Supplement to: A simple method for finding related sequences by adding probabilities of alternative alignments

Martin C. Frith

Supplemental methods

Rescaling to avoid overflow in Algorithm 2

Here is one way to avoid overflow, used in this study's implementation. First, initialize an extra variable: $u \leftarrow 0$. Then, execute alg. 2 in order of increasing antidiagonal: $k = i + j$. Every 16th antidiagonal, just after executing alg. 2 with $k = 15 \pmod{16}$, update u:

$$u \leftarrow u + \log(v') \,. \tag{S1}$$

Also, rescale the just-calculated Y'_{ij} and Z'_{ij} values with $i + j = k + 1$:

$$Y'_{ij} \leftarrow Y'_{ij}/v' \tag{S2}$$

$$Z'_{ij} \leftarrow Z'_{ij}/v' \,, \tag{S3}$$

and the X'_{ij} values with $(i + j) \in \{k + 1, k + 2\}$:

$$X'_{ij} \leftarrow X'_{ij}/v' \,. \tag{S4}$$

Then, reset $v' \leftarrow 1$. After the algorithm finishes, the final extension score is $u + \log(v')$.

x-drop implementation

LAST defines regions in which it seeks alignment extensions (gray area in Fig. 1B) by an x-drop method, similar to one described by Zhang et al. (2000). For alg. 1, it executes the algorithm in order of increasing antidiagonal: $k = i + j$. When executing alg. 1 on antidiagonal k (i.e. when $i + j = k$), i is restricted to a range: $A_k \leq i \leq B_k$. It starts with $A_0 = B_0 = 0$, and adjusts A_k and B_k as follows:

$$A_{k+1} = \begin{cases} A_k & \text{if } W_{A_k\, k-A_k} \geq V_k - x \\ A_k + 1 & \text{otherwise} \end{cases} \tag{S5}$$

$$B_{k+1} = \begin{cases} B_k + 1 & \text{if } W_{B_k\, k-B_k} \geq V_k - x \\ B_k & \text{otherwise} \end{cases} \tag{S6}$$

Here, W_{ij} is defined to be $\max(X_{ij}, Y_{ij}, Z_{ij})$. Also, V_k is the value of v (the highest score seen so far) just before executing alg. 1 on antidiagonal k. x is the maximum score drop. The recommended value for x is just below the minimum score for reporting an alignment (Zhang et al. 1999; Frith et al. 2010).

For alg. 2, LAST first runs alg. 1 and then re-uses the same gray area for alg. 2. The best method is unclear. LAST's DNA-versus-protein alignment is different: it's version of alg. 2 has its own x-drop-like algorithm (Yao and Frith 2023). If we do not re-use the same gray area, the similarity score (from alg. 2) may be inconsistent with the representative alignment (from alg. 1), e.g. huge score but tiny alignment.

prob(null alignment)

If we define gap probabilities as in Fig. 2, a null alignment is produced by a path that never traverses the α_D, α_I, or γ arrows. Suppose we compare a length-m sequence (R_1, R_2, ..., R_m) to a length-n sequence (Q_1, Q_2, ..., Q_n).

$$\text{prob(null alignment)} = \left(\prod_{k=1}^{m} \omega_D \phi_{R_k}\right)\left(\prod_{k=1}^{n} \omega_I \psi_{Q_k}\right)(1-\omega_D)^2(1-\omega_I)^2(1-\gamma-\alpha_D-\alpha_I)\,. \tag{S7}$$

Parameters for Algorithm 1

The parameters for alg. 1 are defined like this (Supplement 3.1 in Frith 2020):

$$S_{xy} = \log[S'_{xy}] \tag{S8}$$
$$a_D = \log[a'_D] \tag{S9}$$
$$a_I = \log[a'_I] \tag{S10}$$
$$b_D = \log[b'_D + a'_D] \tag{S11}$$
$$b_I = \log[b'_I + a'_I] \tag{S12}$$

Eq. S11 defines the probability of extending a deletion as the sum of two probabilities: extension via the β_D arrow, and extension via the $1-\beta_D$ and α_D arrows (Fig. 2). Eq. S12 does the same for insertions.

Estimating K from random sequences

As mentioned in the main text, we can estimate K by generating some pairs of random i.i.d. sequences, calculating $s_{\max}$ for each pair, and fitting K. The probability of a score between random sequences is supposed to be

$$\text{prob}(s_{\max} \geq s) = 1 - e^{-Kmn/exp(s)}\,. \tag{S13}$$

This means that the probability density is proportional to

$$e^{-Kmn/\exp(s)} Kmn/\exp(s)\,. \tag{S14}$$

Suppose we generate N pairs of random sequences, and their scores are $s_1, s_2, \ldots, s_N$. The probability density of these scores is propotional to

$$\prod_{i=1}^{N} e^{-Kmn/\exp(s_i)} Kmn/\exp(s_i)\,. \tag{S15}$$

We wish to find the value of K that maximizes this, or equivalently its logarithm:

$$\sum_{i=1}^{N} \ln(Kmn/\exp(s_i)) - Kmn/\exp(s_i)\,. \tag{S16}$$

The maximum value occurs when the derivative with respect to K equals zero:

$$\sum_{i=1}^{N} 1/K - mn/\exp(s_i) = 0\,. \tag{S17}$$

Thus, the maximum-likelihood value of K is

$$K = N/(mn \sum_{i=1}^{N} 1/\exp(s_i)) \tag{S18}$$

$$= \text{harmonic mean}\,[\exp(s_{\max})]\,/\,mn\,. \tag{S19}$$

Finding $\omega_D = \omega_I$ that satisfies eq. 12

As mentioned in the main text, `last-train` chooses the values of ω_D and ω_I in the following way. It assumes that $\omega_D = \omega_I$, and finds the unique value that avoids bias towards long or short alignments (eq. 12).

First, it assumes that $\omega_D > \beta_D$ and $\omega_I > \beta_I$. Otherwise, alignment gaps would have non-negative scores, which is not useful for sequence comparison.

As $\omega_D = \omega_I$ increases from $\max(\beta_D, \beta_I)$ to 1, the LHS of eq. 12 decreases, from a value > 1 to a value < 1. So it equals 1 at a unique point, which can be found by e.g. repeated bisection.

Alignment column probabilities

We can calculate the probability that each pair of letters is aligned. In other words, the probability that R_i is aligned to Q_j. First, note that alg. 2 calculates X'_{ij}, which is the probability ratio sum of alignment extensions ending with R_i aligned to Q_j. Another "backward" algorithm (alg. S1) calculates W''_{ij}: the probability ratio sum of alignments starting just after R_i aligned to Q_j. This means that $X'_{ij}W''_{ij}$ is the probability ratio sum of all alignment extensions that include R_i aligned to Q_j. Finally,

$$\text{prob}(R_i \text{ aligns to } Q_j) = X'_{ij}W''_{ij}/v'\,. \tag{S20}$$

Here, v' is the output of alg. 2: the probability ratio sum of all alignment extensions.

Algorithm S1 Probability ratios of alignments starting at each point in an $n \times n$ block

```
$W''_{n+1\,n+1} \leftarrow 0$    $W''_{i\,n+1} \leftarrow 0$    $W''_{n+1\,i} \leftarrow 0$    $(0 < i \le n)$
                                 $Z''_{i\,n+1} \leftarrow 0$    $Y''_{n+1\,i} \leftarrow 0$    $(0 \le i \le n)$

$W''_{ij} \leftarrow W''_{i+1\,j+1} \cdot S'_{R_{i+1}Q_{j+1}} + Y''_{i+1\,j} \cdot a'_D + Z''_{i\,j+1} \cdot a'_I + 1$  }  $n \ge i \ge 0$
$Y''_{ij} \leftarrow W''_{ij} + Y''_{i+1\,j} \cdot b'_D$                                                                     }  $n \ge j \ge 0$
$Z''_{ij} \leftarrow W''_{ij} + Z''_{i\,j+1} \cdot b'_I$                                                                     }
```

It's also possible to calculate gap probabilities:

$$\text{prob}(R_i \text{ aligns to a gap between } Q_j \text{ and } Q_{j+1}) = Y'_{ij} Y''_{ij} / v' \,, \tag{S21}$$

$$\text{prob}(Q_i \text{ aligns to a gap between } R_j \text{ and } R_{j+1}) = Z'_{ij} Z''_{ij} / v' \,. \tag{S22}$$

The colors of gap columns in Fig. 7 indicate the probability of aligning to a gap anywhere. In other words, $\sum_j Y'_{ij} Y''_{ij} / v'$ and $\sum_i Z'_{ij} Z''_{ij} / v'$.

It may be more efficient to replace the red 1 in alg. S1 with $1/v'$. This causes all the values calculated by the algorithm to be divided by v', saving the division in eqs. S20–S22.

There are various ways to make alignments based on these probabilities. Two such ways are implemented in LAST: γ-centroid alignment and LAMA alignment (Hamada et al. 2011). These were not used in the present study.

Column probabilities with rescaling

Alg. S1 can be modified to use rescaling. First, we need to store the rescaling values used during alg. 2. We can store them in an array T by doing this at each rescaling step during alg. 2:

$$T_{\lfloor k/16 \rfloor} \leftarrow 1/v' \,. \tag{S23}$$

In alg. S1, replace the red 1 with a variable c, initialized by $c \leftarrow 1/v'$. This uses the value of v' at the end of alg. 2.

Then, execute alg. S1 in order of decreasing antidiagonal: $k = i + j$. Just after executing alg. S1 with $k = 14 \pmod{16}$, rescale all the recently-calculated W''_{ij}, Y''_{ij}, and Z''_{ij} values with $i + j \in \{k, k+1\}$:

$$W''_{ij} \leftarrow T_{\lfloor k/16 \rfloor} W''_{ij} \tag{S24}$$

$$Y''_{ij} \leftarrow T_{\lfloor k/16 \rfloor} Y''_{ij} \tag{S25}$$

$$Z''_{ij} \leftarrow T_{\lfloor k/16 \rfloor} Z''_{ij} \,. \tag{S26}$$

Also, rescale c:

$$c \leftarrow T_{\lfloor k/16 \rfloor} c \,. \tag{S27}$$

How the results were obtained

Sequence data

The genomes, proteins, and U2 DNA sequence (`NR_002716.3`) are from NCBI:

Organism	Sequence data identifier
Homo sapiens	`hg38_no_alt_analysis_set`
Plasmodium falciparum	`GCF_000002765.5`
Plasmodium yoelii	`GCF_900002385.2`
Aquifex aeolicus	`GCF_000008625.1`
Pyrolobus fumarii	`GCF_000223395.1`

The consensus sequences of ancient repeats (shared by mammals and reptiles) are at `https://gitlab.com/mcfrith/sum-align`. They are repeats from Dfam 3.7 with taxon "Amniota" and classification
`root;Interspersed_Repeat;Transposable_Element`
or
`root;Interspersed_Repeat;Unknown`.
Some repeats were excluded, because they are confusingly similar to younger repeats: `LINE/L2`, `LINE/CR1`, `MIR1_Amn`, `AmnSINE1`, `Chompy-6_Croc`, and `LmeSINE1c`.

Finding ancient repeats in the human genome

LAST version 1471 was used throughout. First, an index (called `repDB`) was made of the repeat consensus sequences:

```
lastdb -c -S2 -uMAM8 repDB repeats.fasta
```

- `-c` masks sequence regions that have biased composition.
- `-S2` indexes both DNA strands (so we needn't use both strands of the genome).
- `-uMAM8` increases sensitivity but also run time and memory use (Frith and Noé 2014).

Next, rates of matches, mismatches, and gaps were found between the repeat sequences and the human genome:

```
last-train --revsym -X1 --sample-number=5000 -D1e9 repDB human.fasta > rep.train
```

- `--revsym` makes the match and mismatch rates strand-symmetric, e.g. the `a`:`c` rate equals the `t`:`g` rate.
- `-X1` treats unknown "n" bases in the repeats, which are numerous, as ambiguous: $S_{xy} = \log[\gamma/(\omega_D \omega_I)]$ if $x =$ `n`.
- `--sample-number=5000` increases the number of random genome fragments that `last-train` uses, for fear that the ancient repeats are rare in the genome.

- For the same reason, `-D1e9` sets a stricter similarity threshold. It uses similarities that are expected by chance $\leq$ once per 10^9 genome base-pairs. (This is no longer recommended for LAST versions $>$ 1519, because `last-train` tunes its `D` automatically.)

The reversed genome was searched against the repeats like this:

```
lastal -u3 -m100 -p rep.train -J1 repDB human-rev.fasta > rev.maf
```

- `-u3` applies masking throughout. (By default, after related regions have been found with masking, they are aligned without masking.)
- `-m100` makes it more sensitive and slow.
- `-J1` specifies alg. 2.

The (non-reversed) genome was searched against the repeats like this:

```
lastal -u3 -m100 -p rep.train -J$J -K1 repDB human.fasta > fwd.maf
```

- `$J` was replaced with either `0` (alg. 1) or `1` (alg. 2).
- `-K1` omits any alignment whose genome range lies in a higher-scoring alignment.

Plasmodium DNA analysis

First, an index (called `falcDB`) was made of the *P. falciparum* genome:

```
lastdb -c -R02 falcDB falciparum.fasta
```

- `-R02` uses `tantan` parameters suitable for `at`-rich DNA (Frith 2011).

Then, rates of matches, mismatches, and gaps were found:

```
last-train --revsym falcDB yoelii.fasta > plasmo.train
```

The reversed *P. yoelii* genome was searched like this:

```
lastal -u3 -m100 -D1e5 -p plasmo.train -J1 falcDB yoelii-rev.fasta > rev.maf
```

Proteins

First, an index (called `aquiDB`) was made of the *A. aeolicus* proteins:

```
lastdb -c -p aquiDB Aquifex-prot.fasta
```

- `-p` specifies that these are protein sequences.

Then, rates of matches, mismatches, and gaps were found:

```
last-train -m1000 aquiDB Pyrolobus-prot.fasta > prot.train
```

The reversed *P. fumarii* proteins were searched like this:

```
lastal -u3 -m1000 -D1e3 -p prot.train -J1 aquiDB Pyrolobus-prot-rev.fasta > rev.maf
```

The (non-reversed) *P. fumarii* proteins were searched like this:

```
lastal -u3 -m1000 -D1e3 -p prot.train -J$J -K1 aquiDB Pyrolobus-prot.fasta > fwd.maf
```

U2

First, an index (called `U2db`) was made of the U2 DNA sequence:

```
lastdb -c -S2 U2db U2.fasta
```

Then, rates of matches, mismatches, and gaps were found:

```
last-train --revsym --sample-number=50000 -D1e9 U2db human.fasta > U2.train
```

The human genome was searched against U2 like this:

```
lastal -u3 -p rep.train -J$J -K1 U2db human.fasta > fwd.maf
```

The two U2 results were obtained by running lastal with either `rep.train` or `U2.train`.

Similarity search without masking biased composition

Unmasked similarities were found by running `lastdb` without `-c` and `lastal` without `-u3`.

Supplemental Results

Length-dependence of random similarity scores

The formula for probability of similarity scores between random sequences (eq. 15) is inaccurate for short sequences. To check this, the non-heuristic similarity score ($s_{\max}$) was calculated for pairs of random i.i.d. sequences, of varying length (Fig. S1). These scores were used to estimate K (eq. S19). For sequence length 40 (Fig. S1 top row), the similarity scores clearly differ from the expected distribution. The estimate of K seems to stabilize by sequence length 10000, though it is impossible to prove that by such tests. For sequence length 500, the estimates of K are too low by 8–16%.

Random match/mismatch similarity scores

It may be interesting to see similarity scores between random sequences, for the simplest possible alignment probabilities: gapless match/mismatch probabilities. This means that $\alpha_D = \alpha_I = 0$, and π_{xy} has a fixed value when $x = y$, and a second fixed value when $x \neq y$. Four probability settings were checked (which correspond to simple match/mismatch scores):

π_{xy} $(x = y)$	π_{xy} $(x \neq y)$	% identity	match score	mismatch score
0.222	0.009337	89	2	-3
0.1875	0.02083	75	1	-1
0.1628	0.02905	65	5	-4
0.141	0.03634	56	3	-2

The non-heuristic similarity score ($s_{\max}$) was calculated for pairs of random i.i.d. sequences (Fig. S2). These scores were used to estimate K (eq. S19). Interestingly, K increases as the match probability (π_{xx}) decreases. This is opposite to the behavior of K for ordinary gapless alignment (Altschul 1993), but consistent with previous results for hybrid alignment (Yu and Hwa 2001; Yao and Frith 2023).

Another trend is that the estimate of K converges more slowly as the match probability decreases. In other words, the % difference between K estimated with sequence length 500 versus 10000 becomes bigger.

References

Altschul SF. 1993. A protein alignment scoring system sensitive at all evolutionary distances. *Journal of Molecular Evolution* **36**: 290–300.

Frith MC. 2011. A new repeat-masking method enables specific detection of homologous sequences. *Nucleic Acids Research* **39**: e23–e23.

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

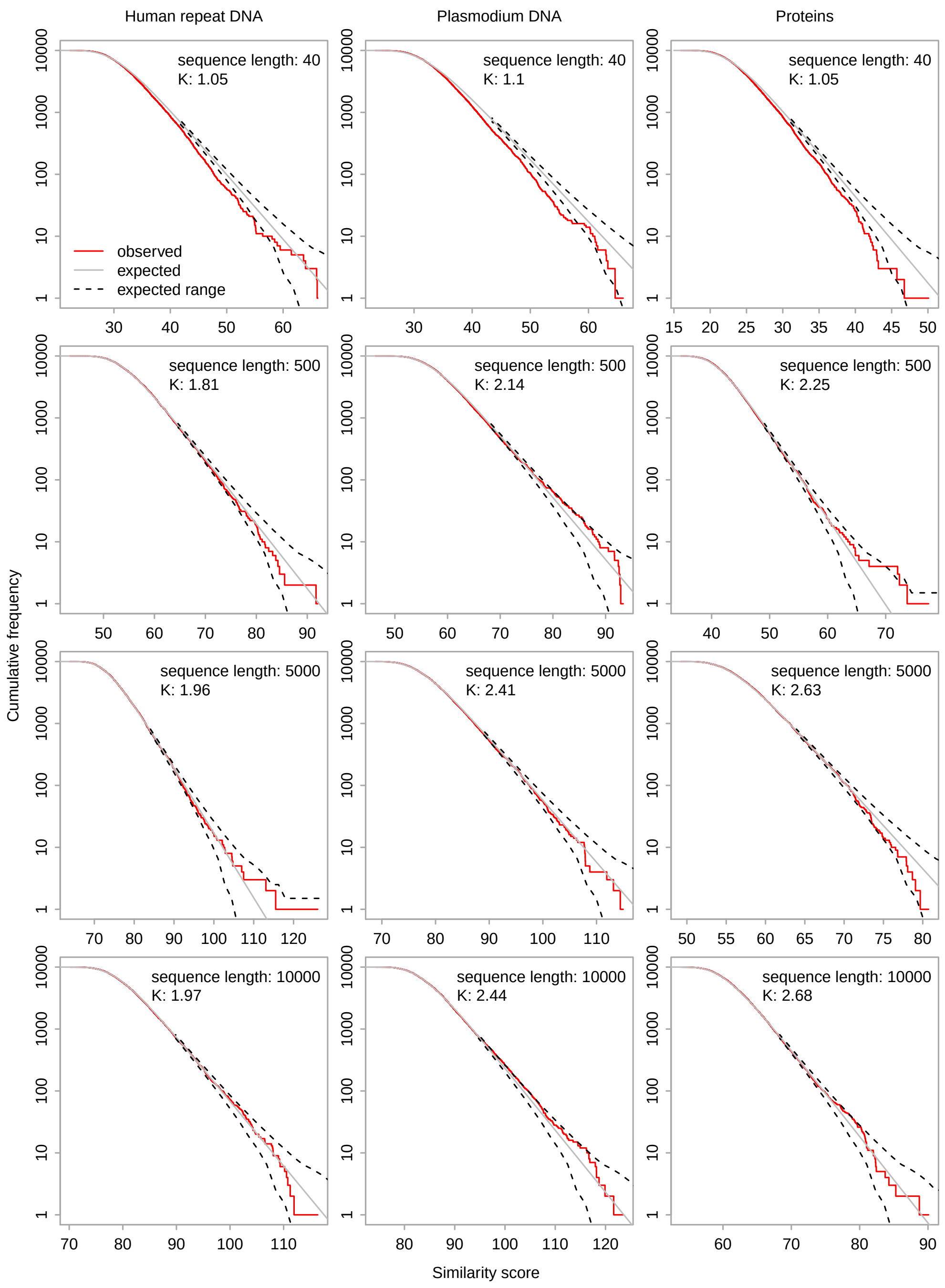

Figure S1: Non-heuristic similarity score ($s_{\max}$) between 10000 pairs of random i.i.d. sequences, for three sets of alignment parameters. The frequency has a 5% chance of being outside the dashed lines (2.5% each).

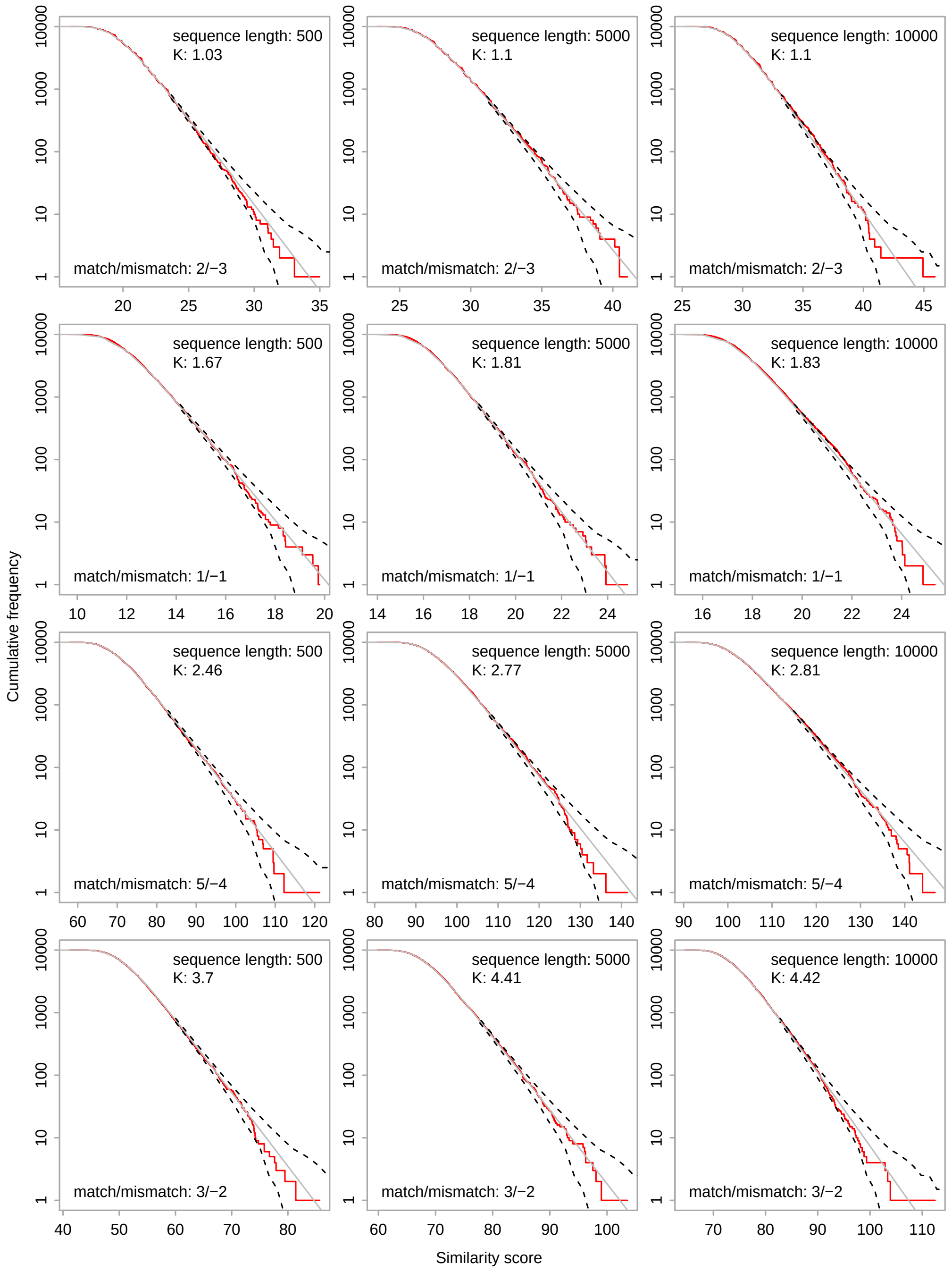

Figure S2: Non-heuristic similarity score ($s_{\max}$) between 10000 pairs of random i.i.d. sequences, for four sets of match/mismatch parameters. The frequency has a 5% chance of being outside the dashed lines (2.5% each).

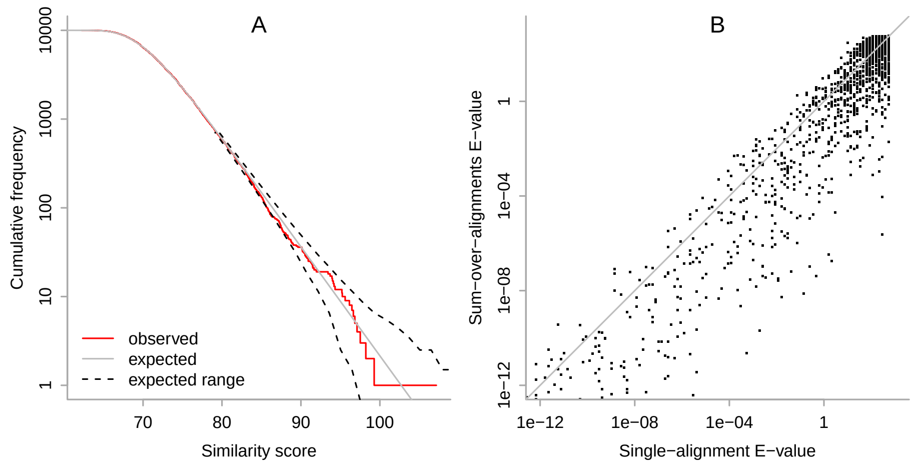

Figure S3: Results for protein sequences with BLOSUM62 letter probabilities. Panel A was prepared in the same way as the right-hand panel of Fig. 3, and panel B was prepared in the same way as Fig. 6B, except that `--fixmat=BLOSUM62` was added to the `last-train` options (using LAST version 1549).

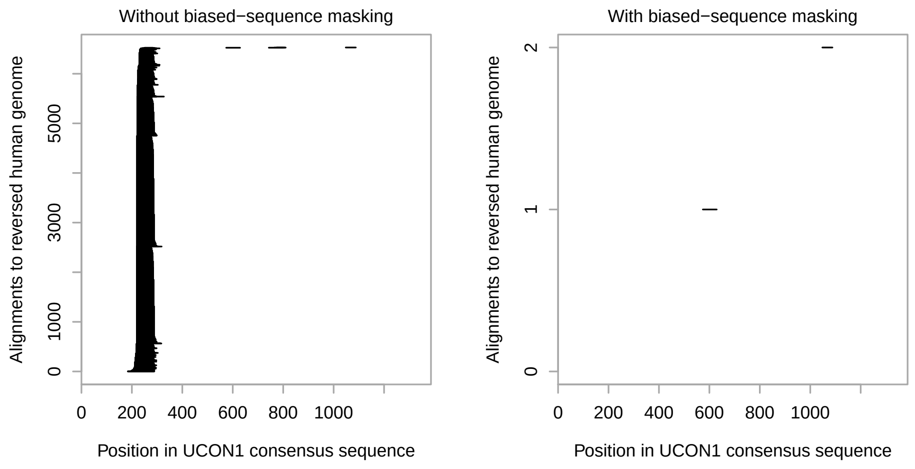

Figure S4: Alignments between the UCON1 consensus sequence and the reversed human genome, with or without biased-sequence masking. Similarity scores were found by alg. 2, and a representative alignment for each similarity was found by alg. 1. Each horizontal line shows the part of UCON1 covered by one alignment. Shown here are alignments with score ≥ 126.2, which is expected to occur once between random sequences with the same length as: UCON1, and the whole genome. The left panel shows a huge pile of alignments, on the region of biased composition shown in Fig. 4A.

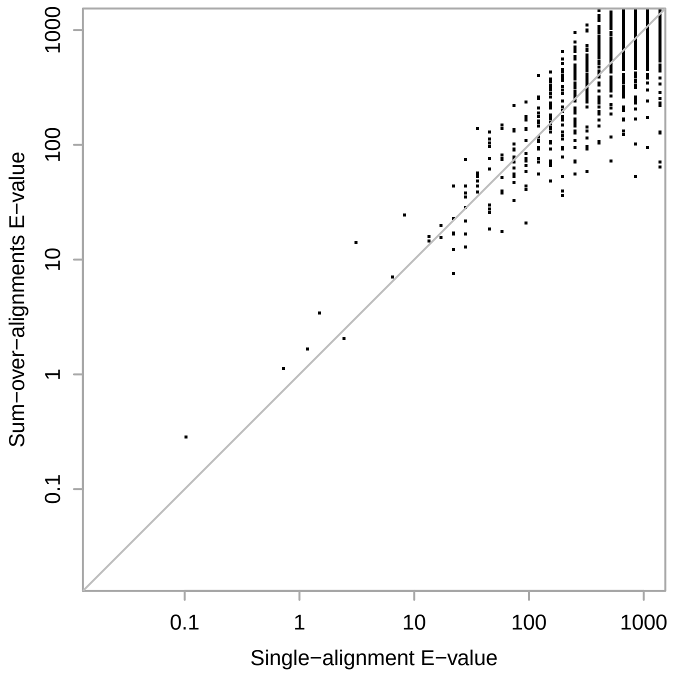

Figure S5: *E*-values for identical alignments found by alg. 1 (horizontal axis) and alg. 2 (vertical axis). Each point is one alignment. The diagonal gray line indicates equal *E*-values. These alignments are between shuffled DNA sequences: the same sequences as in Fig. 6A. They were shuffled with `fasta-shuffle-letters` from MEME Suite 5.5.5. The genome was shuffled with the `-fix N` option, which fixes the large blocks of `N` that are present. The alignments were found in the same way as for Fig. 6A, using the same `last-train` output file.

Frith MC, Hamada M, Horton P. 2010. Parameters for accurate genome alignment. *BMC Bioinformatics* **11**: 1–14.

Frith MC, Noé L. 2014. Improved search heuristics find 20 000 new alignments between human and mouse genomes. *Nucleic Acids Research* **42**: e59–e59.

Hamada M, Wijaya E, Frith MC, Asai K. 2011. Probabilistic alignments with quality scores: an application to short-read mapping toward accurate SNP/indel detection. *Bioinformatics* **27**: 3085–3092.

Yao Y, Frith MC. 2023. Improved DNA-versus-protein homology search for protein fossils. *IEEE/ACM Transactions on Computational Biology and Bioinformatics* **20**: 1691–1699.

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, Wiehe T, Miller W. 1999. Post-processing long pairwise alignments. *Bioinformatics* **15**: 1012–1019.

Zhang Z, Schwartz S, Wagner L, Miller W. 2000. A greedy algorithm for aligning DNA sequences. *Journal of Computational Biology* **7**: 203–214.