v)} \right) \tag{2.19}$$

The scaling constant c_s is chosen to ensure sufficient precision of the log-odds ratios. It is to be emphasized here that both the substitution scores and the gap costs (as well as any new score parameters considered) should be treated consistently with respect to this scaling.

The estimation of ϕ as well as π is not straightforward. Many evolutionary models ([90, 101, 62, 76, 193]) have been proposed that model the dependencies between two sequences and provide an estimate for ϕ. This of course is advantageous since the scores then are directly connected to a possible evolutionary substitution process. We will discuss three such models in Section 2.2.3.

As a side note, it should be pointed out here that given an arbitrary substitution matrix, it is possible to retrace the path to the evolutionary probabilities given by ϕ. This can be seen from the equation 2.19 which on re-writing yields:

$$\phi(u, v) = \pi(u)\pi(v)e^{\mathtt{s}(u,v)/\mathrm{c}_{\mathtt{s}}} \tag{2.20}$$

and since ϕ is a probability distribution, therefore:

$$\sum_{u,v \in \Sigma} \phi(u, v) = 1 \tag{2.21}$$

Thus, given $\mathtt{s}$ and a background distribution π, it is possible to retrieve $\mathrm{c}_{\mathtt{s}}$ and the substitution probabilities of the underlying evolutionary model. Indeed, in their works Karlin and Altschul ([95, 5, 94]) showed that any arbitrary substitution scoring matrix can be rewritten as a log-odds matrix, with a proportionality constant, that describes the "target" distribution ϕ.

Commonly used substitution matrices include the PAM (Accepted Point Mutation) family of matrices [47] and the BLOSUM (BLOcks SUbstitution Matrices) family of matrices [78]. For human-mouse comparisons the HOXD70 matrix [42] has been shown to be effective for DNA sequences and is used as the standard scoring matrix for the BLASTZ program [176].

Gap Costs While the derivation of the substitution scores is more or less concrete, the estimation of an appropriate gap cost poses considerable challenge. Analogous to substitution scores, gap costs also need to reflect the evolutionary distance between a pair of sequences; the closer the sequences, the higher the gap costs. In their review article, Vingron and Waterman [203] present the influence of the gap cost choice on alignment scores. They showed that alignment scores have a phase transition from linear to logarithmic regime as the gap costs increase. Linear phase is uninteresting since it corresponds to additional gaps being inserted (lower gap costs) to align more matches. This yields alignments which are long purely because of more but scattered matches. To retrieve biologically relevant alignments, hence the gap costs should yield alignment scores in the logarithmic phase.

Despite the immense research [95, 136, 203, 152, 153, 160] there is still no strict prescription for the selection of gap penalties. In an empirical study on protein sequences, Reese and Pearson [160] showed that while the gap opening cost changes as a function of the divergence between two sequences, the gap extension cost does not. Hence, usually the gap opening cost is chosen taking evolutionary distance, alignment algorithm and substitution scores into account and the gap extension cost is taken as 10% of the gap opening costs.

In practice, most applications for sequence alignment provide the user a selection of substitution matrix-gap cost combinations. For example, the BLASTZ alignment program by default uses the HOXD70 scoring matrix with gap costs of -400 and -30, respectively. The emboss suite of alignment programs has the default scoring matrix for DNA with match, mismatch and gap costs given by $5, -4, 10$ and 0.5, respectively. For extracting short highly conserved regions in human and mouse, Dieterich *et al.* [50] derived a substitution scoring matrix based on a Kimura model [101] normalized to a distance of 1 PAM and propose gap cost settings of 2189 and 99.5, respectively.

2.2.3 Evolutionary Models

Through alignments, we wish to decide whether a pair of sequences from different organisms evolved from a common ancestral sequence. At each position in the ancestral sequence, an evolutionary process consisting of *mutational events* like substitutions, insertions, deletions, etc. occurs. The intuition is that during the passage of time, an ancestral sequence accumulated such changes to yield non-identical but similar sequences whose common elements reflect conservation. Evolutionary models are a way to model this process of evolutionary change probabilistically. In the following, we first provide a brief overview of the basic concepts and then describe three evolutionary models of interest to us. Although sophisticated approaches that model indels have been proposed [193], our focus here is restricted to modeling substitutions only.

Basics We begin with the assumption that the evolutionary process at each *site* (note the difference to multiple-base binding *sites* of Section 2.1) can be modeled independent of the others using a first order Markov chain. The DNA nucleotides $\mathrm{A}, \mathrm{C}, \mathrm{G}$ and T constitute the

state space of this chain. Let $\rho(u, v, t)$ denote the probability of transition from nucleotide u to v in time interval t. These can be used to formulate the *transition probability matrix* $\mathbf{P}(t) = \rho(u, v, t)$, $u, v \in \Sigma$ with $\sum_v \rho(u, v) = 1$, $\rho(u, v) \geq 0$ for a given time interval t.

The Markov process is assumed to have the following properties:

- *time homogeneity*: transition probabilities in a time interval remain same irrespective of the starting time point. In matrix notations, this leads to:

$$\mathbf{P}(t + t') = \mathbf{P}(t) \cdot \mathbf{P}(t')$$

- *stationarity*: the probability distribution over the nucleotides remains same over time. Assuming that each nucleotide can mutate to any other, this implies that the overall nucleotide distribution π remains same despite evolutionary change. Hence, the process is assumed to be in equilibrium with π being the equilibrium (stationary) distribution.

- *reversibility*: the overall probability of change from u to v in time t is equal to that from v to u in the same time. That is, $\pi(u)\rho(u, v) = \pi(v)\rho(v, u)$ for all $u, v \in \Sigma$ and time t.

For infinitesimally small time increment $h > 0$, the time homogeneity property yields:

$$\mathbf{P}(t + h) = \mathbf{P}(t)\mathbf{P}(h)$$

which on subtracting $\mathbf{P}(t)$ from both sides, dividing by h and taking limit $h \to 0$, yields the rate of change:

$$\begin{aligned} \frac{d\mathbf{P}(t)}{dt} &= \lim_{h \to 0} \frac{\mathbf{P}(t)\mathbf{P}(h) - \mathbf{P}(t)}{h} \\ &= \mathbf{P}(t) \lim_{h \to 0} \frac{\mathbf{P}(0 + h) - \mathbf{P}(0)}{h} \end{aligned}$$

The second equality above follows from the observation that there is no change when no time elapsed, $\mathbf{P}(0) = \mathbf{I}$, the identity matrix. Thus,

$$\mathbf{P}'(t) = \mathbf{P}(t)\mathbf{P}'(0) \tag{2.22}$$

Denoting $\mathbf{P}'(0)$ by $\mathbf{Q}$, the *instantaneous rate matrix*, leads to:

$$\mathbf{P}'(t) = \mathbf{P}(t) \cdot \mathbf{Q} \tag{2.23}$$

The instantaneous rate matrix $\mathbf{Q}$ gives the instantaneous rate of transition $\mathtt{q}(i, j)$ from nucleotide i to j. The diagonal entries $\mathtt{q}(i, i)$ are negative and off-diagonal entries $\mathtt{q}(i, j), j \neq i$ positive to ensure each row sums to zero. Using the initial condition $\mathbf{P}(0) = \mathbf{I}$, since at initial time we start from a constant state, solving the above differential equation results in:

$$\mathbf{P}(t) = e^{\mathbf{Q}t} \tag{2.24}$$

Thus, given an instantaneous rate matrix, it is possible to calculate the transition probabilities as a function of time. It is in the formulation of this matrix $\mathbf{Q}$ that most evolutionary models differ.

	A	C	G	T
A	–	$\mu/4$	$\mu/4$	$\mu/4$
C	$\mu/4$	–	$\mu/4$	$\mu/4$
G	$\mu/4$	$\mu/4$	–	$\mu/4$
T	$\mu/4$	$\mu/4$	$\mu/4$	–

(a) JC model

	A	C	G	T
A	–	$\mu\pi(\mathrm{C})$	$\mu\pi(\mathrm{G})$	$\mu\pi(\mathrm{T})$
C	$\mu\pi(\mathrm{A})$	–	$\mu\pi(\mathrm{G})$	$\mu\pi(\mathrm{T})$
G	$\mu\pi(\mathrm{A})$	$\mu\pi(\mathrm{C})$	–	$\mu\pi(\mathrm{T})$
T	$\mu\pi(\mathrm{A})$	$\mu\pi(\mathrm{C})$	$\mu\pi(\mathrm{G})$	–

(b) F81 model

Figure 2.5: *Evolutionary Models.* The instantaneous rate matrices for the Jukes-Cantor model (left) and the Felsenstein 1981 (right) model are shown. μ represents the mean instantaneous rate and $\pi(i)$ represent the background frequencies.

Given the transition probabilities and time span t, we can then calculate the probability of observing a letter pair u, v in an alignment of two sequences diverged t time ago.

$$\phi(u, v) = \pi(u) \cdot \rho(u, v, t) \tag{2.25}$$

The substitution score matrices described in Section 2.2.2 are usually formulated as log-odds ratios using the above derived probabilities. In the following, we use the notation ϕ_{uv} instead of $\phi(u, v)$ for simplicity.

Thus, most evolutionary models present an instantaneous rate matrix, whose product with time t is then exponentiated to yield the transition probability matrix. For a more detailed discourse on the concepts presented here and detailed discussion on existing evolutionary models, we refer the reader to introductory textbooks by Ewens *et al.* [59] and Hillis *et al.* [81]. An excellent review article is presented by Lió and Goldman [110].

Next, we present the commonly used Jukes-Cantor [90] model for modeling DNA substitution. Following this, we present two additional models that are relevant to this work for their applicability in modeling position-specific evolution.

Jukes-Cantor Model According to the Jukes-Cantor [90] model proposed in 1969, each substitution is equally likely, leading to uniform background frequencies, $\pi_i = 0.25$. The instantaneous rate matrix $\mathbf{Q}_{JC}$ looks as depicted in Fig 2.5(a) and the probabilities are given as:

$$\phi_{uv}(t) = 1/4 - 1/4e^{-\mu t} \qquad \forall \quad u \neq v \tag{2.26}$$

$$\phi_{uu}(t) = 1/4 + 3/4e^{-\mu t} \qquad \text{otherwise} \tag{2.27}$$

where μ is the mean instantaneous substitution rate and t the time interval.

Felsenstein 1981 model According to the Felsenstein model introduced in 1981 [62], the probability that a nucleotide is substituted by another is proportional to the stationary distribution of the incoming nucleotide. Hence, the difference to the Jukes-Cantor model introduced before is simply the possibility of unequal base frequencies. The instantaneous rate matrix $\mathbf{Q}_{F81}$ is of the form shown in Fig. 2.5(b), where π is not necessarily uniform. The probabilities are then calculated as:

$$\phi_{uv}(t) = e^{-\mu t}\delta(u, v) + (1 - e^{-\mu t})\pi(v) \qquad \forall \quad i, j \tag{2.28}$$

Through such a model, Felsenstein introduced the concept of an evolutionary steady state that respects the initial base composition. Note that here it is possible that some substitutions are not even observable. We will see later (Chapter 4) how the F81 model can be used to model binding site evolution.

Halpern-Bruno Model Introduced initially for modeling position-specific evolution in protein coding sequences, the Halpern-Bruno model (HB) [74] has also been used to model binding site evolution [133].

In this model, a site-invariant mutation rate is combined with a site-specific fixation rate to yield position-specific mutation rates $\mathsf{q}^i(u,v)$. The former is taken to be the background evolutionary model given by $\mathsf{q}_\mathsf{B}(u,v)$. The latter is calculated using the position-specific letter distribution $f^i(u), u \in \Sigma$ and the background mutation rates. For a position i, the mutation rate is then given as the following proportionality:

$$\mathsf{q}^i(u,v) \propto \mathsf{q}_\mathsf{B}(u,v) \times \frac{\ln x}{1-1/x} \tag{2.29}$$

where

$$x = \frac{f^i(v)\mathsf{q}_\mathsf{B}(v,u)}{f^i(u)\mathsf{q}_\mathsf{B}(u,v)}$$

For background evolution, any of the existing evolutionary models can be used. When $f^i(v)\mathsf{q}_\mathsf{B}(v,u) = f^i(u)\mathsf{q}_\mathsf{B}(u,v)$, then the rate is equal to the background mutation rate $\mathsf{q}_\mathsf{B}(u,v)$. The position-specific transition probabilities for a time interval t are then calculated by exponentiating the product of the rate matrix $\mathbf{Q}_{HB}$ formed with the above entries and time t. Note that the above values are for the off-diagonal entries, the diagonal entries are simply calculated such that the sum of the row entries is zero.