A simple way to find related sequences with position-specific probabilities

Martin C. Frith[1,2,3]

[1]Department of Computational Biology and Medical Sciences, University of Tokyo
[2]Artificial Intelligence Research Center, AIST
[3]AIST-Waseda University Computational Bio Big-Data Open Innovation Laboratory

Summary

One way to understand biology is by finding genetic sequences that are related to each other. Often, a family of related sequences has position-varying probabilities of substitutions, insertions, and deletions: we can use these to find distant and subtle relationships. There are popular software tools for this task, which all have limitations. They either do not use all probability evidence (e.g. PSI-BLAST, MMseqs2), or have excessive complexity and minor biases (e.g. HMMER). This complexity inhibits fertile development of alternative tools.

This study describes a simplest reasonable way to find related sequences with position-specific probabilities, using all probability evidence. The algorithms likely use the fewest operations that such algorithms possibly could, so they are fast and simple. This study does not describe practical heuristics for large data, but rather a "theory" that can be used in practical tools.

Introduction

This study was motivated by seeking subtly-related DNA sequences. Non-coding DNA is hardly ever conserved between vertebrates and invertebrates, but we recently found gene-regulating DNA conserved between humans, molluscs, arthropods, and even corals (Frith and Ni 2023). These DNA regions all control gene expression in embryonic development: a control system conserved since early Precambrian animals.

These DNA sequences have position-varying rates of substitutions, insertions, and deletions, for example, a conserved "CCAAT box" near the right of Fig. 1. This suggests we could find more-subtly related sequences by using these position-specific probabilities. This is a classic approach: a set of position-specific rates is termed a "profile", and there are many methods for finding sequence regions related to a profile (Gribskov et al. 1987, Krogh et al. 1994, Karplus et al. 1998, Yu et al. 2002, Steinegger and Söding 2017).

The currently popular profile methods are of two types. The first is exemplified by PSI-BLAST: it uses position-specific rates of substitution, but its insertion and deletion parameters are position-nonspecific and ad hoc (Altschul et al. 1997). Also, it considers just one alignment between potentially related regions. It would be more powerful to combine the evidence from alternative ways of aligning them (Eddy 2008, Frith 2024).

The other type of method uses position-specific probabilities of substitution, insertion, and deletion, and combines alternative alignments. The de facto standard tool seems to be HMMER, which is complex and quirky (Eddy 2008). The perceived need for this complexity might explain why few of the many sequence comparison tools use this full-probability approach. The present study shows that this complexity is not necessary.

HMMER has a large "implicit probabilistic model" (Eddy 2008). It doesn't allow insertions next to deletions. It is not free of alignment length bias (see below). It has parameters that change with the length of each sequence that is compared to a profile, so that equally good similarities get lower scores in longer sequences, which is arguably unnatural (Frith 2024). It is optimized for a yes/no question—do the sequence and profile have related parts?—which is not so useful for a long sequence (or profile) like a chromosome. Therefore, nhmmer (nucleotide HMMER) heuristically finds windows in a chromosome with potential matches to a profile, and runs the HMMER analysis on each window (Wheeler and Eddy 2013). It is unclear what the non-heuristic version of nhmmer would be. Since there is no cutoff between short and long sequences, it is desirable to have one method that is good for any length.

We can seek "local" similarities between part of a profile and part of a sequence, or "semi-global" similarities between a whole profile and part of a sequence. The popular tools are all local. The main reason is that it's hard to determine the probability of a semi-global similarity occuring by chance in a random sequence.

All these tools produce a similarity score, indicating the strength of evidence for related sequences. It is important to know the probability of a similarity score occurring by chance, in a random sequence. Unfortunately, this is hard to determine. There is a solution for gapless local alignment, in the limit of infinitely long sequences (Karlin and Altschul 1990). Gapped alignment relies on conjecture and simulations, which limit PSI-BLAST to its simple profile search. HMMER uses conjectures that apply to its sophisticated profile search. The scope of HMMER's conjectures is unclear: their practical application does not always give accurate results (Eddy 2008), and they seem to contradict the gapless solution described by Karlin and Altschul (1990) (see below).

The terms p-value and E-value are used in confusingly varying ways. This study defines them as follows. If we compare a profile to one or more random sequences:

- the p-value is the probability of a score $\geq s$ occurring,
- the E-value is the expected number of distinct sequence regions with score $\geq s$.

In a long random sequence, similarities to a profile occur randomly at a uniform rate. So the number of similar regions follows a Poisson distribution (in the limit where the sequence is long). Therefore $p = 1 - e^{-E}$.

This study shows a minimal, bias-free way to find sequence parts related to a profile, using position-specific substitution, deletion, and insertion rates, and combining evidence

```
gggtggtctc-tctac-tttggtgagctgttgctaagcagctaataatag---tcatgtttgctgagtaatttctgccct-ccgcgagccaatcg human PTCH1
gggtggtctc-tctac-tttggtgagctgttgctaagcagctaataatag---tcgtgtttgctgagtaatttctgcct--tcctcggccaatga chimaera ptch1
gggtggtcctggccac-tttggtgagctgttgcttagcagctaataatagt--agatgtttgctaagtaatttgcttcttccctttagccaatgg chimaera ptch2
gtgtggtctg-tcttc-tttggcgagccgtttcctagcagcaaatgacggg--ccgtgttactcgagcaaattgtggag--cccgcggccaatca lancelet
gtgtggtcta-cctctttttctggtgacgttactaagcaactaatgagaac--tcgtgttacatgagtcatttggccg---tgagtgaccaatca acorn worm
gtgtggtcttctccac-tttagcgaaccgttgcttggcaactaatgcaaggttcgctgttacgagaggagctagaaag---ccacgagccaatcg horseshoe crab
gtgtggtcttctatac-tttagtaagccgttgcttggcaactaatacaaggctcgctattacgagaggagctagaaag---ccacgagccaatcg horseshoe crab
gggtggtctt-tctac-tttggtgagccgttgcctggcaactaatgtgcagc-tgatgttacttgagcaatctgcgcta--taattgaccaatca chiton
gggtggtcta-tctac-ttgggtgagctgttgctaagcaactaatgttgggc-tggtgttacacgagcaatcctactga--taattgaccaatca sea hare
gggtggtctg--ctac-ctgggggagctgttgccaggcaactaatgttgggc-tggtgttacacgagcaatcctactga--taattgaccaatca abalone
gggtggtctt-tctac-tttggtgagctgttgctaggcaactaatgtcgggc-tgcagttacatgagctgcctgtgggc--ttaagaaccaatca shamisen shell
```

Figure 1: Alignment between promoter DNA sequences of *Ptch* (patched) genes in some animals (Frith and Ni 2023).

from alternative alignments. It determines E-values using a conjecture of Yu and Hwa (2001). It extends a recent position-nonspecific method (Frith 2024): the non-obvious aspect of this extension is how precisely to define alignment probabilities. It doesn't consider profile-versus-profile search, which is powerful but outside our scope (Steinegger et al. 2019).

Methods

Alignment scores

Let us start by reviewing score-based alignment. We wish to find related parts between a profile of length m, and a sequence of length n (Q_0, Q_1, ..., Q_{n-1}). The profile has these position-specific scores:

$S_i(y)$	$(0 \le i < m)$	score for letter type y at position i
d_i	$(0 \le i < m)$	score for first letter of a deletion
e_i	$(0 < i < m)$	score for other letter of a deletion
a_i	$(0 \le i \le m)$	score for first letter of an insertion
b_i	$(0 \le i \le m)$	score for other letter of an insertion

Typically, $a_i < b_i < 0$ and $d_i < e_i < 0$. This is the standard "affine gap" scheme, which works quite well because insertions and deletions are somewhat rare (low start probability) but often long (high extension probability).

Here, "insertion" means insertion in the sequence relative to the profile, and "deletion" likewise. Insertions can occur at $m+1$ sites: between adjacent positions in the profile, or before the first position, or after the last position.

Finding the maximum alignment score

We wish to find the maximum possible alignment score between any part of the sequence and any part of the profile. This can be found by the Smith-Waterman-Gotoh algorithm (Smith and Waterman 1981, Gotoh 1982). A slightly unusual version is shown in Algorithm 1. It considers each length-i prefix of the profile and length-j prefix of the sequence:

$X_{i-1\ j-1}$	is the maximum possible score for an alignment starting anywhere and ending at the ends of these prefixes with a match.
$Y_{i-1\ j}$	is the same ending with a deletion.
$Z_{i\ j-1}$	is the same ending with an insertion.

Thus, w is the maximum score for any alignment ending at the ends of these prefixes. Finally, the maximum score is the maximum w.

It might be more intuitive to write the algorithm with shifted definitions:

$$\widehat{X}_{ij} = X_{i-1\ j-1} \qquad \widehat{Y}_{ij} = Y_{i-1\ j} \qquad \widehat{Z}_{ij} = Z_{i\ j-1}\,. \quad (1)$$

This is merely a change of notation, and it obscures the similarity to the next algorithm we will consider.

This version of the Smith-Waterman-Gotoh recurrences probably has the lowest possible number of operations: max, add, and access to previous X, Y and Z values (Cameron et al. 2004). It also has simple initialization, just setting values to $-\infty$. As a tradeoff, it does excess calculations at the borders: when $i = 0$ or m, or $j = 0$ or n. In particular, it uses undefined values e_0, S_m, d_m, e_m, and Q_n. They can be set to arbitrary values, and have no effect on the outcome.

So far, we just have the maximum score. We can find an alignment with this score by tracing back from a point (i, j) with maximum w (see the Supplement). This does not require storing any trace pointers in Algorithm 1.

Limited alignment extension from an anchor

The Smith-Waterman-Gotoh algorithm is rarely used in practice, because it is too slow for large data. Instead, fast heuristics are used to find potentially related regions, which are then aligned. One way is to fix an "anchor" (tiny alignment), and extend alignments in a limited range on both sides (Fig. 2). The ranges (grey tiles) are not fixed in advance: they are adjusted as the algorithm proceeds. Several ways to do that have been suggested (Zhang et al. 1998, Suzuki and Kasahara 2017, Liu and Steinegger 2023).

This alignment extension algorithm is shown in Algorithm 2. It is almost the same as Algorithm 1: the main change is that the "0" is moved from the recurrence to the initialization at the extension start point. This is because we only consider alignments starting at this point, rather than starting anywhere.

Again, it is possible to use shifted definitions (Equations 1), but I suspect that makes it tricky to handle the irregular grey regions.

Summing alignment probabilities

Score-based alignment is equivalent to probability-based alignment (Frith 2020). An alignment's score is:

$$\text{score} = \log \frac{\text{prob(alignment)}}{\text{prob(length-0 alignment)}}\,. \quad (2)$$

Here, prob(alignment) is the product of probabilities of insertions, deletions, aligned letters, and unaligned letters.

One alignment between two related regions does not hold all evidence that they are related. To judge whether they are related, it is better to calculate their probability without

Algorithm 1: Maximum alignment score

$$
\begin{array}{lll}
X_{-1\ j-1} \leftarrow -\infty & Y_{-1\ j} \leftarrow -\infty & 0 \le j \le n \\
X_{i-1\ -1} \leftarrow -\infty & Z_{i\ -1} \leftarrow -\infty & 0 \le i \le m \\
\left.\begin{array}{l}
w \leftarrow \max(X_{i-1\ j-1}, Y_{i-1\ j}, Z_{i\ j-1}, 0) \\
X_{ij} \leftarrow w + S_i(Q_j) \\
Y_{ij} \leftarrow \max(w + d_i, Y_{i-1\ j} + e_i) \\
Z_{ij} \leftarrow \max(w + a_i, Z_{i\ j-1} + b_i)
\end{array}\right\} & & \begin{array}{l} 0 \le i \le m \\ 0 \le j \le n \end{array}
\end{array}
$$

Algorithm 2: Maximum alignment extension score

$$
\begin{array}{lll}
X_{ij} \leftarrow 0 & & \text{at the star} \\
X_{ij} \leftarrow -\infty & Y_{ij} \leftarrow -\infty & \text{at each white circle} \\
X_{ij} \leftarrow -\infty & Z_{ij} \leftarrow -\infty & \text{at each grey circle} \\
\left.\begin{array}{l}
w \leftarrow \max(X_{i-1\ j-1}, Y_{i-1\ j}, Z_{i\ j-1}) \\
X_{ij} \leftarrow w + S_i(Q_j) \\
Y_{ij} \leftarrow \max(w + d_i,\ Y_{i-1\ j} + e_i) \\
Z_{ij} \leftarrow \max(w + a_i,\ Z_{i\ j-1} + b_i)
\end{array}\right\} & & \text{at each black dot}
\end{array}
$$

Algorithm 3: Sum of alignment probabilities

$$
\begin{array}{lll}
X_{-1\ j-1} \leftarrow 0 & Y_{-1\ j} \leftarrow 0 & 0 \le j \le n \\
X_{i-1\ -1} \leftarrow 0 & Z_{i\ -1} \leftarrow 0 & 0 \le i \le m \\
\left.\begin{array}{l}
w \leftarrow X_{i-1\ j-1} + Y_{i-1\ j} + Z_{i\ j-1} + 1 \\
X_{ij} \leftarrow S'_i(Q_j) w \\
Y_{ij} \leftarrow d'_i w + e'_i Y_{i-1\ j} \\
Z_{ij} \leftarrow a'_i w + b'_i Z_{i\ j-1}
\end{array}\right\} & & \begin{array}{l} 0 \le i \le m \\ 0 \le j \le n \end{array}
\end{array}
$$

Algorithm 4: Sum of alignment extension probabilities

$$
\begin{array}{lll}
X_{ij} \leftarrow 1 & & \text{at the star} \\
X_{ij} \leftarrow 0 & Y_{ij} \leftarrow 0 & \text{at each white circle} \\
X_{ij} \leftarrow 0 & Z_{ij} \leftarrow 0 & \text{at each grey circle} \\
\left.\begin{array}{l}
w \leftarrow X_{i-1\ j-1} + Y_{i-1\ j} + Z_{i\ j-1} \\
X_{ij} \leftarrow S'_i(Q_j) w \\
Y_{ij} \leftarrow d'_i w + e'_i Y_{i-1\ j} \\
Z_{ij} \leftarrow a'_i w + b'_i Z_{i\ j-1}
\end{array}\right\} & & \text{at each black dot}
\end{array}
$$

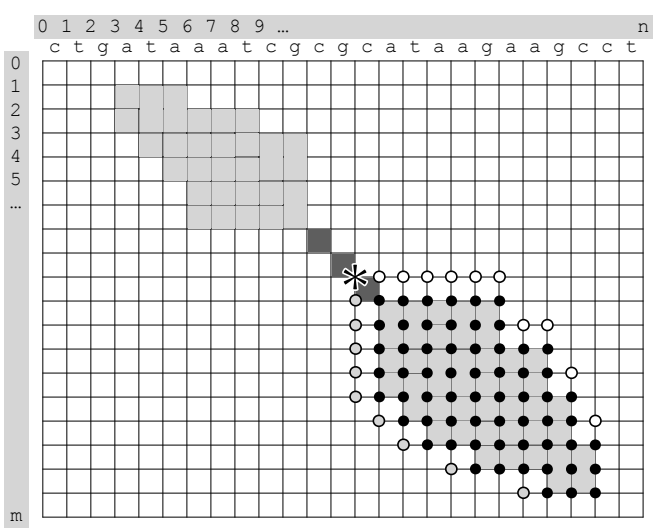

Figure 2: Extending alignments in a limited range either side of an anchor. The dark grey tiles show the anchor; the light grey tiles show the regions in which we seek alignments extending from it. The sequence is shown horizontally, with coordinates $0 \le j \le n$. The profile is vertical, with coordinates $0 \le i \le m$. The circles and star mark calculations in Algorithms 2 and 4.

fixing an alignment (Frith 2024). In other words, use a score like this:

$$
\begin{aligned}
\text{score} &= \log \frac{\sum_{\text{alignments}} \text{prob(alignment)}}{\text{prob(length-0 alignment)}} \\
&= \log \sum_{\text{alignments}} \frac{\text{prob(alignment)}}{\text{prob(length-0 alignment)}} .
\end{aligned}
\tag{3}
$$

Which alignments should be included in the sum? The most extreme answer is to include all alignments between any parts of the sequence and profile. This is the most powerful way to judge if there are any related parts, but it does not tell us where those parts are, which is not very useful if the sequence or profile is long (for example, a chromosome).

Yu and Hwa (2001) suggested an "end-anchored" approach: include all alignments that start anywhere and end at some fixed coordinates (i, j) in the profile and the sequence. This enabled them to determine E-values. It is, however, unnatural: it would be appropriate if related regions have similarity with unclear starts but clear and sharp ends. A "mid-anchored" approach is more natural: include all alignments that pass through some fixed coordinates (i, j) (Frith 2024). This is approximated by a practical "heuristic mid-anchored" method: include all alignment extensions each side of an anchor, in a limited range (Fig. 2).

The algorithms for summing alignment probabilities depend on the precise definition of insertion and deletion probabilities. It is possible to define them in such a way that the algorithms have the same number of operations as Algorithms 1 and 2. This is likely the lowest number of operations any such affine-gap algorithm could have.

Algorithm 3 calculates the sum of probability ratios in Equation 3. It considers each length-i prefix of the profile and length-j prefix of the sequence:

$X_{i-1\ j-1}$ is the sum of probability ratios for all alignments starting anywhere and ending at the ends of these prefixes with a match.
$Y_{i-1\ j}$ is the same ending with a deletion.
$Z_{i\ j-1}$ is the same ending with an insertion.

So, w is the sum of probability ratios of all alignments ending at the ends of these prefixes: the end-anchored probability ratio. The sum of all w values is the sum over all alignments. We can get mid-anchored probability ratios by combining end-anchored and start-anchored probability ratios (see the Supplement).

Algorithm 3 is a simple transformation of Algorithm 1, but it is not obvious that it validly sums alignment probabilities. To my knowledge, it has not been proposed before.

Algorithm 4 sums the probability ratios of all alignment extensions from an anchor: it is similar to Algorithm 2. There are alternative fast heuristics that do not extend from an anchor (Roddy et al. 2024): they can also use Algorithm 3 or 4 in a limited range, similar to the grey shapes in Fig. 2.

Algorithm parameters in terms of probabilities

The algorithms are based on these letter and gap probabilities:

$\theta_i(y)$	$(0 \le i < m)$	probability of letter type y at position i
$\psi(y)$		background probability of letter type y
δ_i	$(0 \le i < m)$	probability of starting a deletion
ϵ_i	$(0 < i \le m)$	probability of extending a deletion
α_i	$(0 \le i \le m)$	probability of starting an insertion
β_i	$(0 \le i \le m)$	probability of extending an insertion

Surprisingly, ϵ_m is included: it implies deletion of the non-existing position after the end of the profile. We will use it in the form $1 - \epsilon_m$, the probability of not extending past the end.

These probabilities are defined precisely in the next section, but in the end we get these parameters for Algorithms 3 and 4:

$$a'_i = \alpha_i(1 - \beta_i) \qquad (0 \le i \le m) \qquad (4)$$
$$b'_i = \beta_i \qquad (0 \le i \le m) \qquad (5)$$
$$d'_i = \delta_i(1 - \epsilon_{i+1}) \qquad (0 \le i < m) \qquad (6)$$
$$e'_i = \epsilon_i(1 - \epsilon_{i+1})/(1 - \epsilon_i) \qquad (0 < i < m) \qquad (7)$$
$$S'_i(y) = (1 - \alpha_i - \delta_i)\theta_i(y)/\psi(y) \qquad (0 \le i < m) \qquad (8)$$

Finally, the score parameters for Algorithms 1 and 2 are

$$S_i(y) = \log[S'_i(y)] \qquad (9)$$
$$a_i = \log[a'_i] \qquad (10)$$
$$d_i = \log[d'_i] \qquad (11)$$
$$b_i = \log[a'_i + b'_i] \qquad (12)$$
$$e_i = \log[d'_i + e'_i] \qquad (13)$$

The base of the log (for example, $\log_2$) is not specified, because any base is ok: changing the base merely rescales the scores.

Equations 12 and 13 need explanation. As an example, consider two adjacent inserted letters. The algorithms allow them to come from either one length-2 insertion, or two length-1 insertions. We should sum these probabilities, to include as much evidence as possible. This is done by Algorithms 3 and 4, and also by Equations 12 and 13.

Algorithms 1–4 lack one feature of HMMER: it allows position-specific probabilities of letter types in insertions. Our algorithms can gain this feature with just one extra array lookup (see the Supplement).

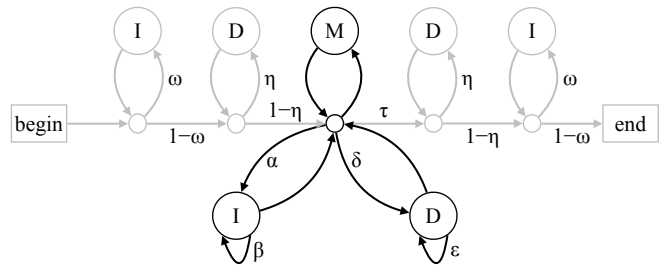

Figure 3: A scheme for assigning probability to: two sequences with one aligned region. The black part handles the aligned region; the grey parts handle unaligned flanks. Any path from **begin** to **end** represents an alignment: its probability is the product of the arrow probabilities (for example: ω, η) and letter probabilities. Each pass through a **D** matches the next letter in the "reference" sequence, with probability $\phi(z)$ for letter type z. Each pass through an **I** matches the next letter in the "query" sequence, with probability $\psi(z)$ for letter type z. Each pass through **M** matches a reference letter x and a query letter y, with probability $\pi(x, y)$.

Precise definition of probabilities

Let us start by reviewing position-nonspecific alignment of two sequences (Frith 2020). Here, prob(alignment) means the product of probabilities of insertions, deletions, aligned letters, and unaligned letters in both sequences. A precise definition of these probabilities is shown in Fig. 3. Here, α is the insertion start probability, and β the insertion extend probability. Also, δ and ϵ are the deletion start and extend probabilities. There are also probabilities τ for ending the aligned region, ω for a flanking letter in the "query" sequence, and η for a flanking letter in the "reference" sequence. Overall, this defines probabilities for alignments between part of one sequence and part of the other.

Note that we apply these probabilities to two given sequences, of length m and n, and only consider paths that correspond to sequences of lengths m and n.

A definition of position-specific probabilities is shown in Fig. 4. It is derived by "unrolling" Fig. 3 for each position in the reference sequence, which is now the profile. An unusual feature is that we have arrows leading to empty space: these correspond to profile length $\neq m$. As above, we only consider paths that correspond to profile length m, so we never use these arrows. It is possible to omit these arrows, but doing so makes the probabilities complicated (see the Supplement).

Algorithm parameters from probabilities

We wish to calculate scores by Equation 2 or 3. Firstly, a length-0 alignment between the sequence and profile only traverses grey arrows in Fig. 4, never black arrows, so

$$\text{prob(length-0 alignment)} = \eta^m(1-\eta)^2\omega^n(1-\omega)^2\tau\prod_{k=0}^{n-1}\psi(Q_k)\,. \qquad (14)$$

Next, consider a length-1 alignment that matches Q_j to M_i:

$$\text{prob(alignment)} = \eta^{m-1}(1-\eta)^2 \omega^{n-1}(1-\omega)^2(1-\alpha_i-\delta_i-\tau)\tau\theta_i(Q_j)\prod_{k\neq j}\psi(Q_k)\,. \qquad (15)$$

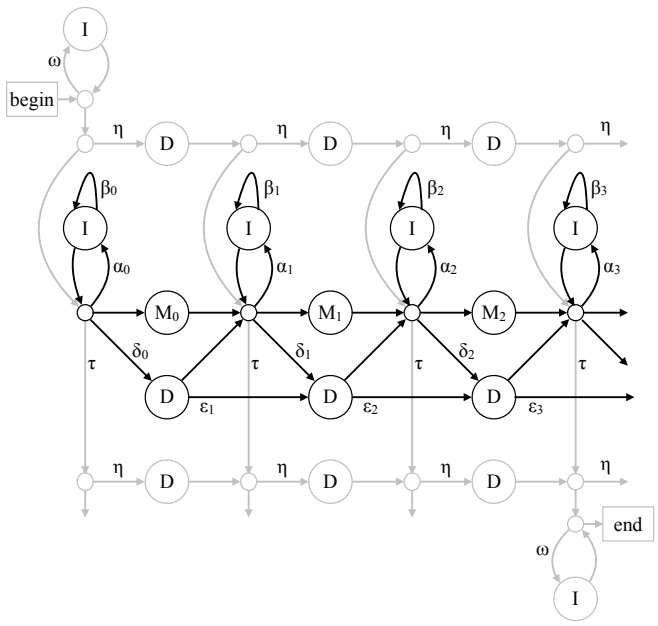

Figure 4: A scheme for assigning probability to: a sequence with one segment aligned to a profile. The black part handles alignment to the profile; the grey parts handle unaligned flanks. In this example, the profile has length $m = 3$. Any path from **begin** to **end** represents an alignment: its probability is the product of the arrow probabilities (for example: ω, η) and letter probabilities. Each pass through an **I** matches the next letter in the sequence, with probability $\psi(y)$ for letter type y. Passing through $\mathbf{M}_i$ matches the next letter in the sequence, with probability $\theta_i(y)$.

From dividing Equation 15 by Equation 14, we can infer that

$$S'_i(y) = (1 - \alpha_i - \delta_i - \tau)\theta_i(y)/(\eta\omega\psi(y)) \,. \tag{16}$$

By similar reasoning, we can infer that

$$a'_i = \alpha_i(1 - \beta_i)/\omega \tag{17}$$
$$b'_i = \beta_i/\omega \tag{18}$$
$$d'_i = \delta_i(1 - \epsilon_{i+1})/\eta \tag{19}$$
$$e'_i = \epsilon_i(1 - \epsilon_{i+1})/(1 - \epsilon_i)/\eta \tag{20}$$
$$c'_i = (1 - \alpha_i - \delta_i - \tau)/(\eta\omega) \tag{21}$$
$$S'_i(y) = c'_i\theta_i(y)/\psi(y) \tag{22}$$

These are the same as Equations 4–8, assuming that $\tau \approx 0$ and $\omega \approx 1 \approx \eta$. These choices for ω, η, and τ mean there is no alignment length bias.

Alignment length bias

The arrow probabilities may be biased towards long or short alignments, between a sequence and profile of given lengths. If ω, η, and τ are all high (almost 1), shorter alignments (with longer unaligned flanks) have higher probability. Conversely, if ω, η, and τ are all near 0, longer alignments (shorter flanks) have higher probability. There is no length bias if $\omega \approx 1 \approx \eta$ and $\tau \approx 0$ (Frith 2020).

Lack of such bias may be desirable, for two reasons. Firstly, it means alignment is based purely on similarity, whereas long-bias aggressively aligns slightly dissimilar regions, and short-bias declines to align slightly similar regions. Secondly, it enables a way to determine E-values (Yu and Hwa 2001).

For gapless alignment ($\alpha_i = \delta_i = 0$), there is no alignment length bias when

$$c'_i = 1 \,, \tag{23}$$

or equivalently

$$1 - \tau = \eta\omega \,. \tag{24}$$

This means alignments of any length have the same product of arrow probabilities. Equation 23 is equivalent to Equation 1 in the work of Karlin and Altschul (1990). Also, ω, η, and τ are partly arbitrary: their values make no difference to our algorithms as long as they satisfy Equation 24.

As a point of comparison, HMMER3 is biased to longer alignments, at least in its gapless unihit mode. It has a probability $p < 1$ equivalent to our ω, without a counterbalancing equivalent of $\tau > 0$ (Eddy 2008).

Chance similarities in random sequences

Suppose we get a similarity score s by Equation 3. We would like to know the probability of getting a score $\geq s$ by chance, in a random sequence. This can be estimated for a sequence of random independent letters, if:

- We use end-, start-, or mid-anchored scores.
- There is no alignment length bias.
- The random sequence has letter frequencies $\psi(y)$.

Based on the conjecture of Yu and Hwa, the E-value is:

$$E = Kn/\exp(s) \,. \tag{25}$$

Here, exp is defined to be the inverse of log (i.e. it depends on the base of the log). K is an unknown parameter, which is typically different for different profiles. It is unfortunate that K is unknown, but it can be estimated in milliseconds by fitting to similarity scores of random sequences (Yu et al. 2002, Frith 2024). K is roughly proportional to profile length, because a longer profile has more chances to match a random sequence, provided that the profile is not repetitive.

HMMER3 E-value conjectures

The HMMER3 E-value conjectures seem to contradict the solution for gapless local alignment of two sequences (Karlin and Altschul 1990). For gapless alignment, we have scores S_{xy} for matching letter type x in one sequence and y in the other. If they are based on the probabilities in Fig. 3 (with $\alpha = \delta = 0$),

$$S_{xy} = \log_{\text{base}} \frac{(1 - \tau)\pi_{xy}}{\eta\omega\phi_x\psi_y} \,. \tag{26}$$

Other ways of defining local alignment probabilities would produce similar equations. If we compare two random sequences with letter probabilities Φ_x and Ψ_y, the maximum alignment score follows a Gumbel distribution with a parameter λ, which is the solution of $\sum_{x,y} \Phi_x\Psi_y e^{\lambda S_{xy}} = 1$ (Karlin and Altschul 1990). According to HMMER's conjectures, $\lambda = \log[\text{base}]$. But that is not always true.

Making a profile

We would like to make a profile from sequences such as those in Fig. 1. This can be done by previously-described methods, summarized in the Supplement (Krogh et al. 1993, Krogh et al. 1994, S Henikoff and JG Henikoff 1994, Karplus et al. 1998).

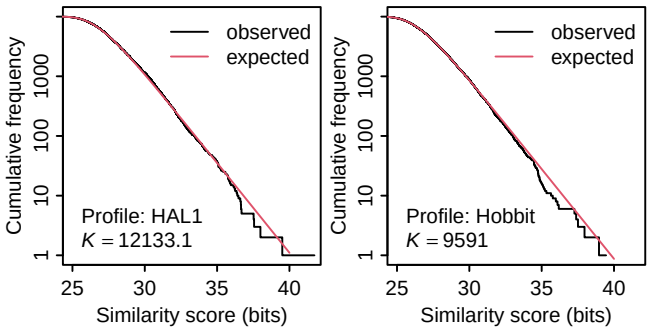

Figure 5: Distribution of similarity scores between: the HAL1 profile and random DNA sequences (left); the Hobbit profile and random protein sequences (right). 10000 random sequences were generated, each of length 10000, using the profile's background letter probabilities $\psi(y)$. For each sequence, the maximum mid-anchored similarity score was found. The expected distribution depends on K, which was fitted to the observed similarity scores.

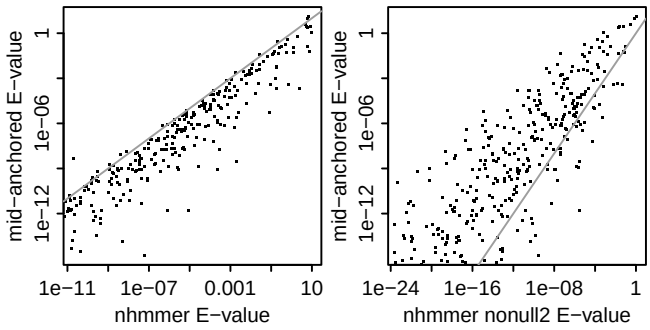

Figure 6: E-values for similarity between: the HAL1 profile, and human DNA with HAL1 RepeatMasker annotation. Each dot shows one DNA sequence. Each sequence has 3 kinds of E-value: mid-anchored, nhmmer, and nhmmer with option `--nonull2`.

We can also use existing profiles, for example, HMMER profiles are available in the Dfam and Pfam databases. HMMER defines insertion and deletion probabilities slightly differently than in Fig. 4, so they will be slightly suboptimal for our algorithms.

This study implemented reading of HMMER profiles (see the Supplement), calculation of end-, start-, and mid-anchored similarity scores, and estimation of K (`https://gitlab.com/mcfrith/seq-position-probs`). One detail is that the background letter probabilities $\psi(y)$ are set proportional to the geometric mean of $\theta_i(y)$. This aims to stop the profile matching sequences that have no similarity other than similar letter abundances (Barrett et al. 1997).

Results

Chance similarities in random sequences

Let us check the probabilities of similarity scores in random sequences. Fig. 5 shows two examples, using a Dfam profile for HAL1 (an ancient LINE retrotransposon), and a Pfam profile for Hobbit (a family of lipid transfer proteins). In both cases, the observed probability of getting a mid-anchored similarity score agrees closely with the expected probability. This is also true for end- and start-anchored scores, and for other Dfam profiles (Supplementary Figs. S4–12).

One limitation is that all these profiles are long ($m > 1500$). The E-values are less accurate for short profiles, due to an edge effect (Yu and Hwa 2001).

Sensitivity/specificity comparison to nhmmer

Finally, here is a preliminary test of sensitivity/specificity, relative to nhmmer. Some HAL1 DNA sequences were extracted from the human genome (hg38), by taking the first 1000 segments (plus 100 nt flanks) annotated as HAL1 in `rmskOutCurrent` at the UCSC genome database. These annotations come from nhmmer/RepeatMasker v4.0.7 and Dfam 2.0. So this test is unfair in favor of nhmmer.

These HAL1 sequences were stored in one multi-fasta file, which was searched against the HAL1 profile. For each sequence, the highest mid-anchored score and its E-value were found, and the best nhmmer score and E-value were retained. These two E-values are comparable: they mean the expected number of matches in random sequences with the same length as *all* the HAL1 sequences.

The mid-anchored E-values tend to be lower than the nhmmer E-values (Fig. 6 left). This means the mid-anchored approach can find the HAL1 matches with a lower rate of random matches (Fig. 5 left). So it is more sensitive, for any given rate of random matches (specificity).

As a further test, the nhmmer search was repeated with an option `--nonull2`, which turns off its correction for letter abundances. This time, the mid-anchored E-values tend to be higher than the nhmmer E-values (Fig. 6 right).

Overall, these preliminary tests suggest the proposed approach has comparable sensitivity/specificity to nhmmer. Some other Dfam profiles produced similar results (Supplementary Figs. S13–20). In a detailed test, the "winner" might depend mainly on the way of handling biased sequence composition.

Discussion

This study shows how to achieve powerful HMMER-like detection of sequences related to profiles, without the complexity and minor quirks of HMMER. It is only necessary to implement a Gotoh-like algorithm (for example, Algorithm 3 or 4), whose parameters have simple formulas (Equations 4–8).

A practical implementation is still lacking. Some use cases (rapid search of large profiles against large genomes) would require standard, well-engineered heuristics for large data. Alternatively, searching small profiles ($m <$ a few hundred) against typical animal genomes could perhaps be done directly by an efficiently-implemented Gotoh-like algorithm. Probability-summing algorithms suffer from the values getting too large to for computers to handle efficiently: this is not a problem for use cases where there are only weak similarities.

One missing ingredient is a correction for E-value edge effect: the E-values are too conservative if the sequence or profile is short. This seems important, because profiles are often short. Such corrections are possible, but seem a bit complicated (Yu and Hwa 2001, Eddy 2008).

This study neglects one important feature of HMMER: multihit comparison (Eddy 2008). We have assumed relationship between *one* part of a sequence and a profile, with unrelated flanks. This assumption can produce bad results when there are multiple related parts (Zhang et al. 1999). HMMER-style

multihit comparison fixes the root cause of this problem, by assuming a *set* of related parts (Frith and Kawaguchi 2015). It is ideal if there is an ancestor-to-derived profile-to-sequence relationship, allowing for any number of duplications, deletions, repositionings, and insertions (Frith and Khan 2018).

Acknowledgments

I thank Travis Wheeler for explaining HMMER *E*-values.

References

Altschul SF et al. (1997). Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Research* **25**:3389–3402.

Barrett C, Hughey R, Karplus K (1997). Scoring hidden Markov models. *Bioinformatics* **13**:191–199.

Cameron M, Williams HE, Cannane A (2004). Improved gapped alignment in BLAST. *IEEE/ACM Transactions on Computational Biology and Bioinformatics* **1**:116–129.

Eddy SR (2008). A probabilistic model of local sequence alignment that simplifies statistical significance estimation. *PLoS Computational Biology* **4**:e1000069.

Frith MC (2020). How sequence alignment scores correspond to probability models. *Bioinformatics* **36**:408–415.

Frith MC (2024). A simple method for finding related sequences by adding probabilities of alternative alignments. *Genome Research* **34**:1165–1173.

Frith MC, Kawaguchi R (2015). Split-alignment of genomes finds orthologies more accurately. *Genome biology* **16**:1–17.

Frith MC, Khan S (2018). A survey of localized sequence rearrangements in human DNA. *Nucleic acids research* **46**:1661–1673.

Frith MC, Ni S (2023). DNA conserved in diverse animals since the Precambrian controls genes for embryonic development. *Molecular Biology and Evolution*:msad275.

Gotoh O (1982). An improved algorithm for matching biological sequences. *Journal of molecular biology* **162**:705–708.

Gribskov M, McLachlan AD, Eisenberg D (1987). Profile analysis: detection of distantly related proteins. *Proceedings of the National Academy of Sciences* **84**:4355–4358.

Henikoff S, Henikoff JG (1994). Position-based sequence weights. *Journal of molecular biology* **243**:574–578.

Karlin S, Altschul SF (1990). Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes. *Proceedings of the National Academy of Sciences* **87**:2264–2268.

Karplus K, Barrett C, Hughey R (1998). Hidden Markov models for detecting remote protein homologies. *Bioinformatics* **14**:846–856.

Krogh A et al. (1993). *Hidden Markov models in computational biology: Applications to protein modeling*. Tech. rep. University of California at Santa Cruz.

Krogh A et al. (1994). Hidden Markov models in computational biology: Applications to protein modeling. *Journal of molecular biology* **235**:1501–1531.

Liu D, Steinegger M (2023). Block Aligner: an adaptive SIMD-accelerated aligner for sequences and position-specific scoring matrices. *Bioinformatics* **39**:btad487.

Roddy JW, Rich DH, Wheeler TJ (2024). nail: software for high-speed, high-sensitivity protein sequence annotation. *bioRxiv*. DOI: 10.1101/2024.01.27.577580.

Smith TF, Waterman MS (1981). Identification of common molecular subsequences. *Journal of molecular biology* **147**:195–197.

Steinegger M, Söding J (2017). MMseqs2 enables sensitive protein sequence searching for the analysis of massive data sets. *Nature biotechnology* **35**:1026–1028.

Steinegger M et al. (2019). HH-suite3 for fast remote homology detection and deep protein annotation. *BMC bioinformatics* **20**:1–15.

Suzuki H, Kasahara M (2017). Acceleration of nucleotide semi-global alignment with adaptive banded dynamic programming. *bioRxiv*. DOI: 10.1101/130633.

Wheeler TJ, Eddy SR (2013). nhmmer: DNA homology search with profile HMMs. *Bioinformatics* **29**:2487–2489.

Yu YK, Bundschuh R, Hwa T (2002). Hybrid alignment: high-performance with universal statistics. *Bioinformatics* **18**:864–872.

Yu YK, Hwa T (2001). Statistical significance of probabilistic sequence alignment and related local hidden Markov models. *Journal of Computational Biology* **8**:249–282.

Zhang Z, Berman P, Miller W (1998). Alignments without low-scoring regions. *Journal of Computational Biology* **5**:197–210.

Zhang Z et al. (1999). Post-processing long pairwise alignments. *Bioinformatics* **15**:1012–1019.