KEZW-BI-ME/00007

August 13, 2020

Substitutional Residue-Difference Map (SRD Map) to help locate mis-alignments in Multiple Sequence Alignment (MSA): toward Artificial-Intelligence-assisted probability distribution of alternative MSAs.

Kiyoshi Ezawa

Independent (ORCID: 0000-0003-4906-8578)

Current address: 3-1-33 Nakamura-machi, Chichibu 368-0051, JAPAN;

Phone & Fax: +81-494-22-5501; E-mail: kezawa.ezawa3@gmail.com

Abstract

Although reconstruction of multiple sequence alignments (MSAs) is central to the advanced study of homologous (*i.e.*, ancestor-sharing) biological sequences, such as DNA and protein sequences, this process is also known to be error-prone. In order to correct errors, an important step is to *accurately* locate the mis-alignment, *i.e.*, the portion where the alignment is erroneous.

Here, we propose a new method, named "substitutional residue-difference map" (or SRD map for short). It maps "observed" SRDs along tree branches, which are computed from the residue configuration of each column using the likelihood method, onto the direct product of the set of sites (or column positions) and the set of branches. If this "observed" SRD is significantly larger than the SRD expected from the substitution model, that position (specified by the column-position and the tree branch) is suspected to be involved in a mis-alignment. To examine whether this hypothesis is true, we used 10,000 reconstructed MSAs of DNA sequences simulated along the phylogenetic tree of 15 mammals, and found that the average SRDs on the mis-alignments (along individual branches) are indeed significantly larger than those off the mis-alignments.

To take advantage of this finding, we wrote a Perl module and a script that performs the sliding window analysis to find candidates of mis-alignments in an input reconstructed MSA. By tuning several parameters, it captured as much as 66-71% of mis-alignments with the false-positive rate of 1.8-0.8%.

As a somewhat related subject, we also developed a tool (Perl script) that attempts to detect (or rather exclude) MSA regions containing "complex" errors using the gap-configurations alone. Despite depending on this limited information, tool successfully excluded about 65% of gapped segments in "complex" errors while keeping 80% of other segments.

These results imply that combining the SRD map and the gap-configuration will enable us to locate, and possibly characterize, mis-alignments more accurately. Because such a combination contains a rich amount and variety of information, it should be necessary to resort to a machine learning technique, especially the deep learning of artificial-intelligence (AI), to take advantage of the combination. If such an AI-assisted method is incorporated into our new program package, ANEX (standing for "alignment neighborhood explorer"), which constructs a probability distribution of alternative MSAs in each window, we may be able to construct much more accurate probability distributions of alternative MSAs.

Meanwhile, if you combine the SRD map and the "Position-Shift map" we previously developed [1], you may be able to conduct more meticulous simulation studies to characterize MSA errors in more details; such studies will also help improve more accurate location of mis-alignments in the future.

The tools developed in this study are available as modules and supplementary scripts of the ANEX(_P) open-source package [2], at an FTP repository of Bioinformatics.org (https://www.bioinformatics.org/ftp/pub/anex/).

[**Keywords:** sequence alignment, multiple sequence alignment (MSA), error, mis-alignment, artificial intelligence (AI), deep learning, machine learning, substitutions, insertion/deletion (indel), probability distribution, accuracy, DNA sequence, biological sequence]

[**Abbreviations:** alignment neighborhood explorer (ANEX), multiple sequence alignment (MSA), pairwise sequence alignment (PWA) substitutional residue-difference (SRD)]

Contents

1 Introduction

1.1 Background

The reconstruction of multiple sequence alignments (MSAs) is *central to* the advanced studies of homologous (*i.e.*, ancestor-sharing) biological sequences, such as DNA, RNA and protein sequences (*e.g.*, [3, 4, 5, 6, 7, 8]). Hence, the development of an aligner that accurately and/or quickly reconstructs MSAs has been the subject of vigorous and diligent efforts during the recent decades (e.g., [3, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24]).

As it has turned out, however, this crucial step of MSA reconstruction is *highly* error-prone [25, 19, 26, 27, 1]. And the errors, or mis-alignments, in MSAs influence the results, and even the conclusions, of the downstream analyses (*e.g.*, [28, 29, 30, 4, 26, 31]), in particular the inference of insertions/deletions [19, 32].

For these reasons, a number of methods have been developed that attempt to identify and correct errors, or mis-alignments, (or, more precisely, low reliability regions) in MSAs (*e.g.*, [33, 34, 35, 36, 37, 38, 39]). On the other hand, as some studies on sequence alignments suggest (*e.g.*, [40, 41, 1]), a (near-)majority of such mis-alignments are due to the stochastic nature of sequence evolution processes, and thus it is inevitable to construct a probability

distribution of alternative sequence alignments (*e.g.*, [42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 32, 53, 54, 55, 56, 2]), instead of merely reconstructing a single optimum alignment.

Whether you aim to detect and correct mis-alignments in an MSA reconstructed by an aligner of single-optimum type, or to construct a probability distribution of alternative MSAs, it should be greatly useful and beneficial to understand more about the nature of mis-alignments. With such a philosophy in mind, some studies were conducted to characterize mis-alignments (*e.g.*, [27, 1]). Especially, in [1], we proposed the concept of "Position-Shift Map," which maps position-shifts (the difference in residue-positions between true and reconstructed MSAs) onto either true or reconstructed MSAs. The "Position-Shift Map" is useful because it can identify mis-alignments at the level of single-site resolution. It has, however, an obvious drawback: it depends essentially on the true MSAs, which usually is not available in the real-life sequence study. Therefore, its use is inevitably limited to, *e.g.*, simulation studies, where the true MSAs are available.

Aiming to overcome this drawback of the "Position-Shift Map" while keeping its usefulness, we came up with the concept of "substitutional residue-difference map" (or "SRD Map" for short), which maps the SRD (the difference in residue states on the two sites of each branch) onto the MSA extended to include internal branches, as a sort of surrogate for the "Position-Shift Map." Our hypothesis was that mist-alignments tend to make SRDs higher than usual, and thus the positions in the extended MSAs with high SRDs will highlight the boundaries between regions of different position-shifts. To see whether this hypothesis is correct or not, we developed a tool to construct the SRD Map, as well as some other tools, and conducted simulation analyses. Irrespective of the results, it should be worth while to characterize mis-alignments in terms of SRDs.

In addition, we also developed a tool to identify regions with "complex" errors [27, 1] using gap-configurations *alone*, in order to examine the power of gap-configurations to identify, or characterize, mis-alignments.

This paper reports the principles and theories behind these tools, as well as the results

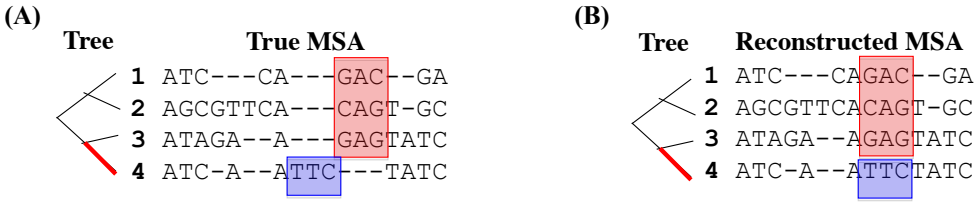

Figure 1: **Typical mis-alignment.** This example shows a "purge." **A.** The tree of aligned sequences (on the left) and the true MSA (on the right). The rectangles shaded in red and blue are the blocks of residues, each of which is aligned with a block of gaps. The residue-blocks are *vertically* supported by complementary sets of sequences, which are separated from each other by the thick red branch in the tree. **B.** An example reconstructed MSA, in which the red and blue blocks in panel A are erroneously aligned, causing the removal (*i.e.*, "purge") of the gaps that actually existed.

of the analyses.

1.2 Structure of this paper

This article is structured as follows. In Section 2, we explain the basic principles and theories underlying the method proposed here. More specifically, we describe: the SRD Map (in subsection 2.1), the sliding-window analysis to identify likely mis-alignments (in subsection 2.2), and using gap-configurations to detect "complex" errors (in subsection 2.3). Then, in Section 3, we describe how the methods have been implemented (in subsection 3.1), and the results of their validations (in subsection 3.2). Finally, in Section 4, we discuss the implications of this study and some of possible further developments.

We also provided a number of appendixes, in which we discuss some important yet detailed subjects.

2 Principles and Theories on the New Methods

2.1 Substitutional Residue-Difference Map (SRD Map) to help locate mis-alignments in MSA

Typically, a **mis-alignment** occurs when an aligner attempts to optimize the MSA score by decreasing the number (or sizes) of gaps. (See *e.g.*, Figure 1, which shows a typical case of "purge.") In consequence, a typical mis-alignment tends to create a small-scale "**shear**," where mutually aligned residue blocks exhibit a *nearly* random residue match/mismatch pattern against each other. Taking advantage of this tendency, we may be able to identify, and visualize, MSA portions that likely harbor mis-alignments.

In the following, we describe a possible method. For this purpose, we *define* a quantity, "**substitutional residue-difference**," or "**SRD**" for short. Let us consider a branch (referred to as b here) in a phylogenetic tree of aligned sequences (for example, the thick red branch in Figure 1 A), and pay attention to the residue states at both ends of b in an MSA column. If the residue states are identical, the SRD along the branch and in the column is *defined* as 0 (zero); if they are different, the SRD there is *defined* as 1 (one).

Usually, however, we do *not* know the residue states at the internal nodes. Therefore, SRDs in an MSA *must* be *estimated*, in some way, from the residue configurations of the MSA. Here, we propose a method to estimate SRDs, which is *the most natural* from the viewpoint of the theory to calculate the likelihood of a tree (and a substitution model) given (the residue configuration of) an MSA(, in other words, the probability of (the residue configuration of) an MSA, given a tree (and a substitution model)) (see, *e.g.*, [57, 58, 59]). Here, for convenience, we consider that a tree ($\mathcal{T}$) and a substitution model (Θ_S) is given.

We also assume that the substitution model (Θ_S) is such that the evolution (*via* substitutions) of each site (or column) is independent of the other sites (or columns). Therefore, we can focus on the calculation in each column (referred to as c).

Now, let $P[c]$ be the probability of observing the residue configuration of column c, and let $P_{(U,L)}[\ \omega,\ \omega',\ c \mid b\ ]$ be the *joint* probability of observing the residues ω and ω' at the upper- and lower-ends, respectively, of branch b in column c, *in addition to* observing the residue configuration of c. [1] Then, let $D_{obs}(b,c)$ be the expected value of the SRD "observed" along b, estimated from the residue configuration of c. It is *most naturally* calculated as:

$$\begin{aligned} D_{obs}(b,c) &= \left\{ \sum_{\omega\ \in\Omega} \sum_{\omega'(\neq\omega)\ \in\Omega} P_{(U,L)}[\ \omega,\ \omega',\ c \mid b\ ] \right\} \Big/ P[c] \\ &= 1 - \left\{ \sum_{\omega\ \in\Omega} P_{(U,L)}[\ \omega,\ \omega'=\omega,\ c \mid b\ ] \right\} \Big/ P[c]\ . \end{aligned} \tag{1}$$

Here the Ω is the set of all possible residue states. Therefore, if we know the values of $P[c]$ and $P_{(U,L)}[\ \omega,\ \omega',\ c \mid b\ ]$ for all c's and b's, we can calculate the **"observed" SRDs** in all c's and along all b's. Incidentally, to obtain the 2nd equality in Eq. 1 we used the identity:

$$P[c] \equiv \sum_{\omega\ \in\Omega} \sum_{\omega'\ \in\Omega} P_{(U,L)}[\ \omega,\ \omega',\ c \mid b\ ]\ . \tag{2}$$

Actually, the $P[c]$ can be calculated by the standard pruning algorithm [57, 58]. To calculate the $P_{(U,L)}[\ \omega,\ \omega',\ c \mid b\ ]$, we further introduce the following probabilities:

(i) $P(\omega \mapsto \omega'; b)$ denotes the conditional probability that, given residue ω ($\in \Omega$) at the upper-end of branch b, we have residue ω' ($\in \Omega$) at its lower-end;

(ii) $P_L(\omega; b, c)$ denotes the conditional probability that, given residue ω ($\in \Omega$) at the lower-end of branch b, we observe the residues of all its descendant extant sequences present in column c; (see Figure 12 A for illustration);

(iii) $P_U(\omega; b, c)$ denotes the joint probability that, in column c, we observe ω ($\in \Omega$) at the upper-end of branch b *and* the residues of all extant sequences on the upper-end-side of branch b; (see Figure 12 A for illustration).

[1] Here, for clarity, we omit the dependence on the substitution model (Θ_S) from the notation, considering that calculations are done under a single, fixed model. Extension to the case where we have a distribution of models should be obvious, though.

The $P(\omega \mapsto \omega'; b)$'s actually correspond to the components of the finite-time transition probability matrix calculated under Θ_S. Thus we assume that we can calculate all of them. The $P_L(\omega; b, c)$'s are actually by-products of the pruning algorithm [57, 58]. Thus, if we modify the algorithm so that they will be kept to the end and output along with the $P[c]$, they also can be calculated easily. The $P_U(\omega; b, c)$'s are somewhat new, and they can be calculated *via* a top-down recursion algorithm that is somewhat similar to the pruning algorithm. In appendix A, we have provided the concrete algorithm(s) to calculate the $P_L(\omega; b, c)$'s and $P_U(\omega; b, c)$'s. The time complexities of both these algorithms are $O(|\Omega|^2 \cdot N_S \cdot N_c)$, where the $|\Omega|$ is the size of the alphabet (Ω) (*i.e.*, the number of all possible residue states (at a site)), the N_S is the number of aligned sequences, and the N_c is the number of columns in the MSA.

Now, using the probabilities introduced above, the desired $P_{(U,L)}[\ \omega,\ \omega',\ c \mid b\ ]$'s are easily calculated as follows:

$$P_{(U,L)}[\ \omega,\ \omega',\ c \mid b\ ] = P_U(\omega; b, c)\ P(\omega \mapsto \omega'; b)\ P_L(\omega'; b, c)\ . \tag{3}$$

This calculation (for all ω, ω', c and b) has the time complexity of $O(|\Omega|^2 \cdot N_S \cdot N_c)$, and calculating the $D_{obs}(b, c)$'s *alone* should take the time complexity of $O(|\Omega| \cdot N_S \cdot N_c)$(, if the 2nd equality in Eq. 1 is used). Thus, by substituting Eq. 3 into (the 2nd equality of) Eq. 1, we can easily estimate the "observed" SRDs along all branches in all MSA columns.

By mapping the "observed" SRDs calculated as above onto the direct product of the set of all MSA columns and the set of all branches, we can create an **SRD Map** (Figure 2 E). The SRD Map visualizes, especially, some horizontal segments in the direct product space where the "observed" SRDs are higher than the surrounding (or higher than expected from the branch lengths) (Figure 2 E). In a previous study [1], we proposed the concept of a "position-shift map," which visualizes how mis-alignments occurred in the reconstructed MSA at the level of site-wise resolution (Figure 2 C). If desired, we could extend the "position-shift map" to the direct product space on which the SRD Map is created, by mapping the

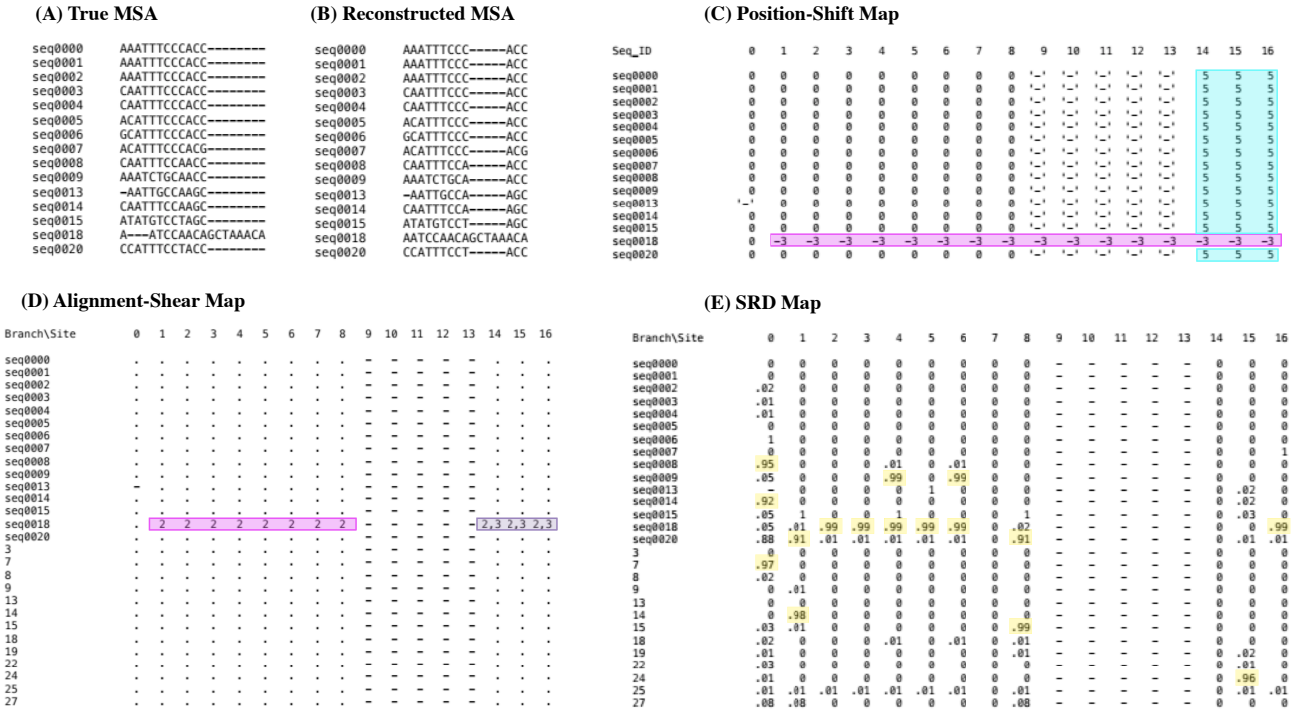

Figure 2: **Alignment-Shear Map and SRD Map.** **A.** The true MSA of 15 simulated mammalian sequences. **B.** A reconstructed MSA of the same sequences. **C.** The position-shift map, based on the reconstructed MSA (in B). The colored rectangles represent position-shift-blocks. **D.** The alignment-shear map, based on the position-shift map (in C). The colored rectangles represent the "alignment-shear"s. The '.' (dot) indicates the position (specified by a site and a branch) has no alignment-shear. **E.** The substitutional-residue-difference (SRD) map, based on the reconstructed MSA (in B). The SRDs shown are rounded to the nearest hundredth. The positions (specified by a site and a branch) at which SRD ≥ 0.9 are highlighted in yellow. In each panel, the '-' (dash) represents a gap. In panels D and E, if even one end of the branch has an "absence" state (*i.e.*, a gap), the position is assigned an "absence" state. Again, in panels D and E, each row represents a branch: the external branches are identified by the corresponding sequence IDs, and the internal branches are identified by integer IDs. Here are the key to the integer IDs (the left-hand-side is the branch ID, and the right-hand-side is a set of indexes (from 0 through 14) of MSA rows under the branch): 3 = {0, 1}; 7 = {3, 4}; 8 = {2, 3, 4}; 9 = {0, 1, 2, 3, 4}; 13 = {6, 7}; 14 = {5, 6, 7}; 15 = {0, 1, 2, 3, 4, 5, 6, 7}; 18 = {8, 9}; 19 = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}; 22 = {10, 11}; 24 = {10, 11, 12}; 25 = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12}; 27 = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13}.

vertical boundary of each block of position-shifts onto the branch that delimits the block (times its horizontal support) (Figire 2 D). In some typical reconstructed MSAs of simulated sequences (taken from [1]), the (horizontal) lines of "observed" SRDs in the SRD Map seem *well* correlated with the "alignment-shear"s, which are the (horizontal) lines *delimiting* the position-shift blocks (as in Figure 2 D). [2]

These sample cases give us a hope that the SRD Map may, to some extent, serve as a surrogate for the "Position-Shift Map." To confirm this idea (or hope), we used simulated MSAs (in [1]) and computed the averages and distributions of SRDs *on* the delimiting lines of position-shift blocks and those *off* the delimiting lines, and compared them. See sub-subsection 3.2.1 below for details on this validation analysis.

2.2 Sliding-window analysis to identify MSA portions where mis-alignments (or "purge"-like errors) are likely

Another possible usage of the "observed" SRDs calculated with Eq. 1 (& Eq. 3) is to attempt to *directly* locate the mis-alignments using the SRDs, regardless of the visualization. This should be more suitable for massive MSA data analyses, or automated processing of SRDs. For this usage, we also developed a method for the **sliding-window analysis**.

To conduct a sliding-window analysis, we need to *compare* the "observed" SRD (in each column along each branch) with a value of SRD *theoretically expected* from the substitution model (Θ_S) and possibly from the residue configuration of the column. Depending on the substitution model, the substitution rates can vary substantially among the initial residue states. Therefore, the **theoretical expectation** (in column c along branch b)), denoted as

[2] We have created a new Perl script, "map_aln_shear_clm_based.ver0.6.pl" and a few related scripts (, all in the ANEX(_P) package [2]), which automatically map the "alignment-shear"s onto the "extended MSA" (*i.e.*, the direct product of the set of sites times the set of branches), given an input "position-shift map." The "core" of these scripts is the new subroutine, "list_minicls_shift_respos_clm_based" (in the module, "MyPosShiftMap.pm"). Appendix B describes the algorithmic aspect of the subroutine.

$D_{exp}(b,c)$, should be the average, over initial residue states, of the SRDs predicted by the finite-time transition matrix, weighted by the expected frequencies of the residue states at the branch-ends. Because each branch has an upper-end and a lower-end, we calculate such an average from each end, and take their arithmetic mean.

Mathematically, the aforementioned recipe can be expressed as follows. Let $D_{exp(U)}(b,c)$ and $D_{exp(L)}(b,c)$ be the aforementioned average theoretical expectations from the upper-end and the lower-end, respectively. Then, they are calculated in the following steps. First, the "theoretical expectation" from the upper-end is given as:

$$D_{exp(U)}(b,c) = 1 - \sum_{\omega\ \in\Omega} \left\{ P_{(U)}[\ \omega \mid n_U(b), c\ ]\ P(\omega \mapsto \omega' = \omega; b) \right\}\ . \tag{4}$$

Here, the $P_{(U)}[\ \omega \mid n_U(b), c\ ]$ is the frequency of ω ($\in \Omega$) at the upper-end of branch b (denoted as $n_U(b)$), expected from the residue configuration of the sequences on the upper-end-side of b in column c. It is defined as:

$$P_{(U)}[\ \omega \mid n_U(b), c\ ] \stackrel{\text{def}}{=} P_U(\omega; b, c) \Big/ \left\{ \sum_{\omega'\ \in\Omega} P_U(\omega'; b, c) \right\}\ . \tag{5}$$

The "theoretical expectation" from the lower-end is somewhat difficult:

$$D_{exp(L)}(b,c) = 1 - \sum_{\omega\ \in\Omega} \left\{ P_{(L)}[\ \omega \mid n_L(b), c\ ]\ P^T(\omega \mapsto \omega' = \omega; b) \right\}\ . \tag{6}$$

Here, the $P_{(L)}[\ \omega \mid n_L(b), c\ ]$ is the frequency of ω ($\in \Omega$) at the lower-end of branch b (denoted as $n_L(b)$), expected from the residue configuration of the sequences on the lower-end-side of b in column c. It is defined as:

$$P_{(L)}[\ \omega \mid n_L(b), c\ ] \stackrel{\text{def}}{=} P(\omega; n_L(b))\ P_L(\omega; b, c) \Big/ \left\{ \sum_{\omega'\ \in\Omega} P(\omega'; n_L(b))\ P_L(\omega'; b, c) \right\}\ . \tag{7}$$

Here, the $P(\omega; n_L(b))$ denotes the frequency of ω ($\in \Omega$) at the lower-end of b, purely expected by the substitution model (and the residue frequencies at the root (n^R)); such frequencies can be obtained by evolving the residue frequencies at the root (denoted as $\left\{P(\omega; n^R)\right\}_{\omega\ \in\Omega}$) according to $P(\omega' \mapsto \omega''; b')$ along the branches (b's) from the root to $n_L(b)$. Back to Eq.

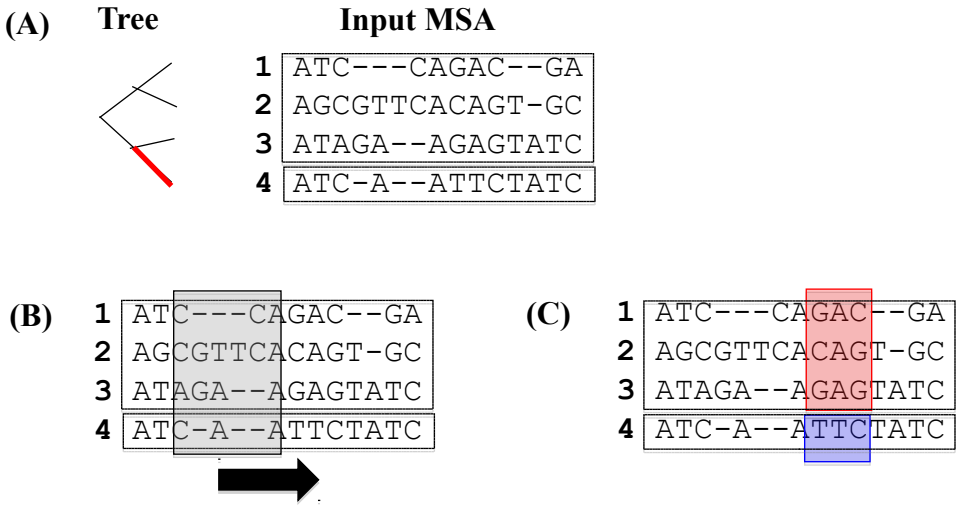

Figure 3: **Sliding-window analysis to identify mis-alignments (or "purge"-likee errors). A.** In this example, we search for possible mis-alignments (or "purge"-like errors) between the two sub-alignments enclosed by dashed rectangles, which are mutually separated by the red thick branch. **B.** The window (shaded gray) contains a user-specified number (3 here) of residue pairs, and slides from left to right (the right-headed arrow). **C.** This red-and-blue-shaded window shows more predicted substitutions than expected from the branch length. Thus, the program suspects that a mis-alignment (or a "purge"-like error) is likely in this region.

6, the $P^T(\omega \mapsto \omega'; b)$ is the "*time-reversed* transition probability" (from ω at $n_L(b)$ to ω' at $n_U(b)$); it is defined as:

$$P^T(\omega \mapsto \omega'; b) \stackrel{\text{def}}{=} P(\omega'; n_U(b))\ P(\omega' \mapsto \omega; b) \Big/ P(\omega; n_L(b))\ . \tag{8}$$

Taking the arithmetic mean of Eqs. 4 & 6, we get the final value of the "theoretically expected SRD" (along branch b in column c):

$$D_{exp}(b, c) = \left\{ D_{exp(U)}(b, c) + D_{exp(L)}(b, c) \right\} \Big/ 2\ . \tag{9}$$

Now, we set the **sliding windows** (Figure 3). Because each mis-alignment (or "purge"-like error) typically causes a "shear" (or displacement) between the upper- and lower-end-sides of a branch (Figure 1), it should be better to perform a series of sliding window analyses (along the input MSA) on each fixed branch, and repeat such analyses to cover all branches. On each branch (b), the window (denoted as W) is set so that it will contain a user-specified number (say, N_W) of columns in which the upper- and lower-ends of b are both in the

"presence" state (*i.e.*, occupied by some residues). [3] , [4] Then, the "observed" and "expected" SRDs in the window (W), denoted as $D_{obs}(b, W)$ and $D_{exp}(b, W)$, respectively, are simply the summations of the corresponding column-wise values over the columns *constituting* W:

$$D_{obs}(b, W) = \sum_{c \in W} D_{obs}(b, c) \ , \tag{10}$$

$$D_{exp}(b, W) = \sum_{c \in W} D_{exp}(b, c) \ . \tag{11}$$

Then, we **compare** $D_{obs}(b, W)$ **to** $D_{exp}(b, W)$; if the former is *significantly* larger than the latter, we conclude that a **mis-alignment** (or a **"purge"-like error**) is likely in the window (W) along the branch b. Probably, a simplest way to assess the significance should be to calculate the P-value of the "observation" under the binomial distribution with the number of "trials", N_W, and with the probability of "difference" per trial, $p_D \stackrel{\text{def}}{=} D_{exp}(b, W)/N_W$:

$$P[D_{obs} \geq D_{obs}(b, W)] = \sum_{x=D_{obs}(b,W)}^{N_W} \frac{(N_W)!}{x!(N_W - x)!} (p_D)^x (1 - p_D)^{N_W - x} \ , \tag{12}$$

where $n!$ $(= n \cdot (n-1) \cdots 1$ denotes the factorial of n. [5] We can set a threshold P-value

[3] Here, the "presence"/"absence" states (*i.e.*, residue/gap states) at internal nodes are determined by the **Dollo parsimony principle** [60]. This is because the "presence" states in the Dollo parsimonious indel history are certain to be "present" in any of the indel histories that satisfy the *phylogenetic correctness condition* [61, 62]. Thus, if both ends of a branch are in the "presence" state in the Dollo parsimonious history, we can be sure that, on each side of the branch, at least one sequence has a residue in the column in question; hence examining the SRDs in that column will make sense.

[4] Those columns in each of which either end (of b) has an "absence" state are excluded from the analysis. At least theoretically, however, they can be *just physically* encompassed by the window(s) (Figure 3 B). (Nevertheless, the current version of ANEX does *not* allow each window even to physically encompass these columns; in other words, ANEX currently prohibits these columns from mediating the constituent columns of each window. This means that windows as shown in Figure 3 B are actually *excluded* from the downstream analyses in the current ANEX. This is not for technical reasons, but rather to avoid conceptual complications. In the future, this restriction may be loosened.)

[5] When the $D_{obs}(b, W)$ is *not* an integer, we could either interpolate Eq. 12 (with integers) or resort to the analytic continuation of the corresponding formula using an incomplete beta-function (*e.g.*, [63]).

(P_{Thrsh}), and regard the window as potentially harboring a mis-alignment (or a "purge"-like error) along b if Eq. 12 is less than P_{Thrsh}.

The aforementioned method should be very useful if the substitution rate is (nearly) uniform along the sequence. If, however, the substitution rate is expected to vary across sites, due to, *e.g.*, selection and/or mutation hotspots, it may be better to take account of such rate variation. One way would be to use a model that *explicitly* incorporates rate variation across sites (*e.g.*, [64, 65, 66, 59, 67]), and reformulate the above series of calculations. In appendix C, we devise a more heuristic method.

2.2.1 Additional filtering based on random-matching model

The aforementioned method works well when applying to the branch that is not very short. When the branch is very short, however, it could regard each *true* substitution as a candidate of mis-alignment (or "purge"-like error), resulting in lots of "false-positive"s.

Hence, it would be desirable if we have a method to filter a substantial fraction of such "false positive"s while keeping most of true positives. One way would be to filter the windows *via another* **P-value** (again regarding the SRDs), which is defined under a **random-matching model**. The rationale for this is as follows. First, the "purge" errors are expected to occur in general by falsely eliminating a pair of neighboring complementary indels at the expense of generating false substitutions. Thus, roughly speaking, the "false-homologous blocks" caused by a "purge" should be like an alignment of two random segments (or two *non-homologous* sub-alignments). The *original* P-value (calculated, *e.g.*, *via* Eq. 12) attempts to identify "likely false-homologous pairs" that show *significantly more SRDs than* expected under a given substitution model. Along a short branch, however, this measure is likely to pick even a window showing only one (or two) substitution(s). In contrast, the **new P-value** attempts to identify "likely **true**-homologous pairs" that show significantly less SRDs *than* expected under the random matching model, like BLAST does (see, *e.g.*, [68]).

To be more mathematically specific, the SRD "expected" from a random matching model, denoted as $D_{exp\ (rand)}(b, W)$, is given by:

$$D_{exp\ (rand)}(b, W) \;=\; N_W \left\{ 1 - \sum_{\omega\ \in \Omega} p_{exp}(\omega; n_L(b), W)\ p_{exp}(\omega; n_U(b), W) \right\} . \qquad (13)$$

Here, $p_{exp}(\omega; n, W)$ denotes the expected probability (or relative frequency) of residue ω in window W at node n. (Remember that $n_L(b)$ and $n_U(b)$ denote the lower- and upper-ends, respectively, of branch b.) There are a number of possibilities for the $p_{exp}(\omega; n, W)$'s. The two simplest options among them would be: (1) the residue frequencies, $P(\omega; n)$'s, expected from the substitution model (Θ_S) and the frequencies at the root ($P(\omega; n^R)$'s); and (2) the "observed" residue frequencies, $P_{(L)}[\omega \mid n_L(b), c]$'s (in Eq. 7) and $P_{(U)}[\omega \mid n_U(b), c]$'s (in Eq. 5), averaged over the columns (c's) in the window (W). The current version of ANEX [2] uses the former, *i.e.*, $p_{exp}(\omega; n, W) = P(\omega; n)$. Then, the P-value employed for this additional filtering is the probability:

$$P[D_{obs} \leq D_{obs}(b, W)] = \sum_{x=0}^{D_{obs}(b,W)} \frac{(N_W)!}{x!(N_W - x)!} (p_{D(rand)})^x (1 - p_{D(rand)})^{N_W - x} , \qquad (14)$$

where $p_{D(rand)} \stackrel{\text{def}}{=} D_{exp\ (rand)}(b, W)/N_W$. (Footnote 5 applies also here.) In this additional filtering, if Eq. 14 is *less than* a given threshold (denoted, *e.g.*, as $P_{Thres\ (rand)}$), the window is considered to be "significantly better-matching than random" (in other words, "likely correctly aligned"), and is *excluded* from the mis-alignment candidates.

Because the random-matching model is nearly independent of the branch length, the new P-value is expected to give filtering that is *effectively* "orthogonal" to that *via* the original P-value, even though both of them are *defined* in terms of the same measure, *i.e.*, the SRD.

We validated the original and this additional screening methods. The results are described in sub-subsection 3.2.2 below.

2.3 Exploiting gap-configurations to detect "complex" errors

In the previous subsection, we devised methods to detect **mis-alignments** (or **"purge"-like errors**) by taking advantage of the residue-configurations of input MSAs. In this subsection, we attempt to detect **"complex" errors** (explained below) by exploiting the gap-configurations of input MSAs. This is an *extremely* heuristic endeavor, because currently there are no established theories, or models, to deal with (or describe) the "complex errors"; this is not so surprising because of the elusive nature of the "complex" errors; as the method to classify MSA errors advances, the definition of "complex" errors will change, too. This means that the specific method, especially parameters, etc, that we found to be useful in this study may *not* apply in the (maybe near) future. Still, we believe that it's worth recording this endeavor here, because some principles or philosophy itself (especially of exploiting gap-configurations) may continue to be useful.

In a previous simulation study [1], we attempted to classify errors (or mis-alignments) in reconstructed MSAs into various types, such as "shift"s, "merge"s, "split"s, "purge"s, etc., by extending the past attempt on PWA errors [27]. And those MSA errors that could not be classified into fixed types, or combinations of fixed types, in that study were put into the category of **"complex" errors**. Therefore, by definition, the category of "complex" errors is a mixture of various types of errors, with the only shared nature being "too complex to be unraveled into a combination of definite classes by the current method." [6] Moreover, the erroneous segments that are "too long" [7] were also put into the category of "complex" errors.

[6] The "current method" here means the classification method used in [1].

[7] In [1], if each gapped segment is longer than 150 bases or if it contains a run of gaps longer than 120 bases, it was considered "too long" and was excluded from the analyses. The erroneous segments(, in which the reconstructed MSA is not equal (or equivalent) to the true MSA,) were classified as "too long" if each of them contains at least one such "too long" gapped segment.

2.3.1 Characterization of "complex" errors

Before devising any concrete method for detecting "complex" errors, we characterized them via meticulous analyses, in which the "complex" errors were compared with a set of **"composite" controls**, which consists of correctly reconstructed segments and non-"complex" errors. [8]

Briefly, we found the following broad tendencies:

(i) the properties (*e.g.*, size, number of gap-blocks, total horizontal length of gap-blocks, number of insertions minus the number of deletions) of *individual* gapped segments included in "complex" errors are *slightly* different from the properties of those included in the "composite" controls, but *not clearly enough* to sharply separate the two categories;

(ii) *in contrast*, the properties (*e.g.*, the number of gapped segments, total number of columns, maximum number of indels, etc.) of each "complex" error *as a whole* showed *marked* differences from the properties of each control segment *as a whole*.

These results implied that, *once correctly partitioned*, the properties of the ***entire*** **segments** can be exploited to detect the "complex" errors. The question is: how can we (quite) *accurately* cluster the gapped segments into the "erroneous" and correctly aligned segments?

Therefore, we next examined the sizes of spacers, which are gapless segments that mediate neighboring gapped segments. We found that the spacer size distributions differ *notably between* the spacers lying *within* individual (erroneous or correct) segments and those *mediating* different (erroneous or correct) segments. However, within each of the above two categories, the distributions did *not conspicuously* depend on whether the flanking gapped segments are in "complex" errors or not. These results indicated that the **spacer size difference** could be exploited at least to ***artificially*** **cluster** the gapped segments into *likely* erroneous and correct segments, with some (moderate) accuracy.

[8] For details on these meticulous analyses, see "suppl2_blueprint1_ANEX.pdf," which is among the documents accompanying the "ANEX(_P)" program package [2].

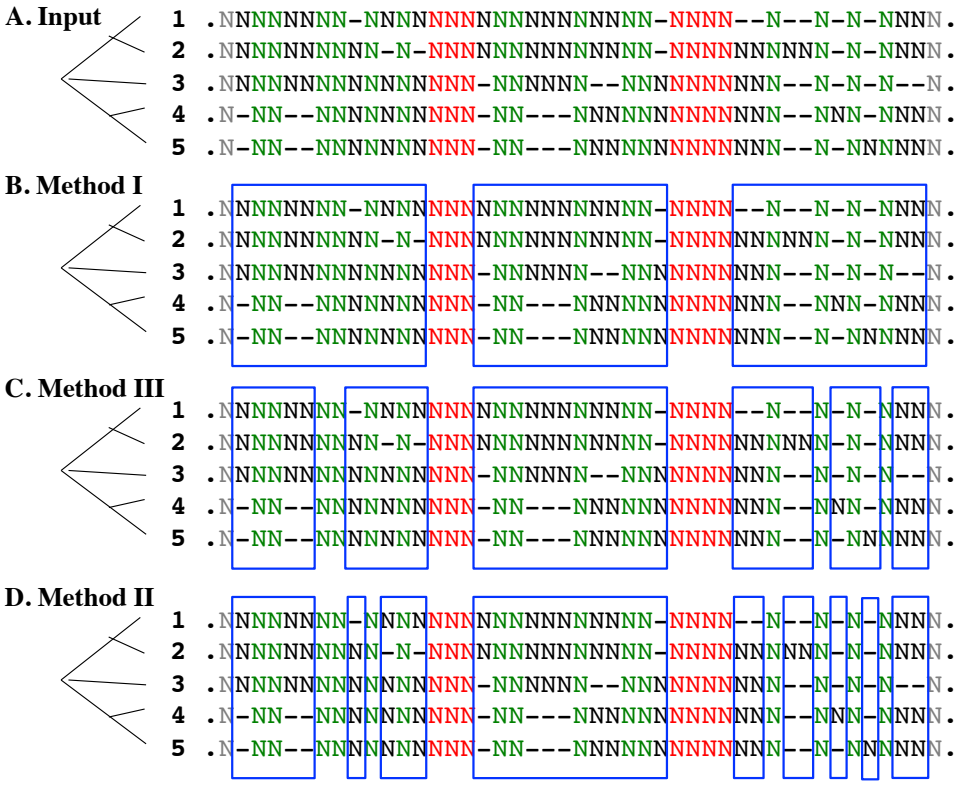

Figure 4: **Methods to artificially cluster gapped segments.** This figure illustrates the results of different clustering methods applied to the same fictitious MSA (panel **A**, on the right), which resulted from an indel history along a given tree (panel **A**, on the left). **B.** Result of Method I. **C.** Result of Method III. **D.** Result of Method II. See the text and appendix D for details on these methods. To focus on the topological issue, we assume that {spacer-size threshold} = {spacer-size upper-bound}. Then, the 2nd half of condition (a) for Methods II & III can be ignored, hence the condition on the spacer-size becomes identical to that for Method I. Here, for illustration, the {spacer-size threshold} is assumed to be 2. In each panel, the spacers (*i.e.*, gapless segments) are colored; red indicates that the spacer-size is larger than the threshold, and green indicates otherwise. Each blue rectangular box encloses an artificial cluster of gapped segments.

2.3.2 *Artificially* clustering gapped segments

In the real-life sequence studies, no true MSAs are available. Therefore, *nobody* can tell, *with absolute certainty*, how to partition each reconstructed MSA into correct and "erroneous" segments. The only thing we can do is to devise a method to ***artificially* cluster** the gapped segments, to approximate, *as accurately as possible*, the *true* partitioning of the input MSA into correct and "erroneous" segments. If a good method is invented, it will enhance the accuracy of detecting "complex" errors, as the results in sub-subsection 2.3.1 indicates.

We tried three different methods (see Figure 4 for schematic illustration). In a simple

clustering method (**Method I**), two nearest-neighboring gapped segments are clustered if the spacer between them is shorter than a threshold value, *regardless of the gap patterns of the gapped segments* (Figure 4 B). **Methods II** and **III** take account of gap patterns in addition to spacer sizes. **Method III** clusters two quite close gapped segments if they undergo (effective-)indels along the same branch or along two phylogenetically nearest-neighboring branches (Figure 4 C). **Method II** is similar to Method III, but the requirement on the gap pattern is stricter and more complex; more precisely, Method II clusters two (not necessarily nearest-neighboring) gapped segments, as well as all segments in between them, if they are quite close to each other and, *additionally*, *either* if they undergo indels along the same branch *or* if all three branches connected at an internal node undergo (effective-)indels in these gapped segments and *yet another* neighboring segment (Figure 4 D). For details, see appendix D.

After artificially clustering the gapped segments with each of these three methods, we attempted to detect "complex" errors *as accurately as possible*, using the number of insertions and the number of deletions. See appendix E for details on this attempt.

The results of validating these methods are described in sub-subsection 3.2.3 below.

3 Implementing and Validating Ideas and Methods

3.1 Implementation

Most of the methods described in section 2 have been implemented as supplementary Perl scripts and/or subroutines in the ANEX(_P) program package [2]. For example:

- The methods described in subsection 2.1 have been implemented into a main Perl script, "detect_purge_cands.ver0.5.pl," as well as a couple of its supplementary Perl scripts including "srd_map.ver0.6.pl."

- The methods described in subsection 2.2 have been implemented as the subroutine,

"detect_purge_cands2," in the module, "MyDetect_purge_cands.pm"; actually, the subroutine is incorporated into the main script, "detect_purge_cands.ver0.5.pl."

- The methods described in subsection 2.3 have been implemented into subroutines in the module, "MyDetect_cmplx_error_cands.pm"; see especially those lumped together into the subroutine, "wrapper_detect_cmplx_error_cand_acls0."

ANEX(_P) is an open-source package available at an FTP repository of Bioinformatics.org (https://www.bioinformatics.org/ftp/pub/anex/).

3.2 Validations

3.2.1 Validating SRD Map

To test the potential usefulness of substitutional-residue-difference maps (SRD Maps) introduced in subsection 2.1, we compared the SRDs on the mis-alignments with those off the mis-alignments, using the reconstructed MSAs (via Prank [18, 19]) of simulated 15 mammalian sequences and those of simulated 12 primate sequences that we created previously [1]. (See the sited paper for details on these reconstructed MSAs.) Each set has $10,000 \times 1,000 = 10,000,000$ bases in the ancestral sequences at the root. For this purpose, we divided the points in "extended" MSAs, each of which is specified by an MSA column and a branch, into three categories: (1) the points in the correctly aligned segments (or "correct segments" for short), (2) those in the erroneous segments and off any "alignment-shear"s, and (3) those in the erroneous segments and on some "alignment-shear"s. [9]

The tables in Figures 5 & 6 show the mean values and the standard-deviations of SRDs along individual branches, as well as those for all relevant points. [10] (The former figure is for simulated 12 primates and the latter is for simulated 15 mammals.) As both

[9] The "alignment-shear"s were inferred by using the method described in appendix B (and in footnote 2). When an erroneous segment of a reconstructed MSA has multiple "optimum" "alignment-shear map"s, we took the arithmetic mean of the results of using all such maps.

[10] More detailed data, including the distributions of the SRDs (on and off the "alignment-shear"s) along

Branch_ID	Mean[SRD/site]_on_as	Std_Dev[SRD/site]_on_as	Mean[SRD/site]_off_as	Std_Dev[SRD/site]_off_as	Mean[SRD/site]_corr	Std_Dev[SRD/site]_corr
seq0000	0.0585719039602963	0.228770072748974	0.00542430352669682	0.0717411293829225	0.00537687469637146	0.0723451813009779
seq0001	0.0692438210165759	0.250209094771225	0.00796286462828743	0.0875013607329981	0.00800520907867604	0.0885016002763661
seq0002	0.142523225461632	0.344499739508978	0.0216285040717586	0.143863609991046	0.0211740534644934	0.143165725559794
seq0003	0.146583731413101	0.348616938920735	0.0265971560199577	0.158779998035732	0.0262955141791694	0.15894633590205
seq0004	0.116434317614634	0.31559343200115	0.0189752809882638	0.132978364985406	0.0185399188598393	0.133214419197694
seq0005	0.112006918320023	0.311124240912249	0.018882607881548 5	0.135240611216385	0.0186324569955062	0.134756745003324
seq0006	0.101439056821394	0.29735716906166	0.0125286958569454	0.110159309454928	0.0122053516490609	0.109300487572008
seq0007	0.090505184989129	0.280867099858024	0.0109719059385247	0.102984321229293	0.0106670897584447	0.102185512282284
seq0008	0.169736905906418	0.367177965566591	0.0363589572060993	0.17956705266836	0.0354592234643679	0.181446219987537
seq0009	0.168113670116898	0.362928124574012	0.0282371590382203	0.161999081043853	0.0278971318215356	0.163333844020512
seq0010	0.211721321065521	0.394827106797261	0.0344700452533934	0.17721588054206	0.0322649394002035	0.17471905875793
seq0011	0.234114941232742	0.410311529085333	0.0381842055827389	0.186947342455297	0.0362533861666477	0.185073284426626
3	0.110002957757475	0.301187274321632	0.00994204159147204	0.0954221959970022	0.00959581173748922	0.0958457823524223
5	0.0613555146815926	0.215746822097747	0.00314211474717747	0.047307602995157	0.00268477816977931	0.0475876274445924
7	0.103581775467927	0.277510803552823	0.0114224604041327	0.0895147783649182	0.010650874197863	0.0947813912223342
12	0.0515624276085036	0.204122164469496	0.00193332687199403	0.0393821675638031	0.00156442283533109	0.0372931626826542
13	0.0820722382550376	0.257534795239161	0.00553195615977463	0.0663194131977624	0.00530830992385186	0.0688663629286928
14	0.149509583618087	0.343457320080124	0.0276884543063232	0.152691321475366	0.0263546150419185	0.155023908254562
15	0.0732881185141797	0.142652230063373	0.0224872791926701	0.0832231018380719	0.0199640711985266	0.0803323153286055
20	0.0677152246167858	0.216219233437354	0.00522770985745852	0.0575209582993755	0.00265466950915737	0.0439615636518321
21	0.0795952905304279	0.236674031098415	0.00724587115328852	0.0654725376792165	0.00639598144226506	0.0702221397453032
22	0.132486839130837	0.256102787531447	0.0398181161848339	0.148357172567637	0.0354504928998687	0.143457377062973
Total	0.148119380051356	0.333625631790149	0.0174631174338563	0.121209370028309	0.016941356269237	0.122185775517233

Figure 5: **SRDs along individual branches, on and off misalignments (12 primates).** This table shows the results of the analysis on 10,000 reconstructed MSAs of simulated 12 primate sequences (created in [1]). It shows the mean values ('Mean') and the standard deviations ('Std_Dev') of SRDs (per site) along individual branches (row), at the points that are on some "alignment-shear"s ('_on_as'), off any "alignment-shear"s ('_off_as'), and in the correctly aligned segments ('_corr'). The branches are identified in the same way as in Figure 2. However, the key to the integer IDs is somewhat different (again, the left-hand-side is the branch ID, and the right-hand-side is a set of indexes (from 0 through 11) of MSA rows under the branch): 3 = {0, 1}; 5 = {0, 1, 2}; 7 = {0, 1, 2, 3}; 12 = {6, 7}; 13 = {5, 6, 7}; 14 = {4, 5, 6, 7}; 15 = {0, 1, 2, 3, 4, 5, 6, 7}; 20 = {10, 11}; 21 = {9, 10, 11}; 22 = {8, 9, 10, 11}.

Branch_ID	Mean[SRD/site]_on_as	Std_Dev[SRD/site]_on_as	Mean[SRD/site]_off_as	Std_Dev[SRD/site]_off_as	Mean[SRD/site]_corr	Std_Dev[SRD/site]_corr
seq0000	0.0582477475453448	0.225055618020865	0.0053095297810697	0.0704752650662919	0.00535726856664855	0.0712516960889862
seq0001	0.0701829917340279	0.248185051885182	0.00811394752676058	0.0879733408133466	0.00802897177716349	0.0878512689024305
seq0002	0.121342305350953	0.317542104971307	0.0187425962244411	0.133001330219085	0.0185200519573262	0.132680334125069
seq0003	0.103231160978931	0.297837486321298	0.0123828042247729	0.10921574812478	0.0122718633576091	0.10899735529213
seq0004	0.082873610980697	0.266619757873 41	0.0106598208968812	0.101194498060334	0.0106005903998308	0.101214175707303
seq0005	0.188148587390586	0.375961557310863	0.0357383739842089	0.180965036518533	0.035457699287038	0.181233555946405
seq0006	0.187728325847169	0.375107940226726	0.0277102861212536	0.159929421935166	0.0275965733747627	0.160555613885123
seq0007	0.214809493277032	0.399361620038201	0.0351098703433248	0.180437272216037	0.0348921045050796	0.180670121399388
seq0008	0.279608572893604	0.406752117531249	0.0786631908845502	0.243547566784618	0.0772070794197505	0.249912152379377
seq0009	0.328750610690611	0.441245031189215	0.114162590373745	0.298036774252126	0.112361068536955	0.302376237274599
seq0013	0.398489333079716	0.450018000778644	0.155640900767366	0.337362080643522	0.148073015012369	0.342813872154747
seq0014	0.380824660621264	0.441799892615947	0.142445469695497	0.32295982986385	0.135298996432066	0.329062472036722
seq0015	0.367078314846223	0.4470928923962	0.131769576379536	0.316058580646306	0.128962041728101	0.324575182295388
seq0018	0.336665359867295	0.438929643425626	0.150805784448118	0.328740684800703	0.147328181264902	0.33530002234094
seq0020	0.285671069517751	0.393725488022331	0.144445406161442	0.308091779103276	0.132419123776591	0.305129662303974
3	0.151171415431121	0.339946332523945	0.0227577680547509	0.142202745130944	0.0225970082471617	0.143426039854482
7	0.0761142645945805	0.246454145673592	0.00703150124037856	0.0783974187082907	0.00688629426690577	0.0786755637759065
8	0.169879911963646	0.357649562426633	0.0267270366231989	0.154009180203397	0.0261940048633716	0.15418319586069
9	0.139554204637226	0.305216859762018	0.0203459900549309	0.1222170391994	0.0197277000371691	0.125294000819899
13	0.084788929586549	0.239038979781985	0.00680904763090217	0.0692536551948053	0.00643084442476664	0.0698201928750626
14	0.177768541229595	0.350475574045665	0.0360695602850378	0.171270086303879	0.0353638162723446	0.173433356780599
15	0.260724678021851	0.393156099933647	0.0672489855766178	0.224792477112653	0.0649840194224802	0.229934592303994
18	0.15625239568091	0.285863393909065	0.0376988801721454	0.145349197042199	0.0316539993971713	0.143144271939311
19	0.166465399177065	0.303801527465928	0.0413837678672779	0.156265248490749	0.0362803695918429	0.16038614221252
22	0.0310240634198185	0.0729100931706894	0.009244751699603 32	0.0397726061617495	0.00460245310745152	0.0279397197880864
24	0.134488931842884	0.263611370822548	0.0382086295992386	0.139751151189003	0.0280048378286076	0.133878249393964
25	0.109255151004534	0.214222824446432	0.0326981215404141	0.114027860484179	0.0288689035811588	0.120068327534517
27	0.026354391219146	0.0354302297476264	0.0134771228573646	0.0273135906383073	0.0123651702124714	0.0270265107288709
Total	0.302986371097 31	0.419612736798676	0.0482173328564271	0.193552883733746	0.0482950494732039	0.200699340763208

Figure 6: **SRDs along individual branches, on and off misalignments (15 mammals).** This table shows the results of the analysis on 10,000 reconstructed MSAs of simulated 15 mamalian sequences (created in [1]). The same notes as in Figure 5 apply, except that the key to the integer IDs must be the same as in Figure 2.

tables indicate, the SRDs off the "alignment-shear"s are very similar to those in the correct segments, and the SRDs on the "alignment-shear"s are *substantially* larger *than* those off the "alignment-shear"s (or in the correct segments). Actually, because the number of examined points are quite large, these differences are highly significant. [11] This indicates that it should be somewhat promising to use the SRDs to identify mis-alignments (or "purge"-like errors).

There is, however, a ***caveat***, derived from two observations. First, although the mean SRDs significantly differ between points on and off "alignment-shear"s, the standard deviations of SRDs are *much* larger than the means (Figures 5 & 6). Second, although the mean SRDs on "alignment-shear"s are significantly larger than those off "alignment-shear"s, their values are *much* smaller *than* 0.75, which is the average SRD expected from the random alignment of two sequences (under Jukes and Cantor's base substitution model [69] used in the simulations [1]). These observations implies that the things may not be so simple as we theoretically expected (in subsections 2.1 and 2.2). Inspecting the distributions of SRDs (on and off "alignment-shear"s) (available elsewhere, see footnote 10), we noticed that most of the SRDs are distributed in two regions: (1) near zero (*e.g.*, SRD ¡ 0.005) and (2) near one (*e.g.*, SRD $\geq$ 0.9, and that, *even* on "alignment-shear"s, the fraction of SRDs near zero is *much* larger *than* 1/4. For example, in the total distribution of SRDs for simulated 15 mammals, about 50% and about 22.4% of SRDs on "alignment-shear"s are near zero and near one, respectively; in contrast, about 85% and about 3.2% of SRDs off "alignment-shear"s are near zero and near one, respectively. [12] Therefore, the significant differences in the mean SRDs between on and off "alignment-shear"s seem to have come from the substantial

individual branches, are available as a tar-gzipped archive accompanying the "ANEX(_P)" package [2].

[11] For example, when the SRDs on and off the "alignment-shear"s are compared (using the normalized Z-value), the P-values of the differences are less than 10^{-10} for all comparisons (along the branches).

[12] In the total distribution for simulated 12 primates, about 80% and about 11.6% of SRDs on "alignment-shear"s are near zero and near one, respectively; in contrast, about 96.8% and about 1.2% of SRDs off "alignment-shear"s are near zero and near one, respectively.

differences in the fractions of SRDs near one, which should guarantee that the SRD Map (devised in subsection 2.1) and the sliding-window analysis based on SRDs (devised in subsection 2.2) *should* work *to some extent*. On the other hand, the fairly large fraction of SRDs near zero *even* on "algnment-shear"s implies that these SRD-dependent methods should *not* be perfect, because a considerable fraction of **mis-alignments** should have SRDs *nearly* equal to, or *even* smaller than, those in the correct alignments.

Therefore, in order to detect the latter type of mis-alignments, we need to examine **different features** than SRDs. One promising feature is the **repeat** of sub-sequences (or a single- or oligo-nucleotide) *near* the ends of gaps; in short, if a single- or oligo-nucleotide that flanks a run of gaps occur also near the end of the run of gaps, the chances are high that the single- or oligo-nucleotide is mis-aligned with the other one. Thus, by examining this feature, some fraction of mis-alignments are likely to be detected. Although current version of ANEX [2] does not exploit this feature, taking account of it may somewhat improve the performance of ANEX, or other software that (implicitly or explicitly) attempts to correct alignments. [13]

3.2.2 Validating method to identify likely mis-alignments (or "purge"-like errors)

To validate the method to identify likely mis-alignments (or "purge"-like errors *via* sliding-window analyses (described in subsection 2.2), we applied the two filtering methods (described around Eq. 12 and around Eq. 14), with various P-value thresholds, to the true "purge"-errors (subjects) and the correctly aligned segments (controls) in the 10,000 reconstructed MSAs (*via* Prank) of simulated 15 mammalian sequences.

The 2D tables in Figures 7 & 8 summarize the results. [14] Let us first consider the

[13]The author discovered that these "repeat"-caused mis-alignments are quite frequent, when he was participating in the project led by Dr. Graur and Dr. Landan (see Acknowledgements). The author is very grateful to them for having given him an opportunity to make such a discovery.

[14]More detailed results of the analyses, as well as the characterization of the true "purge"-errors,

	Subjects		Controls	
	P(sbst) < 0.05	**P(sbst) < 0.20**	**P(sbst) < 0.05**	**P(sbst) < 0.20**
(No further condition)	0.495	0.780	0.0088	0.057
P(rand) ≥ 0.05	0.478	0.747	0.0061	0.031
P(rand) ≥ 0.20	0.433	0.664	0.0042	0.018

Figure 7: **Effects of two filtering methods to identify "purge"-like errors: on *all* subjects and controls (*with* block size $= 1, 2, ..., 10$ (siets)). NOTEs:** Shown in each cell is the relative frequency of "purge"-involved blocks (in the subjects) or windows (in the controls) satisfying the specified condition, in the set of 10,000 reconstructed MSAs of simulated 15 mammalian sequences. The "P(sbst)" and "P(rand)" denote, respectively, the (old) P-value (Eq. 12), which is defined with a given base substitution model, and the (new) P-value (Eq. 14), which is defined with a random matching model.

	Subjects		Controls	
	P(sbst) < 0.05	**P(sbst) < 0.20**	**P(sbst) < 0.05**	**P(sbst) < 0.20**
(No further condition)	0.649	0.895	0.0094	0.065
P(rand) ≥ 0.05	0.623	0.844	0.0054	0.027
P(rand) ≥ 0.20	0.552	0.713	0.0025	0.0076

Figure 8: **Effects of two filtering methods to identify "purge"-like errors: on subjects and controls *with* block size ≥ 3 (sites).** The same notes as in Figure 7 apply.

results on *all* subjects and controls (Figure 7), which include those windows with sizes 1 through 10 (sites). When we used the old P-value ("P(sbst)", calculated with Eq. 12) *alone*, 49.5% and 78.0% of the true "purge"-errors were detected when the false-positive rate was 0.88% and 5.7%, respectively. Then, the *additional* filtering using the new P-value ("P(rand)", calculated with 14) *did* refine the (mixed) set of "positive" windows; *that is*, the additional filtering successfully shed a substantial fraction (31-68%) of the false-positives, while keeping most (85-97%) of the true-positives. Thus, the additional filtering based on the random-matching model (in sub-subsection 2.2.1) did indeed work as we originally intended.

When we restricted the analysis to the windows greater than or equal to 3 (sites) (Figure 8), the filtering with the old P-value *alone* had the true-positive rate enhanced to 64.9% and

are described in "suppl2_blueprint1_ANEX.draft5.pdf," which is among the documents accompanying the ANEX(_P) package [2].

89.5% when the false-positive rate remained nearly the same (0.94% and 6.5%, respectively). Moreover, the additional filtering using the new P-value became more effective.

3.2.3 Validating method to detect "complex" errors

To validate the methods to detect (or rather *exclude*) "complex" errors, which has been described in subsection 2.3 and appendixes C & E, we applied the detection method, with each of the artificial clustering methods, **Methods I, II, and III** (appendix C), to the 10,000 reconstructed MSAs (*via* Prank) of simulated 15 mammalian sequences, while changing the values of the two parameters, **{spacer-size threshold}** and **{spacer-size upper-bound}** (in appendix D). Regardless of the artificial clustering methods, the total number of gapped segments in category B (*i.e.*, control) is 589,021, the total number of gapped segments in category A (*i.e.*, "complex" errors) is 416,766, and the ratio of the latter to the former is 0.707557. (Before this analysis, a "complex?"erroneous segment was re-classified as "non-complex" if only less than 4 (effective-)indels were inferred from each of the reconstructed and reference MSAs.)

The tables in Figures 9 & 10 & 11 summarize the results of the analysis. [15] In our analyses, **Method III** performed the best (Figure 11), Method I performed the worst (Figure 9), and Method II performed slightly less efficiently than Method III (Figure 10), under the respective optimum combinations of the two parameters. Following this result, we set **Method III** as the default *artificial* clustering method in ANEX(_P).

It should be worth mentioning, however, that, *even* with the best method (Method III), the accuracy of detecting (or excluding) complex errors is not pleasantly high, though decent (or barely acceptable): when *only* about 80% of control gapped segments are kept, the "optimum" method excluded about 62-63% of segments in "complex' errors; and when about 90% of control gapped segments are kept, the "optimum" method excluded *only* about

[15] For more details on this validation (or parameter tuning), see "suppl2_blueprint1_ANEX.draft5.pdf," as well as some Excel spreadsheets, all of which accompany the ANEX(_P) package.

Spacer-size upper-bound	Spacer-size threshold	Tot# {pure-B}	Tot# {pure-A}	Tot# {mixed}	Target %{in B}	Target %{in A}	Target#{in A} / Target#{in B}
---	2	558793	378425	68389	76.25%	41.31%	0.3833
					82.03%	47.25%	0.4076
					88.42%	56.74%	0.4540
					90.22%	60.59%	0.4752
---	3	534389	343086	128312	78.36%	41.23%	0.3723
					81.15%	45.27%	0.3947
					89.53%	60.46%	0.4778
					90.49%	62.73%	0.4905

Figure 9: **Performance of Method I to *artificially* cluster gapped segments, followed by method to exclude "complex" errors.** [**NOTE:** For each of the three methods (Methods I, II, and III), only the results with the optimum and the near-optimum combinations of the parameters (**{spacer-size threshold}** and **{spacer-size upper-bound}**) are shown here.] The best-performing results are highlighted in red, and relatively well-performing results are highlighted in yellow. In this and the subsequent two figures, the **category A** consists of gapped segments in "complex" errors, and the **category B** consists of "control" gapped segments (, which are correct segments and segments in non-"comlex" errors). **KEY: Tot#{pure-A/B}** = the total number of gapped segments belonging to the artificial clusters consisting only of gapped segments of category A/B; **Tot#{mixed}** = the total number of gapped segments belonging to the artificial clusters each of which is a mixture of gapped segments of categories A and B; **Target%{in A}** = the cumulative percentages of category-A gapped segments (compared to their total number) that are the closest to the target percentage (80% and 90% in this analysis), on each of the upper- and lower-side; **Target%{in B}** = the cumulative percentages of category-B gapped segments (compared to their total number) corresponding the Target%{in A}; **Target#{in A/B}** = the total number of category-A/B gapped segments corresponding to the above Target%{in A/B}.

Spacer-size upper-bound	Spacer-size threshold	Tot# {pure-B}	Tot# {pure-A}	Tot# {mixed}	Target %{in B}	Target %{in A}	Target#{in A} / Target#{in B}
10	5				74.97%	38.97%	0.3678
					81.01%	43.63%	0.3811
					89.51%	53.39%	0.4220
					90.16%	54.61%	0.4286
10	6				78.79%	40.63%	0.3649
					82.01%	43.50%	0.3753
					89.95%	54.38%	0.4278
					90.54%	55.74%	0.4356
15	7				78.48%	38.61%	0.3481
					80.49%	40.45%	0.3556
					89.78%	55.94%	0.4409
					90.09%	56.79%	0.4460
20	7				79.70%	39.57%	0.3513
					80.50%	40.34%	0.3546
					89.93%	57.72%	0.4541
					90.08%	58.19%	0.4571

Figure 10: **Performance of Method II to *artificially* cluster gapped segments, followed by method to exclude "complex" errors.** The same notes as in Figure 9 apply.

Spacer-size upper-bound	Spacer-size threshold	Tot# {pure-B}	Tot# {pure-A}	Tot# {mixed}	Target %{in B}	Target %{in A}	Target#{in A} / Target#{in B}
10	2	556980	369788	79019	73.51% 80.14% 89.44% 90.30%	36.07% 40.76% 52.61% 54.60%	0.3472 0.3599 0.4162 0.4278
10	3	553751	364853	87183	77.81% 81.15% 89.44% 90.06%	37.74% 40.78% 52.65% 54.08%	0.3432 0.3556 0.4165 0.4249
10	6	533101	333460	139226	79.84% 81.01% 89.68% 90.04%	36.92% 38.42% 56.10% 57.12%	0.3272 0.3356 0.4426 0.4489
15	5	533538	333275	138974	77.50% 80.28% 89.90% 90.26%	35.10% 37.66% 51.11% 58.16%	0.3205 0.3319 0.4495 0.4559
15	6	524762	320232	160793	79.65% 81.28% 89.82% 90.08%	36.73% 39.05% 58.17% 58.98%	0.3263 0.3399 0.4582 0.4633
20	5	526973	323550	155264	79.32% 80.34% 89.64% 90.00%	36.78% 38.02% 58.31% 59.47%	0.3280 0.3348 0.4603 0.4675

Figure 11: **Performance of Method III to *artificially* cluster gapped segments, followed by method to exclude "complex" errors.** The same notes as in Figure 9 apply.

45-47% of segments in "complex" errors. This suppressed accuracy may be because the methods here use the gap-configurations *alone*, and the accuracy is expected to improve greatly if some information on residue-configurations are incorporated *smartly*. (This will be discussed further in section 4.)

4 Discussions

Because multiple sequence alignments (MSAs) play central roles in advanced studies of homologous biological sequences (*e.g.*, [3, 4, 5, 6, 7, 8]), it is essential to reconstruct MSAs as accurately as possible. However, *even if* reconstructed by state-of-the-art aligners (*e.g.*, [3, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24]), reconstructed MSAs are *not* free from errors, *i.e.*, mis-alignments (*e.g.*, [28, 25, 29, 30, 4, 19, 26, 27, 31, 32, 1]). With this unwelcome fact as a backdrop, a number of methods have been developed that attempt to identify and correct errors, or mis-alignments, (or, more precisely, low reliability regions) in MSAs (*e.g.*, [33, 34, 35, 36, 37, 38, 39]). On the other hand, as some

studies on sequence alignments suggest (*e.g.*, [40, 41, 1]), a (near-)majority of such mis-alignments are due to the stochastic nature of sequence evolution processes, and thus it is inevitable to construct a probability distribution of alternative sequence alignments (*e.g.*, [42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 32, 53, 54, 55, 56, 2]), instead of merely reconstructing a single optimum alignment.

Whether you aim to detect and correct mis-alignments in an MSA reconstructed by an aligner of single-optimum type, or to construct a probability distribution of alternative MSAs, it should be greatly useful and beneficial to understand more about the nature of mis-alignments. As a successor to the previous studies conducted under such a philosophy [27, 1], this study developed a couple of tools to help better understand the nature of mis-alignments.

The first tool, the "substitutional residue-difference map" (SRD Map), maps SRDs, which are inferred from the residue pattern of an input MSA *via* an extension of the widely used likelihood method (*e.g.*, [57, 58, 59, 67]), onto the MSA extended to include internal branches. Applying this SRD Map, along with another new tool, the "alignment-shear map" (see appendix B), to the reconstructed MSAs of simulated sequences, we found that SRDs on mis-alignments are on average significantly larger than those off mis-alignments. But, at the same time, we also found that, even on mis-alignments, the points with SRDs near zero account for a substantially larger fraction than expected from random matching. These two results imply that attempts to detect mis-alignments using SRDs will be successful *to some extent* but *not* to the level of *absolute satisfaction*. Thus, to better detect mis-alignments, examining some other features, such as the presence of repeats near the ends of runs of gaps, are suggested to be useful.

Next, focusing on "purge"-errors [27, 1], we performed a sliding-window analysis using two different P-values defined with SRDs in each window. For this particular type of errors, SRDs seemed to be quite useful: our simulation study suggested that as much as 66-71% of "purge"-errors can be captured while keeping the false-positive rate as low as 1.8-0.8%.

And, finally, we also examined the usefulness of gap-configurations for the purpose of detecting mis-alignments. Specifically, we developed a method to detect "complex" errors using only gap-configurations. Application of the tool to reconstructed MSAs of simulated sequences indicated that the tool can exclude about 65% of gapped segments in "complex" errors while keeping 80% of other segments. This suggests that *even* gap-configuration *alone* should be *somewhat*, but *not so greatly*, useful for detecting mis-alignments, which is understandable considering that insertions/deletions are about an order of magnitude sparser than substitutions in general (*e.g.* [70]).

These results in conjunction suggest that we may be able to detect (or infer) mis-alignments much more accurately if we combine SRDs with gap-configurations, as well as with some other features (such as repeats near the ends of gaps) *in a smart manner*. But how can we know the "*smart manner*"? One way would be *through* meticulous analyses, as we did in the previous study [1] and this study. The tools we developed, such as the "position-shift map" [1], the SRD map (subsection 2.1), and the "Alignment-Shear Map" (appendix B), will help understand more about the mis-alignments. That said, meticulous analyses are time-consuming, labor-intensive, and tiring. Thus, another, more recommended, way should be to resort to **machine learning** techniques (*e.g.*, [71, 72, 73]), especially **deep learning** *via* **artificial intelligence (AI)** (*e.g.*, [74, 75]). It should be more suitable for handling the aforementioned combination of information. Besides, the category of "complex" errors actually means, "the current error classification method *cannot* unravel this error into a series (or a combination) of elementary errors", and thus this category should keep evolving as the methods to classify mis-alignments keep improving. This makes resorting to the machine learning techniques more advantageous. Still, because the human beings are the creatures of reasoning, meticulous analyses need also be continued.

Recently, we have developed a new method, the "alignment neighborhood explorer" (or "ANEX" for short), that attempts to construct probability distributions of alternative alignments in the neighborhoods of an input reconstructed MSA [2]. To the best of our knowledge,

this ANEX is the first method to apply *genuine* sequence evolution models to the problem of MSA reconstruction as a whole, including the problem of statistical MSAs. Currently, ANEX rests on the simple method to detect (and exclude) "complex'" errors (described in subsection 2.3 in this paper), as well as a simple architecture of MSA neighborhood exploration (Figure 4 of [2]). When deciding how to explore the MSA neighborhood, ANEX currently combines the information on the "purge"-like error candidates, which was derived via the sliding-window analysis using the SRD map (subsection 2.2), with the gap-configuration, to *loosely* restrict the types of elementary moves it attempts. Using this simple architecture as a "starting point," ANEX's architecture of MSA neighborhood exploration may get to evolve further into a more sophisticated yet *smart* architecture, in which a combination of three or more elementary moves may be attempted while avoiding the problem of combinatorial explosion, if aided by machine learning techniques (*e.g.*, [71, 72, 73]) including deep learning via AI (*e.g.*, [74, 75]).

Therefore, ANEX's development may be benefited from machine learning (including deep learning) in a *dual* manner, one in developing a method for more accurate detection (and exclusion) of "complex' errors, and the other in realizing more sophisticated and smart architecture of MSA neighborhood exploration. In each of the ways, the lessons learned from this study should help navigate your further endeavors. In any case, we are now about to enter a new era in which it is commonplace to construct MSA probability distributions (under *genuine* sequence evolution models) with the assistance of artificial-intelligence (AI).

4.1 Final Note

Some of our comments in this paper or other papers may sound like harsh criticisms on other researchers or their works. We strongly urge the readers to understand that such comments are our candid expressions of our sincere and pure hope for the advance of the science *in the right direction*, and that we have no intension to attack, harm, or hurt anybody

or anybody's works. It should be kept in mind that we, all *hard-working* researchers in the world, are *not* enemies to each other but actually *comrades* to each other, who are fighting against the *common enemies*, *i.e.*, insufficient understanding of the Mother Nature and the lack of tools potent enough to uncover the essence of natural phenomena, as well as being complacent of the status quo like that. We truly hope for the future where we, all researchers, go hand-in-hand with each other to improve our understanding of the Mother Nature, by bringing together ones' own strengths under the common cause *instead of* competing against each other or even sabotaging each other's studies , and by sharing all information with each other*instead of* keeping crucial information to oneself. Then, our understanding of the Mother Nature should surely improve much faster than we've ever experienced. (If, however, there are, by any chance, *corrupt* researchers who are indulging in the complacency and/or who attempt to deform the scientific truths to their own interests, we *wlll* resolutely fight against them.)

5 Acknowledgments

The author (K.E.) greatly thanks Prof. Tetsushi Yada at Kyushu Institute of Techonology, Japan for the logistic support and encouragements during the middle third of this project, which includes this study and some others [76, 77, 1, 78, 79, 2, 80], and which was conducted first in the author's home in Yokosuka, Kanagawa, Japan, second in Kyushu Institute of Technology, Japan, and last in the author's home in Chichibu, Saitama, Japan. He is also grateful to Prof. Dan Graur at University of Houston, TX, US and Dr. Giddy Landan at Christian-Albrechts-University of Kiel, Germany for letting me participate in their project, "Error Correction in Multiple Sequence Alignments", which was funded by US National Library of Medicine (grand number: LM010009-01 to Dan Graur and Giddy Landan, then

at the University of Houston), from September 2009 till June 2011; partly inspired by their project, the author came up with this project. The author appreciates the inspiring discussions with Dr. Reed A Cartwright at Arizona State University, US and with Dr. Ian Holmes at University of California, Berkeley, US. He is also grateful to Prof. Naruya Saitou at National Institute of Genetics (NIG), Japan, and Dr. Kirill Kryukov at Tokai University, Japan for helping his interest in sequence alignment methods originate and grow, while he was studying with them at NIG. Last but not least, the author appreciates all of his family members, relatives, mentors, (ex-)friends, (ex-)supervisors, and (ex-)colleagues, for their support since his infancy, which enabled him to tread (or wander?) the scientific path and to manage to "finish" this project through all those difficulties and tough times.

The project including this study was in part supported by the Ministry of Education, Culture, Sports, Science and Technology (MEXT) of Japan (grant numbers: KAKENHI Grant numbers 221S0002, 15H01358, both to Tetsushi Yada).

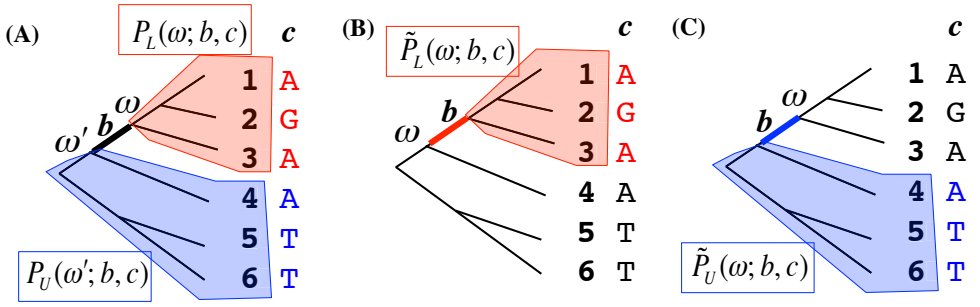

Figure 12: **Four sets of probabilities important for calculation of SRDs. A.** Probabilities, $P_L(\omega; b, c)$ and $P_U(\omega'; b, c)$, regarding the residue configurations of two *complementary* sequence sets in a column (c). The branch b (thick) separates the sequence sets. The portions of the tree yielding $P_L(\omega; b, c)$ and $P_U(\omega'; b, c)$ are shared in red and blue, respectively. And, in column c , the color of each residue indicates which of the probabilities it contributes to. **B.** The extension of $P_L(\omega; b, c)$ to the upper-end of b, to give $\tilde{P}_L(\omega; b, c)$. **C.** The extension of $P_U(\omega; b, c)$ to the lower-end of b, to give $\tilde{P}_U(\omega; b, c)$. In each panel, the numbers assigned to the external nodes also specify the sequences in the MSA.

Appendixes

A Extending Pruning Algorithm to Provide Ingredients of SRDs

In subsection 2.1, in order to compute "substitutional residue-differences" (or SRDs for short) along a branch (b) in an MSA column (c), we introduced two types of probabilities (besides the finite-time transition probabilities), that is: (ii) the conditional probability, $P_L(\omega; b, c)$, that, given residue ω ($\in \Omega$) at the lower-end of branch b, we observe the residues of all its descendant extant sequences present in column c (Figure 12 A); and (iii) the joint probability, $P_U(\omega; b, c)$, that, in column c, we observe ω ($\in \Omega$) at the upper-end of branch b *and* the residues of all extant sequences on the upper-end-side of branch b (Figure 12 A). Here, we describe a pair of algorithms to compute and output these types of probabilities.

Before going on to the algorithms, we additionally introduce the "extensions" of the aforementioned two kinds of probabilities:

(ii') the conditional probability, $\tilde{P}_L(\omega; b, c)$, hat, given residue ω ($\in \Omega$) at the upper-end

of branch b, we observe the residues of all its descendant extant sequences present in column c;

(iii') the joint probability, $\tilde{P}_U(\omega; b, c)$, that, in column c, we observe ω ($\in \Omega$) at the lower-end of branch b *and* the residues of all extant sequences on the upper-end-side of branch b.

The $\tilde{P}_L(\omega; b, c)$'s are obtained by "extend"ing the defining sub-tree of $P_L(\omega; b, c)$'s by the branch b (illustrated by Figure 12 B):

$$\tilde{P}_L(\omega; b, c) = \sum_{\omega' \in \Omega} P(\omega \mapsto \omega'; b)\ P_L(\omega'; b, c)\ . \tag{15}$$

The $\tilde{P}_U(\omega; b, c)$'s are obtained by "extending" the defining sub-tree of $P_U(\omega; b, c)$'s by b (illustrated by Figure 12 C):

$$\tilde{P}_U(\omega; b, c) = \sum_{\omega' \in \Omega} P_U(\omega'; b, c)\ P(\omega' \mapsto \omega; b)\ . \tag{16}$$

These "extended" probabilities somewhat simplify the algorithm, at least in terms of notation. Besides, they also simplify the computations of some probabilities regarding the residue-configuration of a column. (See, *e.g.*, Figure 18, or appendix J.2 in [2].) Therefore, if there is enough memory space, it may be better to keep these "extended" probabilities *even* after the end of the algorithm(s).

Now, we describe the pair of algorithms: a **"bottom-up" algorithm**, which proceeds from the leaves (*i.e.*, external nodes) to the root of the tree, and a **"top-down" algorithm**, which proceeds from the root to the leaves.

The **"bottom-up" algorithm** is a slightly modified version of Felsenstein's pruning algorithm [57, 58], and efficiently calculates the $P_L(\omega; b, c)$'s and the $\tilde{P}_L(\omega; b, c)$'s. It starts with an "initialization" step, where each external branch (denoted as b^X here) is associated with the following probabilities:

$$P_L(\omega; b^X, c) = \delta\Big(\omega, \omega(c, n^X)\Big)\ , \tag{17}$$

$$\tilde{P}_L(\omega; b^X, c) = P(\omega \mapsto \omega(c, n^X); b^X)\ . \tag{18}$$

Here, the $\omega(c, n^X)$ denotes the residue that the sequence at the external node n^X has in column c. (The n^X is the lower-end of b^X.) The $\delta(\omega, \omega')$ denotes Kronecker's delta: $\delta(\omega, \omega') = 1$ if $\omega = \omega'$, $= 0$ otherwise.

[**NOTE:** when the sequence has no residues in the column c, we assign $P_L(\omega; b^X, c) = \tilde{P}_L(\omega; b^X, c) = 1$ for all $\omega \in \Omega$.]

And remember that the $P(\omega \mapsto \omega'; b)$ is the probability that the lower-end of branch b has residue ω', conditioned on that the upper-end of b has ω.

After the initialization, as the algorithm climbs up the tree, each branch (denoted as b) is provided with the probabilities *via* the following recursion relations:

$$P_L(\omega; b, c) = \prod_{b' \in Ch(b)} \tilde{P}_L(\omega; b', c) \ , \tag{19}$$

$$\tilde{P}_L(\omega; b, c) = \sum_{\omega' \in \Omega} P(\omega \mapsto \omega'; b) \ P_L(\omega'; b, c) \ . \tag{20}$$

Here, the $Ch(b)$ denotes the set of "child" branches of branch b. Eq. 19 follows easily from what these probabilities mean (see Figure 12, panels A and B), and Eq. 20 is nothing other than the definition of $\tilde{P}_L(\omega; b, c)$ (*i.e.*, Eq. 15).

Next, the **"top-down" algorithm** calculates the $P_U(\omega; b, c)$'s and the $\tilde{P}_U(\omega; b, c)$'s efficiently. In its "initialization" step, it associates each child branch of the root ($b \in Ch(n^R)$) with the probabilities:

$$P_U(\omega; b, c) = P(\omega, n^R) \prod_{b' (\neq b) \in Ch(n^R)} \tilde{P}_L(\omega; b', c) \ , \tag{21}$$

$$\tilde{P}_U(\omega; b, c) = \sum_{\omega' \in \Omega} P_U(\omega'; b, c) \ P(\omega' \mapsto \omega; b) \ . \tag{22}$$

Here, the $P(\omega, n^R)$ is the probability (or the relative frequency) of residue ω at the root node (n^R). Eq. 21 easily follows from what the $P_U(\omega; b, c)$ and the $\tilde{P}_L(\omega; b', c)$'s mean (Figure 12, panels A and B), and Eq. 22 is nothing other than the definition of $\tilde{P}_U(\omega; b, c)$ (*i.e.*, Eq. 16).

After the initialization, as the algorithm goes down the tree, each branch (again, denoted

as b) is provided with the probabilities *via* the following recursion relations:

$$P_U(\omega; b, c) = \tilde{P}_U(\omega; p(b), c) \prod_{b'(\neq b) \in Ch(p(b))} \tilde{P}_L(\omega; b', c) \ , \tag{23}$$

$$\tilde{P}_U(\omega; b, c) = \sum_{\omega' \in \Omega} P_U(\omega'; b, c) \ P(\omega' \mapsto \omega; b) \ . \tag{24}$$

Here, the $p(b)$ denotes the "parent" branch of b, and Eq. 24 is, again, exactly the defining equation, Eq. 16.

The pair of algorithms (on a single column) has the time complexity of $O\Big(|E_{\mathcal{T}}| \times |\Omega| \times (|\Omega| + \langle |n| \rangle_{\mathcal{T}})\Big)$, where the $|E_{\mathcal{T}}|$ is the number of branches (edges) in the tree ($\mathcal{T}$), the $|\Omega|$ is the size of the alphabet (Ω), *i.e.*, the number of single-residue states, and the $\langle |n| \rangle_{\mathcal{T}}$ is the average connectivity over the nodes in the tree. Because $\langle |n| \rangle_{\mathcal{T}}$ is always less than 2, [16] the time complexity could be approximated as $O\Big(|E_{\mathcal{T}}| \times |\Omega|^2\Big)$, if desired.

[**NOTE:** The author came up with this pair of algorithms *by himself*(, of course based on the pruning algorithm [57, 58]). However, because of its simpleness and elegance, we suspect that this pair of algorithms may have already been invented (and published) by someone else. If this is indeed the case, we sincerely apologize for our failure to credit its authentic inventor(s); because we've run out of time, we could *not* search for the past related literature *exhaustively.*]

[16] Let $|n|$ be the connectivity of the node n, which means the number of edges connected at n. Because each edge always connects with two nodes, each edge is counted *exactly twice* when calculating the total connectivity over all nodes. Hence we have:

$$\sum_{n \in N[\mathcal{T}]} |n| = 2|E_{\mathcal{T}}| \ ,$$

where the $N[\mathcal{T}]$ denotes the set of all nodes in the tree ($\mathcal{T}$). Meanwhile, in a tree, we always have $|E_{\mathcal{T}}| = |N[\mathcal{T}]| - 1$, where the $|N[\mathcal{T}]|$ is the number of all nodes in $\mathcal{T}$. Therefore, we have:

$$\langle |n| \rangle_{\mathcal{T}} \stackrel{\text{def}}{=} \left\{ \sum_{n \in N[\mathcal{T}]} |n| \right\} \Big/ |N[\mathcal{T}]| = 2|E_{\mathcal{T}}| \ / \ |N[\mathcal{T}]| = 2(1 - 1/|N[\mathcal{T}]|) < 2 \ .$$

A.1 Identities among Ingredients

The ingredients calculated via the aforementioned pair of algorithms satisfy a number of equations (or identities). Here, we will provide some, for possible later convenience.

First, the probability of the residue configuration of a column (c), $P[c]$, can be calculated in many ways on branch b:

$$\begin{aligned} P[c] &= \sum_{\omega\ \in\Omega} \sum_{\omega' \in\Omega} P_U(\omega; b, c)\ P(\omega \mapsto \omega'; b)\ P_L(\omega', b, c) && (25) \\ &= \sum_{\omega\ \in\Omega} \tilde{P}_U(\omega; b, c)\ P_L(\omega, b, c) && (26) \\ &= \sum_{\omega\ \in\Omega} P_U(\omega; b, c)\ \tilde{P}_L(\omega, b, c)\ . && (27) \end{aligned}$$

At each bifurcated internal node n (other than the root n^R), the following "three-way equation" holds:

$$P[c] = \sum_{\omega\ \in\Omega} \tilde{P}_U(\omega; b_P(n), c)\ \tilde{P}_L(\omega, b_{C1}(n), c)\ \tilde{P}_L(\omega, b_{C2}(n), c)\ . \quad (28)$$

Here, the $b_P(n)$ is the "parent" branch that is immediately above n, and the $b_{C1}(n)$ and the $b_{C2}(n)$ are the "child" branches that are immediately below n. (It is easy to generalize Eq. 28 to a multi-furcated node.) Its counterpart at the root (n^R) is simply the final equation of the recursion for the pruning algorithm:

$$P[c] = \sum_{\omega\ \in\Omega} \left\{ P[(\omega; n^R) \prod_{b\ \in Ch(n^R)} \tilde{P}_L(\omega, b, c) \right\}\ . \quad (29)$$

Here, the $P[(\omega; n^R)$ is the probability (or the frequency) of residue ω ($\in \Omega$) at the root (n^R), and the $Ch(n^R)$ denotes the set of all child branches of n^R.

These equations provided in this subsection, or their further extensions, may be exploited when you need to upgrade the ANEX [2] in the future, so that it can compute the substitution components of the probabilities of alternative MSAs much more quickly, particularly when the number of aligned sequences is large.

B Creating "Alignment-Shear Map" from "Position-Shift Map"

As briefly explained in footnote 2, we have created a new subroutine, "list_minicls_shift_respos_clm_based" (in the module, "MyPosShiftMap.pm," of ANEX(_P)), which is the core of a couple of Perl scripts that automatically map the **"alignment-shear"s** onto the MSA (*vertically* extended to include internal branches), given an input **"position-shift-map."** (The **"extended MSA"** is regarded as the direct product of the set of sites (or column-positions) times the set of branches.) Here, each **"alignment-shear"** is a horizontal line in the "extended MSA" that vertically (or *phylogenetically*) separates regions with different "position-shift"s.

In this appendix, we describe the main algorithm underlying the subroutine, in a somewhat verbal manner (not using pseudo-code(s)).

The main inputs are: (1) a "position-shift-map," which is based on a reconstructed MSA, and in which the residues are replaced with the corresponding position-shifts; and (2) a phylogenetic tree of the aligned sequences.

1. Horizontally chop the "position-shift-map" into "position-shift-pattern-block"s (or "PSP-blocks" for short), each of which consists of contiguous columns sharing the same pattern of position-shifts.

2. For each position-shift-pattern that was found, enumerate parsimonious clusterings of position shifts, as follows:

   (a) Determine *uniquely* the ancestral "presence"/"absence" (or residue/gap) states according to the Dollo parsimonious principle [60];

   (b) Enumerate *all* the parsimonious sets of "position-shift-states" assigned *only* to the ancestral nodes with "presence" states, so that they are consistent with the *fixed* "position-shift-states" of the aligned sequences; this is done according to Sankoff's

parsimony algorithm [81], using the cost matrix with the diagonal elements = 0, and all others = 1;

(c) For each parsimonious set thus obtained, the set of branches along which the "position-shift"s change dictates a (parsimonious) clustering of position-shifts.

3. Select the "optimum" combination(s) of parsimonious clusterings, each of which is assigned to each PSP-block, as follows:

   (a) For the PSP-blocks each of which has only one way of parsimonious clustering, choose that *unique* clustering;

   (b) For those PSP-blocks flanked by (a) block(s) with *unique* clustering(s), choose the clustering(s) with the largest number of changes consistent with the flanking unique clusterings;

   (c) Repeat (b) until the clusterings cannot be narrowed down any longer;

   (d) If some remaining PSP-blocks are contiguous to each other, examine, from left to right, the consistency of the changes, and chose most consistent pairs of the clusterings; do this selection until such contiguous remaining blocks are exhausted;

   (e) If there are still more than one combinations of the clusterings, further select the combination(s) according to the following criteria (in this order);

      i. those with the largest sum, over all PSP-blocks, of the *total depths* of the branches along which changes occur;

      ii. those with the largest sum, over all PSP-blocks, of the *minimum depths* of the branches along which changes occur;

   (f) If there are still more than one combination of the clusterings, keep *all of them* as equally optimum choices.

4. For each "optimum" combination of parsimonious clusterings assigned to the PSP-blocks, there should be mini-blocks of position-shifts; each mini-block horizontally

spans the PSP-block it belongs to, and is delimited by one branch or more along which the position-shift changes;

5. For each "optimum" combination of parsimonious clusterings assigned to the PSP-blocks, merge the *effectively* adjoining mini-blocks [17] that share the identical position-shift and the identical set of affected sequences; the resulting merged blocks are referred to as "MINI-classes" of position-shifts.

Each "MINI-class" thus created should be assigned a *unique* set of delimiting branches, which in turn should provide the **"alignment-shear"s** to be mapped onto the extended MSA, *via* a downstream subroutine ("mk_map_alnshear_clmbased" in the module, "MyMapAlnShear.pm").

[**NOTE:** In a previous study [1], we created a similar program, "classify_msa_errors_via_mblks.alpha2.pl" in the "ComplLiMment(_P)" package, albeit for a somewhat different purpose of creating "position-shift-blocks." This old program, however, employed an algorithm that *substantially* differs from the new algorithm described here. **The biggest difference is in the strategy.** The old algorithm first *horizontally* chops each sequence into segments, each of which consists of contiguous sites with the same position-shift; then, it attempts to *vertically* (or *phylogenetically*) cluster (or merge) the segments sharing both ends and having the same position-shift. In contrast, the new algorithm first *vertically* (or *phylogenetically*) cluster the sequences with the same position-shifts in each MSA column, forming "position-shift cluster"s; then, it attempts to *horizontally* merge the neighboring "position-shift" clusters with the same position-shift.

Using dozens of typical erroneous segments (created in [1]), we compared, by manual inspection, the "alignment-shear"s predicted by these two algorithms. We found that the

[17] The "*effectively* adjoining" means either actually adjoining or separated solely by columns in which all the affected sequences are occupied by gaps.

"alignment-shear"s predicted by the new algorithm reflects the true boundaries (between different "position-shift"s) *better than* those predicted by the old algorithm.

Thus, if you incorporate the results of this new algorithm into the construction of "position-shift-block"s, as well as the subsequent classification of alignment errors (as attempted in [1]), the results may be improved considerably.]

C Simple Sliding Window Analysis Taking Account of Rate Variation

The method for the sliding window analysis described in subsection 2.2 does not take account of rate variation across sites (like, *e.g.*, [64, 65, 66, 59, 67]), due to some biological factors such as selection and mutation hot-spots. To incorporate such factors, a useful way would be to take advantage of the "observations" *along* different branches (yet in the same window). There would be a variety of ways to implement this idea. Here, we propose a simple method of ***re-scaling*** the "expected" SRD (in window W and along branch b), $D_{exp}(b, W)$ (given by Eq. 11), by the ratio of the total ?observed? number across the branches (except b) to the total ?expected? number. Then, the "rescaled" "expected" SRD, denoted as $D^{*}_{exp}(b, W)$, is calculated as:

$$D^{*}_{exp}(b, W) = R_D(b, W)\ D_{exp}(b, W)\ , \tag{30}$$

$$\tag{31}$$

with

$$R_D(b, W) \stackrel{\text{def}}{=} \left\{ \sum_{b'(\neq b)\ \in E_{\mathcal{T}}} D_{obs}(b', W) \right\} \Big/ \left\{ \sum_{b'(\neq b)\ \in E_{\mathcal{T}}} D_{exp}(b', W) \right\}\ . \tag{32}$$

Here, the $E_{\mathcal{T}}$ denotes the set of all branches (*i.e.*, edges) in the tree ($\mathcal{T}$). Then, we can compare the "observation", $D_{obs}(b, W)$, to this "rescaled" "expectation", $D^{*}_{exp}(b, W)$. For

example, when performing a binomial test, we can use the formula, Eq. 12, with the probability of "difference" per trial, p_D, replaced by its "rescaled" version: $p_D^* \stackrel{\text{def}}{=} D_{exp}^*(b, W)/N_W$.

D *Artificially* Clustering Gapped Segments: Definitions of Methods

In order to artificially cluster the gapped segments, we tried three different methods.

Method I: Two nearest-neighboring gapped segments are clustered if the spacer between them is shorter than **{spacer-size threshold}** (Figure 4, panel B).

Method III: Two (not necessarily nearest-neighboring) gapped segments, as well as all segments in between them, are artificially clustered if the following conditions are satisfied:

(a) EITHER [the (composite-)spacer between them is $\leq$ **{spacer-size threshold}**], OR [the (composite-)spacer is $\leq$ **{spacer-size upper-bound}**, AND {size of smaller gapped segment} $\geq$ {size of (composite-)spacer} /2, AND {size of larger gapped segment} $\geq$ 2 {size of (composite-)spacer}];

(b [the two gapped segments undergo at least one indel each along the same branch] OR [the two gapped segments undergo (effective-)indels along branches that are mutually phylogenetically nearest-neighboring] (Figure 4, panel C).

Method II: Similar to Method III, but the latter half of condition (b) is replaced by the following: [the two gapped segments, and yet another neighboring gapped segment undergo (effective-)indels along three branches connected at an internal node] (Figure 4, panel D).

In verbal description, **Method I** does not care about phylogenetic positioning of gap-blocks. **Method III** examines the phylogenetic positioning of gap-blocks in neighboring gapped segments. More precisely, it examines whether or not the neighboring gapped segments undergo (effective-)indels along the same branch or along phylogenetically neighboring

branches. **Method II** also examines the phylogenetic positioning of gap-blocks in neighboring gapped segments, in a more complex manner. More precisely, it examines whether or not there is a "trio" of branches that are connected at an internal node and each of which undergoes (effective-)indels in each of three neighboring gapped segments.

E Attempting to Detect as Many "Complex" Errors as Possible out of *Artificial* Clusters

After artificially clustering the gapped segments with each of the three methods described in appendix D, we attempted to detect "complex" errors *as accurately as possible*, by conducting the following analysis.

(**NOTE:** The goal of this analysis is to *exclude* as much "complex" errors as possible, while keeping a specified fraction (80% or 90%) of control gapped segments.)

First, the artificial clusters were classified according to the number of insertions and the number of deletions. Second, in each of the resulting classes, we counted the total number of gapped segments involved in "complex" errors and the total number of control gapped segments. Third, the classes were sorted in ascending order of the ratio:

#{gapped segments in "complex" errors} / #{control gapped segments}.

Finally, from the class with the smallest ratio, the classes were chosen until the number of control gapped segments reaches a specified fraction (80% or 90%) of the total number of the control gapped segments.

We tried various combinations of the two parameters, namely, the **{spacer-size threshold}** and the **{spacer-size upper-bound}**, both used in appendix D.

Before this analysis, each "complex" error was re-classified as "non-complex" if each of the reconstructed and reference MSAs in it is inferred to have resulted from only less than 4 indels. This is in anticipation of an error-classification method better than that used in [1].

References

[1] K Ezawa. Characterization of multiple sequence alignment errors using complete-likelihood score and position-shift map. *BMC Bioinformatics.*, 17:133, 2016.

[2] K Ezawa. Alingment Neighborhood EXplorer (ANEX): First attempt to apply *genuine* sequence evolution model with realistic insertions/deletions to Multiple Sequence Alignment reconstruction problem. preprint (KEZW_BI_ME00006.anex.pdf) available at: https://www.bioinformatics.org/ftp/pub/anex/Documents/Preprints/., 2020.

[3] D Gusfield. *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*. Cambridge University Press, Cambridge, UK, 1997.

[4] S Kumar and A Filipski. Multiple sequence alignment: in pursuit of homologous DNA positions. *Genome Res.*, 17:127–135, 2007.

[5] MR Aniba, O Poch, and JD Thompson. Issues in bioinformatics benchmarking: the case study of multiple sequence alignment. *Nucleic Acids Res.*, 38:7353–7363, 2010.

[6] A Löytynoja. Alignment methods: strategies, challenges, benchmarking, and comparative overview. In M Anisimova, editor, *Evolutionary Genomics. Methods in Molecular Biology (Methods and Protocols), vol. 855*, pages 203–235. Humana Press, Totowa, NJ, 2012.

[7] S Iantorno, K Gori, N Goldman, M Gil, and C Dessimoz. Who watches the watchmen? an appraisal of benchmarks for multiple sequence alignment. In D Russell, editor, *Multiple Sequence Alignment Methods. Methods in Molecular Biology (Methods and Protocols), vol. 1079*, pages 59–73. Humana Press, Totowa, NJ, 2014.

[8] T Warnow. *Computational Phylogenetics: An introduction to designing methods for phylogeny estimation*, chapter 9. Cambridge University Press, 2017.

[9] JD Thompson, F Plewniak, and O Poch. A comprehensive comparison of multiple sequence alignment programs. *Nucleic Acids Res.*, 27:2682–2690, 1999.

[10] C Notredame. Recent evolutions of multiple sequence alignment algorithms. *PLoS Comput Biol.*, 3:e123, 2007.

[11] JD Thompson, DG Higgins, and TJ Gibson. CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Res.*, 22:4673–4680, 1994.

[12] C Notredame, DG Higgins, and J Heringa. T-coffee: a novel method for fast and accurate multiple sequence alignment. *J Mol Biol.*, 302:205–217, 2000.

[13] K Katoh, K Misawa, K Kuma, and T Miyata. MAFFT: a novel method for rapid multiple sequence alignment based on fast fourier transform. *Nucleic Acids Res.*, 30:3059–3066, 2002.

[14] K Katoh, K Kuma, H Toh, and T Miyata. MAFFT version 5: implevement in accuracy of multiple sequence alignment. *Nucleic Acids Res.*, 33:511–518, 2005.

[15] K Katoh and H Toh. Recent developments in the MAFFT multiple sequence alignment program. *Brief Bioinform.*, 9:286–298, 2008.

[16] RC Edgar. MUSCLE: multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Res.*, 32:1792–1797, 2004.

[17] CB Do, MS Mahabhashyam, M Brudno, and S Batzoglou. ProbCons: probabilistic consistency-based multiple sequence alignment. *Genome Res.*, 15:330–340, 2005.

[18] A Löytynoja and N Goldman. An algorithm for progressive multiple alignment of sequences with insertions. *Proc Natl Acad Sci USA.*, 102:10557–10562, 2005.

[19] A Löytynoja and N Goldman. Phylogeny-aware gap placement prevents errors in sequence alignment and evolutionary analysis. *Science.*, 320:1632–1635, 2008.

[20] J Pei and NV Grishin. MUMMALS: multiple sequence alignment improved by using hidden Markov models with local structural information. *Nucleic Acids Res.*, 34:364–374, 2006.

[21] U Roshan and DR Livesay. Probalign: multiple sequence alignment using partition function posterior probabilities. *Bioinformatics.*, 22:2715–2721, 2006.

[22] LM Wallace, O O'Sullivan, DG Higgins, and C Notredame. M-Coffee: combining multiple sequence alignment methods with T-Coffee. *Nucleic Acids Res.*, 34:1692–1699, 2006.

[23] AR Subramanian, M Kaufumann, and B Morgenstern. DIALIGN-TX: greedy and progressive approaches for segment-based multiple sequence alignment. *Algorithms Mol Biol.*, 3:6, 2008.

[24] K Kryukov and N Saitou. MISHIMA–a new method for high speed multiple alignment of nucleotide sequences of bacterial genome scale data. *BMC Bioinformatics.*, 11:142, 2010.

[25] EA O'Brien and DG Higgins. Empirical estimation of the reliability of ribosomal rna alignments. *Bioinformatics.*, 14:830–838, 1998.

[26] KM Wong, MA Suchard, and JP Huelsenbeck. Alignment uncertainty and genome analysis. *Science.*, 319:473–476, 2008.

[27] G Landan and D Graur. Characterization of pairwise and multiple sequence alignment errors. *Gene.*, 441:141–147, 2009.

[28] DA Morrison and JT Ellis. Effects of nucleotide sequence alignment on phylogeny estimation: a case study of 18S rDNAs of apicomplexa. *Mol Biol Evol.*, 14:428–441, 1997.

[29] RE Hickson, C Simon, and SW Perry. The performance of several multiple sequence alignment programs in relation to secondary-structure features for an rrna sequence. *Mol Biol Evol.*, 17:530–539, 2000.

[30] TH Ogden and MS Rosenberg. Multiple sequence alignment accuracy and phylogenetic inference. *Syst Biol.*, 55:314–318, 2006.

[31] P Markova-Raina and D Petrov. High sensitivity to aligner and high rate of false positives in the estimates of positive selection in the 12 Dorosophila genomes. *Genome Res.*, 21:863–874, 2011.

[32] O Westesson, G Lunter, B Paten, and I Holmes. Accurate reconstruction of insertion-deletion histories by statistical phylogenetics. *PLoS One.*, 7:e34572, 2012.

[33] T Lassmann and ELL Sonnhammer. Automatic assessment of alignment quality. *Nucleic Acids Res.*, 33:7120–7128, 2005.

[34] G Landan and D Graur. Heads or tails: a simple reliability check for multiple sequence alignments. *Mol Biol Evol.*, 24:1380–1383, 2007.

[35] G Landan and D Graur. Local reliability measures from sets of co-optimum multiple sequence alignments. *Pac Symp Biocomput.*, 13:15–24, 2008.

[36] O Penn, E Privman, G Landan, D Graur, and T Pupko. An alignment confidence score capturing robustness to guide tree uncertanty. *Mol Biol Evol.*, 27:1759–1767, 2010.

[37] J Kim and J Ma. PSAR: measuring multiple sequence alignment reliability by probabilistic sampling. *Nucl Acids Res.*, 39:6359–6368, 2011.

[38] JM Chang, PD Tommaso, and C Notredame. TCS: a new multiple sequence alignment reliability measure to estimate alignment accuracy and improved phylogenetic tree reconstruction. *Mol Biol Evol.*, 31:1625–1637, 2014.

[39] I Sela, H Ashkenazy, K Katoh, and T Pupko. GUIDANCE2: accurate detection of unreliable alignment regions accounting for the uncertainty of multiple parameters. *Nucl Acids Res.*, 43:W7–W14, 2015.

[40] G Lunter, A Rocco, N Mimouni, A Heger, A Caldeira, and J Hein. Uncertainty in homology inferences: assessing and improving genomic sequence alignment. *Genome Res.*, 18:298–309, 2008.

[41] RA Cartwright. Problems and solutions for estimating indel rates and length distributions. *Mol Biol Evol.*, 26:473–480, 2009.

[42] MJ Bishop and EA Thompson. Maximum likelihood alignment of DNA sequences. *J Mol Biol.*, 190:159–165, 1986.

[43] JL Thorne, H Kishino, and J Felsenstein. An evolutionary model for maximum likelihood alignment of DNA sequences. *J Mol Biol.*, 33:114–124, 1991.

[44] JL Thorne, H Kishino, and J Felsenstein. Inching toward reality: An improved likelihood model of sequence evolution. *J Mol Biol.*, 34:3–16, 1992.

[45] J Hein, C Wiuf, B Knudsen, MB Møller, and G Wibling. Statistical alignment: computational properties, homology testing and goodness-of-fit. *J Mol Biol.*, 302:265–279, 2000.

[46] B Knudsen and MM Miyamoto. Sequence alignments and pair hidden Markov models using evolutionary history. *J Mol Biol.*, 333:453–460, 2003.

[47] I Miklós and Z Toroczkai. An improved model for statistical alignment. *WABI 2001.*, LNCS 2149:1–10, 2001.

[48] I Holmes. Using guide trees to construct multiple-sequence evolutionary HMMs. *Bioinformatics.*, 19(Suppl 1):i147–i157, 2003.

[49] MA Suchard and BD Redelings. BAli-Phy: simultaneous Bayesian inference of alignment and phylogeny. *Bioinformatics.*, 22:2047–2048, 2006.

[50] Á Novák, I Miklós, R Lyngsø, and J Hein. StatAlign: an extendable software package for joint Bayesian estimation of alignments and evolutionary trees. *Bioinformatics.*, 24:2403–2404, 2008.

[51] B Paten, J Herrero, S Fitzgerald, K Beal, P Flicek, I Holmes, and E Birney. Genome-wide nucleotide-level mammalian ancestor reconstruction. *Genome Res.*, 18:1829–1843, 2008.

[52] RK Bradley, A Roberts, M Smoot, S Juvekar, J Do, C Dewey, I Holmes, and I Pachter. Fast statistical alignment. *PLoS Comput Biol.*, 5:e1000392, 2009.

[53] JL Herman, Á Novák, R Lyngsø, A Szabó, I Miklós, and J Hein. Efficient representation of uncertainty in multiple sequence alignments using directed acyclic graphs. *BMC Bioinformatics.*, 16:108, 2015.

[54] E Levy Karin, H Ashkenazy, J Hein, and T Pupko. A simulation-based approach to statistical alignment. *Syst Biol.*, 68:252–266, 2019.

[55] N De Maio. The cumulative indel model: fast and accurate statistical evolutionary alignment. *Syst Biol.*, 2020. (available as E-pub).

[56] I Holmes. Application of indel evolution by differential calculus of finite state automata. available in bioRxiv with doi: 10.1101/2020.06.29.178764., 2020.

[57] J Felsenstein. Evolutionary trees from DNA sequences: a maximum likelihood approach. *J Mol Evol.*, 17:368–376, 1981.

[58] J. Felsenstein. *Inferring Phylogenetics*. Sinauer, Sunderland, Massachusetts, 2004.

[59] Z. Yang. *Computational Molecular Evolution*. Oxford Univ. Press, Oxford, UK, 2006.

[60] JS Farris. Phylogenetic analysis under dollo's law. *Syst Zool.*, 26:77–88, 1977.

[61] L Chindelevitch, Z Li, E Blais, and M Blanchette. On the inference of parsimonious evolutionary scenarios. *J Bioinform Comput Biol.*, 4:721–744, 2006.

[62] AB Diallo, V Makarenkov, and M Blanchette. Exact and heuristic algorithms for the indel maximum likelihood problem. *J Comput Biol.*, 14:446–461, 2007.

[63] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. *Numecial Recipes in C: The Art of Scientific Computing (2nd Ed.)*. Cambridge Univ. Press, Cambridge, UK, 1992.

[64] Z Yang. A space-time process model for the evolution of DNA sequences. *Genetics.*, 139:993–1005, 1995.

[65] X Gu and WH Li. A general additive distance with time-reversibility and rate variation among nucleotide sites. *Proc Natl Acad Sci. USA.*, 93:4671–4676, 1996.

[66] Z Yang and S Kumar. Approximate methods for estimating the pattern of nucleotide substitution and the variation of substitution rates among sites. *Mol Biol Evol.*, 13:650–659, 1996.

[67] Z Yang. PAML: a program package for phylogenetic analysis by maximum likelihood. *Mol Biol Evol.*, 24:1586–1591, 2007.

[68] S Karlin and SF Altschul. Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes. *Proc Natl Acad Sci USA.*, 87:2264–2268, 1990.

[69] TH Jukes and CR Cantor. Evolution of protein molecules. In HN Munro, editor, *Mammalian protein metabolism*, pages 21–123. Academic Press, New York, US, 1969.

[70] G Lunter. Probabilistic whole-genome alignments reveal high indel rates in the human and mouse genomes. *Bioinformatics.*, 23:i289–i296, 2007.

[71] Christopher M Bishop. *Pattern recognition and machine learning*. Springer, New York, NY, 2006. ISBN: 9780387310732, 9781493938438.

[72] Ameet Talwalkar Mehryar Mohri, Afshin Rostamizadeh. *Foundations of Machine Learning, 2nd edition*. MIT Press, Cambridge, MA, 2018. ISBN: 0262351366, 9780262351362.

[73] Ethern Alpaydin. *Introduction to Machine Learning, 4th edition (Adaptive Computation and Machine Learning series)*. MIT Press, Cambridge, MA, 2020. ISBN: 0262043793, 9780262043793.

[74] Y Lecun, Y Bengio, and G Hinton. Deep Learning. *Nature.*, 521:436–444, 2015.

[75] Aaron Courville Ian Goodfellow, Yoshua Bengio. *Deep Learning (Adaptive computation and machine learning series)*. MIT Press, Cambridge, MA, 2016. ISBN: 0262337371, 9780262337373.

[76] K Ezawa. General continuous-time Markov model of sequence evolution via insertions/deletions: are alignment probabilities factorable? *BMC Bioinformatics.*, 17:304, 2016. Erratum in: BMC Bioinformatics (2016) 17:457.

[77] K Ezawa. General continuous-time Markov model of sequence evolution via insertions/deletions: local alignment probability computation. *BMC Bioinformatics.*, 17:397, 2016.

[78] K Ezawa, D Graur, and G Landan. Perturbative formulation of general continuous-time Markov model of sequence evolution via insertions/deletions, Part IV: incorporation of

substitutions and other mutations. available in bioRxiv with doi: 10.1101/023622., 2015.

[79] K Ezawa. New perturbation method to compute probabilities of mutually adjoining insertion-type and deletion-type gaps in ancestor-descendant pairwise sequence alignment under *genuine* sequence evolution model with *realistic* insertions/deletions: the "last piece of the puzzle.". preprint (KEZW_BI_ME00005.lastpiece.pdf) available at: https://www.bioinformatics.org/ftp/pub/anex/Documents/Preprints/., 2020.

[80] K Ezawa. (Approximate) Solutions to some technical issues on alignment probability calculation under *genuine* sequence evolution model with realistic insertions/deletions. (in preparation.).

[81] D Sankoff. Minimal mutation trees of sequences. *SIAM J of Applied Math.*, 28:35–42, 1975.