Accurate extension of multiple sequence alignments using a phylogeny-aware graph algorithm: Supplementary material

Ari Löytynoja [1,2]*, Albert J. Vilella [1] and Nick Goldman [1]

[1]EMBL-European Bioinformatics Institute, Hinxton, UK

[2]Institute of Biotechnology, University of Helsinki, Finland

Contents

*to whom correspondence should be addressed: ari.loytynoja@helsinki.fi

1 Graph representation of sequences

The conversion of a regular sequence to a graph is trivial and only requires connecting each character, represented by a vertex, with edges to its preceding and succeeding characters, and adding start and end vertices as the unique end points of the graph (Supplementary Figure 1a). Partial-order graphs provide more than a complex representation of a sequence of characters, however, and allow e.g. for modelling of evolutionary units of more than one character, non-linear dependencies within a sequence and description of uncertainties in the input data (Supplementary Figures 1b–d and 2).

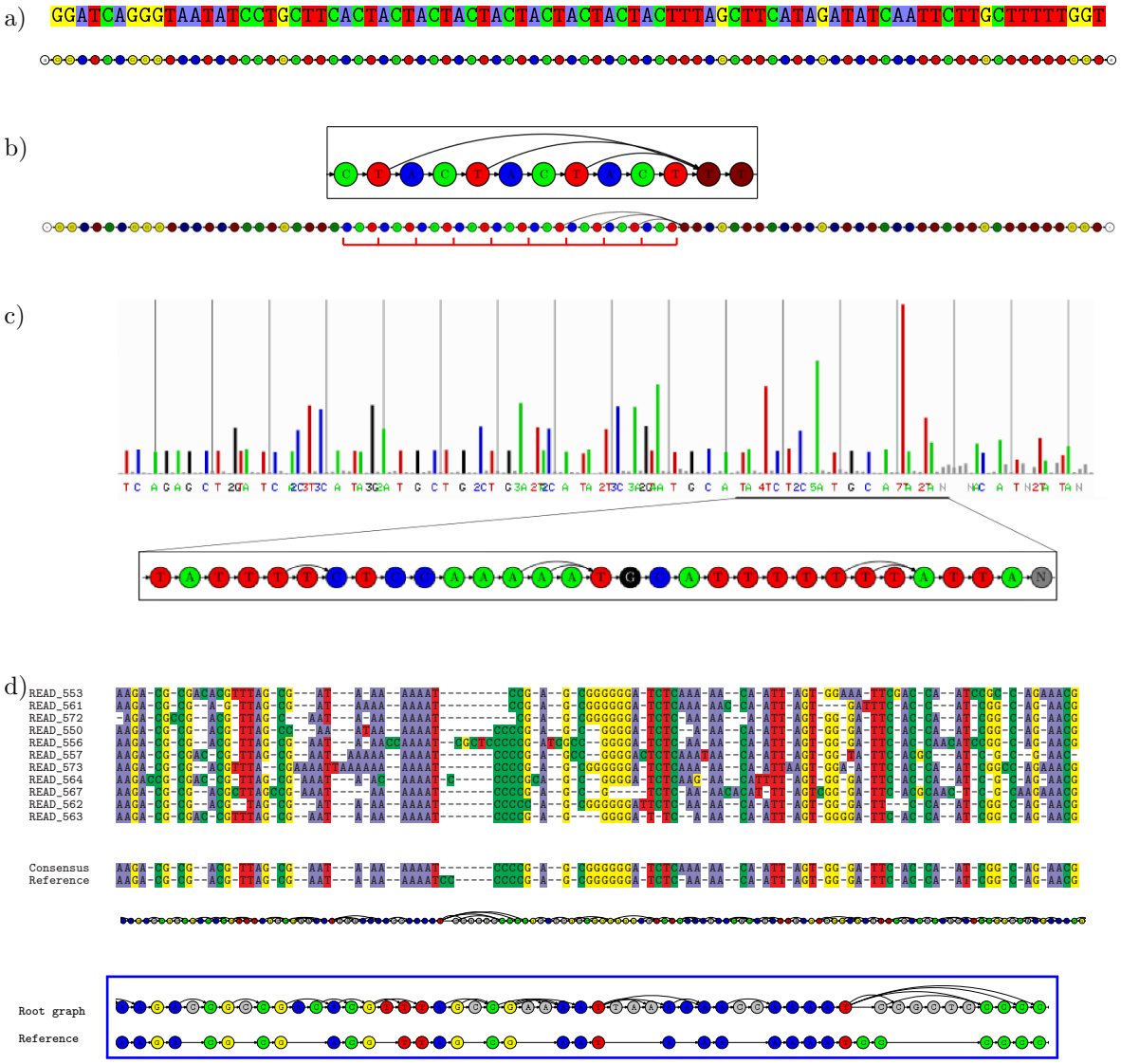
The rise of new sequencing methods has changed the balance between throughput and accuracy and the large amounts of sequence data nowadays come with different characteristics of sequencing errors depending on the sequencing platform used. Incorrect base calls affect all sequencing platforms and widely-used data analysis methods take into account the associated quality scores to improve their results [e.g. 13, 12]. PAGAN supports the input of FASTQ-formatted data and can trim low-quality read ends and mask individual low-quality bases as well as couple paired-end reads into one input graph before their alignment. Some sequencing technologies such as the Roche 454 [17], Pacific Biosciences SMRT [6], Ion Torrent (http://www.iontorrent.com) and Oxford Nanopore (http://nanoporetech.com) are also prone to insertion and deletion errors [8]. PAGAN sequence graphs are exceptionally well suited for describing such uncertainty in character presence at certain positions (Supplementary Figure 1c–d).

2 Alignment of sequence graphs

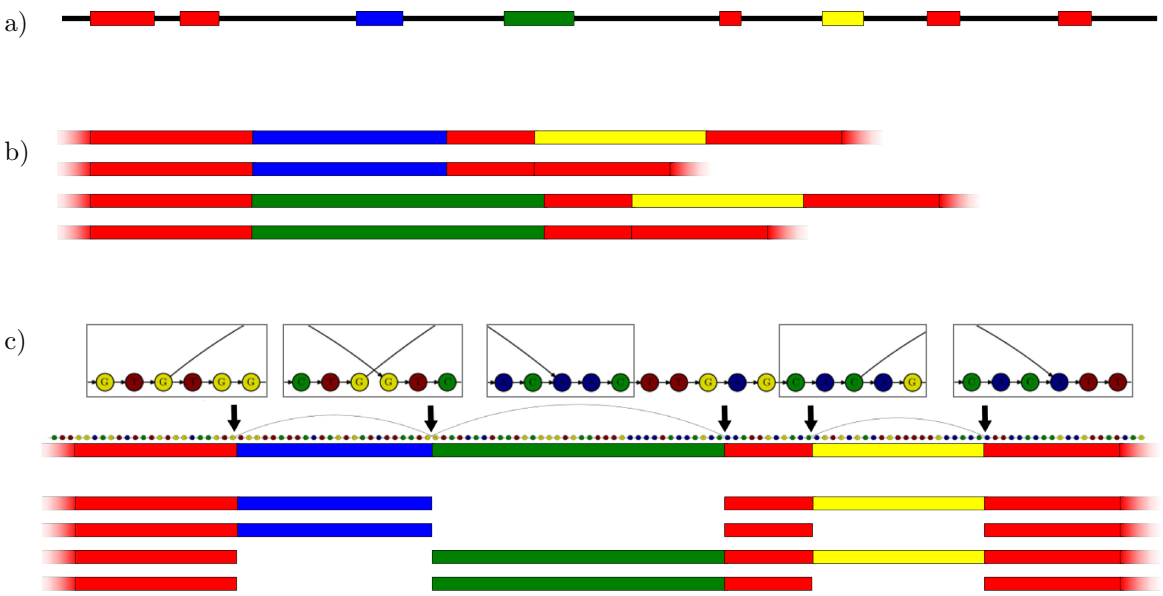
Progressive alignment algorithms attempt to backtrack the tree-like hierarchical structure of relatedness among a set of homologous sequences [16]. Each alignment clusters two sister nodes, representing either single sequences or previous alignments, and defines a new node to represent this pairwise solution, i.e. the inferred ancestor of two descendants. We extend the dynamic-programming algorithm used for the pairwise alignment of sequences [19] to align partial-order graphs and we then apply this in a progressive manner to align multiple sequences. The placement of new sequences into an existing reference alignment is not significantly different from the *de novo* alignment of sequences: we even use a standard progressive algorithm to account for the relatedness of the sequences when placing multiple sequences to one target node. The basic concepts for this are illustrated in Supplementary Figure 3 and the following paragraphs.

In progressive alignment, the handling of deletions is straightforward but insertions require a new gap to be created at each subsequent stage of the alignment process [14]. The challenge is that insertions cannot be distinguished from deletions at the time of aligning two sequences–both appear simply as a length difference between the two sequences–but failing to account for their different properties is likely to cause alignment error [15]. For example, consider two data sets of three sequences, shown in Supplementary Figure 3a and 3b as three graphs associated by a known phylogeny. The length difference between graphs A and B indicates that either an insertion or a deletion has happened (we assume that only one event has taken place) but their pairwise alignment, shown in Supplementary Figure 3c, cannot distinguish the two scenarios.

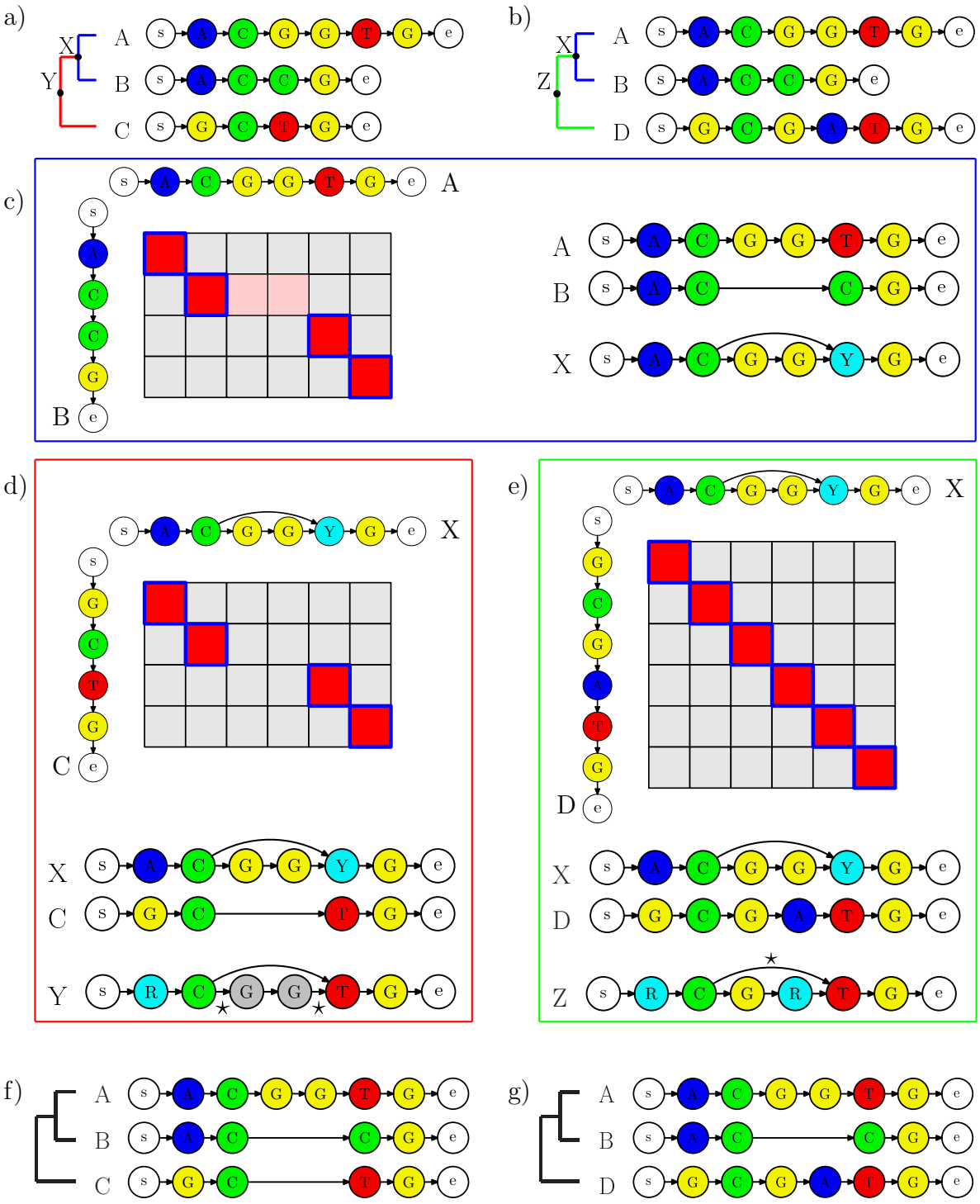
A PAGAN graph can describe this uncertainty in the type of mutation event with edges that connect vertices, representing characters in a sequence, to multiple preceding vertices; each edge is a hypothesis of the true structure of the ancestral sequence. In Supplementary Figure 3c, vertex 5 of the parent graph X has two incoming edges, from vertices 2 and 4, to indicate that the presence of vertices 3 and 4 is uncertain. The path through the graph that correctly represents the ancestral sequence is left undecided and the subsequent alignment is allowed to consider all options and choose the one that best explains the true event. In the first case of our simple example (Supplementary Figure 3a), the length difference was caused by an insertion and the subsequent alignment of X and C uses the edge that skips over the inserted sites, from vertex 2 to vertex



Supplementary Figure 1: (a) A regular sequence can be converted to a graph by connecting each character, represented by a vertex, with an edge to its preceding and succeeding character, and adding start and end vertices as the unique end points of the graph. (b) The evolution of repetitive sequences, such as microsatellites consisting of tandemly-repeated identical units, differs from that of complex sequences: repetitive sequences have very high rates of insertions and deletions and tend to grow or shrink in steps of the unit length, violating the assumption that sequence sites evolve independently and under an identical process. A PAGAN graph representing a linear sequence (a) can be extended to model the stepwise evolution of repeat length with additional edges that skip over one or more of the repeat units (b); in a pairwise alignment, the longer of the two repeat regions accounts for the length difference. (c) Graphs can describe the errors made by the Roche 454 platform on the lengths of homopolymer runs, limiting the sequence uncertainty to the base repeat and enabling correct alignment of adjacent regions. (d) Graphs can also contain the variation observed between multiple reads from the Pacific Biosciences SMRT platform: this allows for an accurate pileup of SMRT reads (top), reconstruction of high-quality consensus sequences (middle) and alignment of consensus graphs against other sequences (bottom).



Supplementary Figure 2: (a) In addition to constitutive exons (red boxes), a gene may contain alternatively-spliced exons (blue, green and yellow). (b) With differential splicing of mutually exclusive exons (blue and green in this example) and cassette exons (yellow), one gene can produce multiple different transcripts. In the alignment of mRNA and cDNA sequences, the correct matching of shorter transcripts may require creation of long gaps to account for the missing exons. Although alignment errors may be reduced by adjusting the parameters of the method, this does not change the fact that alternatively-spliced exons should neither be considered as insertions or deletions nor penalised as such. (c) PAGAN graphs are ideally suited for modelling such non-linear dependencies and a single graph can represent the exon combinations of all splicing variants, incorporating equally well mutually exclusive exons and exon skipping. When a known gene structure is represented with such a graph (top), PAGAN can align transcripts from different isoforms to the reference without incorrect penalisation of missing exons, allowing the target sequence to find the correct splicing isoform across the exon boundaries (bottom).



Supplementary Figure 3: (a, b) Progressive alignment of three sequence graphs consists of two pairwise alignments, shown in blue and red/green. (c) The first alignment is identical for both data sets, creating graph X to represent the inferred ancestor of A and B. In X, the character state of vertex 5 is `Y`, representing both pyrimidines (C and T), and has two incoming edges, from vertices 2 and 4, to indicate that the presence of vertices 3 and 4 is uncertain. (d) The optimal alignment path for graphs X and C, highlighted in blue in the dynamic programming matrix, jumps from vertex 2 in graph X to vertex 5 using the direct connecting edge; the edges flanking the skipped-over fragment are recorded as unused (asterisks). (e) In contrast, the alignment of graphs X and D matches all vertices and the edge skipping over vertices 3 and 4 is now unused. (f, g) Starting from graphs Y and Z at the root of the guide phylogeny and using the alignments at internal nodes, one can reconstruct the inferred homology in the input graphs, i.e. the multiple alignment. Although the alignment of A and B is identical in both data sets, the multiple alignments indicate that in the first data set A has a two-character insertion whereas in the second data set B has a two-character deletion.

5 in graph X (Supplementary Figure 3d). In the second case (Supplementary Figure 3b), the deletion in B only affects the first alignment and the second alignment matches all the positions, ignoring the additional edge (Supplementary Figure 3e).

When the algorithm reaches the root of the guide phylogeny, the pairwise alignments at the internal nodes of the tree structure allow backtracking the progressive process and reconstructing the inferred homology among the vertices of the input graphs. In our simple example, the resulting multiple alignments reveal that the first case is indeed better explained by an insertion (Supplementary Figure 3f) whereas in the second case a deletion has taken place (Supplementary Figure 3g).

2.1 Phylogenetic progressive alignment using sequence graphs

When a phylogenetic alignment is inferred using a progressive algorithm, the distinction between and correct representation of insertions and deletions become crucial. Sites inserted in a sequence at some point in its evolution are not homologous to any sites in the ancestor sequences before the insertion event; nor are independent insertions in other evolutionary lineages homologous to them. In a multiple alignment independent insertions should be placed in columns of their own and, to achieve this with a progressive algorithm, the matching of insertion sites should be prevented at the later stages of the alignment process despite their possible similarities to some non-homologous sites [14]. With sequence graphs, this could be done simply by removing all the edges not used in the previous alignment (those marked with asterisks in Supplementary Figure 3d) and thus preventing the vertices for the apparent insertion being accessed. This would closely resemble the greedy approach of PRANK [14], making the alignment similarly sensitive to errors in the guide phylogeny and failing to properly account for overlapping insertion and deletion events.

Instead of calling insertions based on one outgroup alignment only, PAGAN assigns weights to the graph edges and adjusts them according to the phylogenetic evidence. The edges connecting the rest of the graph to the apparent insertions can be made less favourable with growing support for these sites being true insertions or completely removed if the evidence is considered sufficient; conversely, edge weights can be reset in the light of later evidence that the sites are not insertions. The principle of insertion calling and edge weighting is the same for the *de novo* alignment and the reconstruction of sequence history for a reference alignment.

3 Algorithm

Notation: The alignment is performed according to a rooted binary guide tree that for N extant sequences consists of $2N$-1 *nodes* and $2N$-2 *branches* connecting them. We describe the tree with its *root* node drawn to the extreme left; a node is *terminal* if it is not connected to nodes on its right side, otherwise it is *internal*. An internal node is the *parent* of two *child* nodes that are *sisters* to each other. Each node is associated with a sequence.

Sequences are represented by partial order graphs that consist of *vertices* and *edges*. We consider a pairwise alignment of sequence graphs at sister nodes x and y. Graphs x and y consist of vertices $x_0, x_1, \ldots, x_n, x_{n+1}$ and $y_0, y_1, \ldots, y_m, y_{m+1}$. Start vertices x_0 and y_0 and end vertices x_{n+1} and y_{m+1} are empty and have no incoming and no outgoing edges, respectively; their only purpose is to define unique end points for the graph. Other vertices are associated with a character, representing a set of nucleotides, amino acids or codons, and have at least one incoming and one outgoing edge. For the standard alignment algorithm, only the incoming edges are relevant: edges $e^g_{x_i}$ connect vertex x_i to preceding vertices x_j $(j < i)$ and have weights associated with them; the superscript g indicates that a vertex may have multiple incoming edges. Function $\mathsf{chr}(.)$ gives the character associated to a vertex, and functions $\mathsf{lft}(.)$ and $\mathsf{wgt}(.)$ give the index of the left end of an edge (i.e. start vertex) and the weight of an edge, respectively.

The pairwise alignment of graphs x and y defines a new graph, z, that represents their parent. By breaking a guide phylogeny into parent-children triplets and resolving these in order from the nodes towards the root, the pairwise algorithm can be generalised to progressive multiple alignment. Details of how the pairwise alignments are used to construct the parent graph, i.e. define the vertices and transfer the edges from the child graphs, are given in the sections below.

Alignment algorithm: Our affine-gap pairwise alignment algorithm is loosely based on [7, 2, 14]. The algorithm consists states M, X and Y where either two sequence vertices are matched, a sequence vertex is matched against a gap or a gap is matched against a sequence vertex, respectively. Moves to M are associated with a cost, a normalised evolutionary score for matching characters at vertices x_i and y_j, given by function:

$$\mathsf{sco}(x_i, y_j) = log\left(\frac{q_z\, P(\,\mathsf{chr}(x_i),\, \mathsf{chr}(y_j);\, t\,)}{q(\,\mathsf{chr}(x_i)\,)\, q(\,\mathsf{chr}(y_j)\,)}\right) \tag{1}$$

where $q(a)$ is the equilibrium frequency of character a, $q_z = (\,q(\,\mathsf{chr}(x_i)\,) + q(\,\mathsf{chr}(y_j)\,)\,)/2$, and $P(a, b; t)$ is the substitution probability between characters a and b given the evolutionary distance t and the substitution model. For $P(a, b; t)$, we use the models of Tamura and Nei [24], Whelan and Goldman [25] and Kosiol, Holmes and Goldman [10] and the evolutionary distances as provided in the guide tree. Moves to X and Y are associated with the gap opening cost δ and moves within them with the gap extension cost ϵ. The former, $\delta = log(1 - e^{-rt})$, is a function of the insertion-deletion rate r and the evolutionary distance t, and the latter, $\epsilon = log(\varepsilon)$, of the gap extension probability ε. Parameters r and ε can be provided by the user; the default values are based on empirical estimates from literature.

Each move is additionally associated with the probability of the edges used; for computational reasons, the edge probabilities are converted to log-scale and are hereafter called 'edge weights'. In the alignment of two linear graphs representing sequences with no uncertain sites, all edge weights equal zero and the algorithm reduces to the standard one; cases where the edge weights can be non-zero, e.g. for graphs representing ancestral sequences or uncertainties in extant sequences, are explained below. Moves to X only affect index i and those to Y index j, and thus have the associated costs $\mathsf{wgt}(e^g_{x_i})$ and $\mathsf{wgt}(e^h_{y_j})$, respectively. Moves to M and to the end state affect both indices and have the associated cost $\mathsf{wgt}(e^g_{x_i}) + \mathsf{wgt}(e^h_{y_j})$.

It is notable that, in comparison to the traditional scoring function (e.g. ref. [2], page 15), our scoring function (Equation (1)) has additional term q_z and thus does not produce standard log-odds scores. We do not

fully understand why the performance of this function is superior to that of the standard function but suspect that the explanation is related to the way that progressive algorithms represent ancestral sequences and also include sites inserted in the descendants (see sub-section **Reconstruction of ancestral graph**). The function used here follows the one implemented in PRANK [14] and may not optimal for all alignment tasks with the graph approach due to the differences in the modelling of insertions. We intend to study the scoring functions in greater detail and believe that partial-order graphs provide more elegant ways to correct for the length of ancestral sequences if that indeed is the explanation. Improvements in the scoring function–and thereafter computation of a meaningful full probability for a solution–should also allow for the estimation of algorithm parameters from the data.

Alignment recursion: The recursions to find the optimal alignment for two graphs resemble those for the pairwise alignment of sequences with an affine gap cost [7]. A recursive computation defines matrices v^X, v^Y and v^M as the scores of obtaining the alignment $x_1 \dots x_i{:}y_1 \dots y_j$ by the extension of sub-alignment $x_1 \dots x_{i-k}{:}y_1 \dots y_j$ or $x_1 \dots x_i{:}y_1 \dots y_{j-l}$ by a gap, or $x_1 \dots x_{i-k}{:}y_1 \dots y_{j-k}$ by a match, respectively (where $k \geq 1; l \geq 1$).

However, the recursions for the graph alignment differ from the standard affine-gap alignment in two aspects: first, the graph approach incorporates the edge weights into the alignment cost and, second, it chooses a move to the current state not only from the possible preceding states but also, within each state, from all preceding cells connected to the current cell by incoming edges. As in the standard affine-gap alignment, pointer matrices recording the move chosen at each cell (i, j) need to be stored. For the graph alignment, however, we need to record both the moves between v^X, v^Y and v^M, and the edges that were used.

The recursions for v^X, v^Y and v^M can then be defined as:

Initialisation: $v^\star(i,\text{-}1), v^\star(\text{-}1,j)$ are set to $-\infty$; $v^X(0,0) = v^Y(0,0) = -\infty$; $v^M(0,0) = 0$.
Recursion: for $i = 1, ..., n$; $j = 1, ..., m$:

$$v^X(i,j) = \max_g \begin{cases} v^X(\,\mathsf{lft}(e^g_{x_i}), j)\,) + \mathsf{wgt}(e^g_{x_i}) + \epsilon \\ v^Y(\,\mathsf{lft}(e^g_{x_i}), j)\,) + \mathsf{wgt}(e^g_{x_i}) + \delta \\ v^M(\,\mathsf{lft}(e^g_{x_i}), j)\,) + \mathsf{wgt}(e^g_{x_i}) + \delta \end{cases};$$

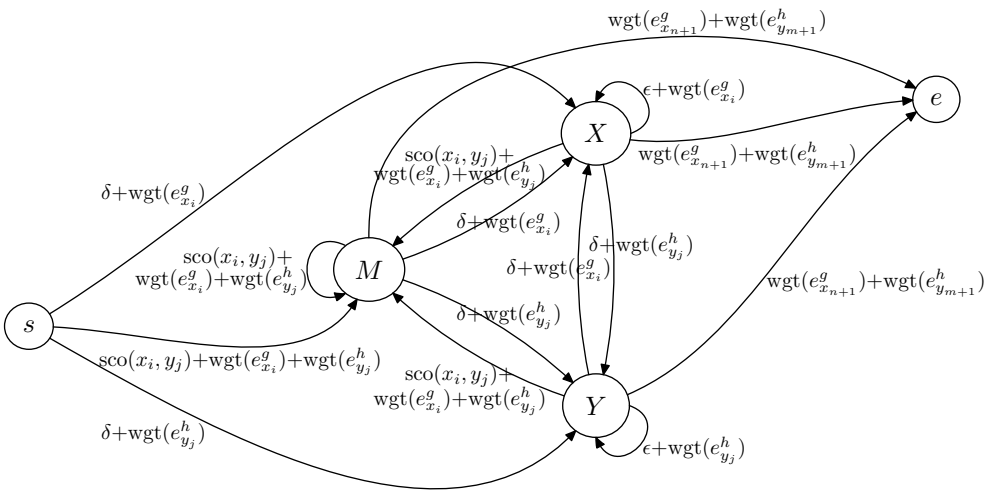
$$v^Y(i,j) = \max_h \begin{cases} v^X(\,i, \mathsf{lft}(e^h_{y_j}))\,) + \mathsf{wgt}(e^h_{y_j}) + \delta \\ v^Y(\,i, \mathsf{lft}(e^h_{y_j}))\,) + \mathsf{wgt}(e^h_{y_j}) + \epsilon \\ v^M(\,i, \mathsf{lft}(e^h_{y_j}))\,) + \mathsf{wgt}(e^h_{y_j}) + \delta \end{cases}; \tag{2}$$

$$v^M(i,j) = \max_{g,h} \begin{cases} v^X(\,\mathsf{lft}(e^g_{x_i}), \mathsf{lft}(e^h_{y_j})\,) + \mathsf{wgt}(e^g_{x_i}) + \mathsf{wgt}(e^h_{y_j}) + \mathsf{sco}(x_i, y_j) \\ v^Y(\,\mathsf{lft}(e^g_{x_i}), \mathsf{lft}(e^h_{y_j})\,) + \mathsf{wgt}(e^g_{x_i}) + \mathsf{wgt}(e^h_{y_j}) + \mathsf{sco}(x_i, y_j) \\ v^M(\,\mathsf{lft}(e^g_{x_i}), \mathsf{lft}(e^h_{y_j})\,) + \mathsf{wgt}(e^g_{x_i}) + \mathsf{wgt}(e^h_{y_j}) + \mathsf{sco}(x_i, y_j) \end{cases}.$$

Termination:

$$v^E = \max_{g,h} \begin{cases} v^X(\,\mathsf{lft}(e^g_{x_{n+1}}), \mathsf{lft}(e^h_{y_{m+1}})\,) + \mathsf{wgt}(e^g_{x_n+1}) + \mathsf{wgt}(e^h_{y_m+1}) \\ v^Y(\,\mathsf{lft}(e^g_{x_{n+1}}), \mathsf{lft}(e^h_{y_{m+1}})\,) + \mathsf{wgt}(e^g_{x_n+1}) + \mathsf{wgt}(e^h_{y_m+1}) \\ v^M(\,\mathsf{lft}(e^g_{x_{n+1}}), \mathsf{lft}(e^h_{y_{m+1}})\,) + \mathsf{wgt}(e^g_{x_n+1}) + \mathsf{wgt}(e^h_{y_m+1}) \end{cases}. \tag{3}$$

The dependencies between the states in the recursions can be depicted as:



where s indicates the start at $v^M(0,0)$ and e the end at v^E.

Reconstruction of ancestral graph: The optimal alignment for two graphs defines a new graph representing their ancestor. The matching of vertices included in the optimal solution is found using a standard algorithm that backtracks the edges used, starting from the end vertices and finishing at the start vertices. Each move backwards in the alignment matrices defines a new vertex in the ancestor graph: this vertex has one descendant (a vertex in either x or y) if the alignment has created a gap (moves to states X and Y; vertices 3 and 4 of graph X in Supplementary Figure 3c) and two descendants (vertices in both x and y) if the alignment has matched the vertices (moves to state M; vertices 1, 2, 5 and 6 of graph X in Supplementary Figure 3c). Pointers to descendant vertices are stored such that the inferred homology can be reconstructed after finishing the full multiple alignment.

When the chosen edge connects a vertex to another one not immediately preceding the current, also the vertices in-between them are included in the reconstructed ancestral graph with flags indicating their status as skipped-over positions. The main advantage from this is that it allows postponing the insertion calling to a later point (see below) and thus makes the inference of type of mutation events, insertions or deletions, less sensitive to errors in the order of aligning the sequences. However, their inclusion also means that the length of a graph for an ancestor may not correspond to the true length of the ancestral sequence but is as long as the alignment of graphs below it. This follows the practice used in progressive algorithms with an important difference: the status flags at the vertices record the effective path used for each alignment and thus allow inferring the true composition and length of ancestral sequences.

When reconstructing an ancestral graph, the incoming edges of child vertices are transferred to the parent without creating duplicates. This automatically connects a vertex immediately after a gap to two vertices, one vertex within the gap and the other vertex immediately before the gap (see graph X in Supplementary Figure 3c); these two edges represent the uncertainty of the cause of the gap, either an insertion or a deletion, and thus the uncertainty of the true composition of the ancestral sequence. In some situations, the length difference between two graphs is not caused by a natural mutation event and thus should not be considered as a potential insertion whose correct alignment requires an additional edge to be added in the graph. We consider two causes for this, terminal gaps caused by missing data in one of the graphs and spacers between paired-end reads. In these cases, no edge skipping over the gap is created (in the case of terminal gaps this is optional) and the gap created in one alignment cannot be freely re-used in a later alignment. However, one has to ensure that each vertex has both an incoming and an outgoing edge; if necessary, an edge to a vertex immediately adjacent to the current one is added.

Edge weights for reconstructed edges: A vertex may have multiple incoming edges, their potentially different weights representing the probability of the true preceding vertex. For ancestral sequences, multiple

edges allow description of the ambiguity of their true composition, the presence of characters at some positions, caused by the different properties of insertion and deletions that cannot be distinguished by pairwise comparisons. Our graph representation has no limit for the number of incoming edges allowed for any vertex and, by modelling the edges as independent objects, information can be attached on them and then adjusted dynamically as a function of the progressing alignment.

We have implemented several functions to adjust the edge weights based on phylogenetic information. The simplest ones define the edge weights as a function of either the number of alignments (default option) or the cumulative evolutionary distance since the last time the edge was used. More complex functions have thresholds for the number of alignments or the evolutionary distance until which the edges can be re-used and set punitive weights for them (or remove the edges completely) after that. For all functions, the edge weighting is kept independent of the length of the fragment by only adjusting the edges connecting the fragment to the rest of the graph (see Supplementary Figure 3d). The performance of alternative functions for edge weighting and edge pruning will be assessed in future work.

Edge weights make the graph alignment algorithm "phylogeny-aware" and play a central part in *de novo* alignment of large sets of sequences. Their role in alignment extension is smaller and is further constrained by the possible errors in inferred reference alignments. With complete removal of edges, fragments appearing as insertions could erroneously be disconnected from the graph and, at a later stage of the process, the missing edges could make the alignment solution indicated by the reference sequences invalid. Less aggressive edge pruning does not appear to cause problems, though, and the alignment accuracy against graphs with redundant sets of edges is good.

Character states for reconstructed vertices: For computational efficiency, we sacrifice the probabilistic description of ancestral sequences [see 14] and instead use weighted parsimony to reconstruct the character states. To further speed up the alignment, we have implemented an approach that combines the ancestral state representation from the Fitch algorithm [4] with the variable cost scheme from the Sankoff algorithm [20]. Using this, we perform the "Up phase" of the parsimony reconstruction along with the progressive alignment stage (or when reading in a pre-defined alignment), possibly leaving some character states ambiguous, and then do the "Down phase" to resolve some of the ambiguous states after reaching the root of the guide phylogeny.

The character state of a vertex is associated with a set of real characters represented by an ambiguity symbol. Using Equation 1 and the standard ambiguity code to represent the possible nucleotides, we can pre-compute hash tables that give the weighted parsimony cost for matching any two nucleotide symbols (the minimum cost between characters in the two sets) and the corresponding ancestral character state for their parent vertex (either intersection or union of the sets, see [4]). The pre-computation of all combinations of character sets is feasible for nucleotides but not for amino acids or codons. For those, we only consider ambiguity states representing either the set of all characters/codons (`X` and `NNN`, respectively) and the sets representing combinations of two characters/codons. We pre-compute hash tables that give the weighted parsimony cost for matching any two ambiguity symbols and the corresponding ancestral character states, representing at most two characters/codons (or all characters/codons, if neither child node contains any information), for their parent vertex.

Character states for reconstructed vertices in NGS sequence placement: In phylogenetic placement of multiple sequences from the same organism, the overlapping parts of the new sequences are expected to be identical and any differences observed represent either heterozygosity (that we ignore), sequencing error, or placement/alignment error. For the placement of DNA sequences – which are expected to be highly redundant and of varying quality when coming from NGS platforms – we have implemented an alternative function to reconstruct the most probable ancestral character states for the pseudo parent nodes. Using the majority consensus rule, the ancestral character state is defined as the most frequent base at that position among the newly placed reads, using the standard parsimony in the case of a tie; if no new read is aligned at the position,

the normal character state reconstruction is used.

The advantage of the majority consensus rule is that a small number of low-quality or misaligned reads does not corrupt the progressive alignment process. Furthermore, the tracking of base frequencies at the reconstructed ancestral sequences allows consensus-based assembly of phylogenetically-related reads. The very last parent for a set of reads defines the contig to represent that particular read cluster. The alignment positions present only in a small number of reads can be ruled out as errors and excluded from the contig. For the positions that are present in the outgroup but not in the reads, the outgroup information is used to indicate the amount and character states of potentially missing sequence, bridging together unconnected regions and creating full-length contigs.

Pre-processing and graph representation of NGS data: A standard sequence graph consists of vertices $x_0, \ldots, x_{n+1}$ where the first and the last vertex are empty and vertices $x_1, \ldots, x_n$ each represent a character from a sequence. Each vertex is connected only to the vertex before it and the one after it.

For the alignment of sequence reads from NGS platforms, we have implemented functions to trim and mask the sequences based on their quality scores and to add information on the character uncertainty in the graphs constructed from them. We allow trimming the low-quality ends of the reads using sliding windows that progress inwards until the mean quality score exceeds a threshold; if the trimming reduces too much of the read, the read is completely discarded. Independently of trimming, we can mask sequence positions with low quality scores and replace them with `N`'s that indicate ambiguity and match with all other characters.

The quality values for NGS data represent the confidence on the base calls and are ill-suited for describing ambiguous lengths of mononucleotide runs in Roche 454 data. Instead of quality values, we model the length uncertainty as a function of the length itself and allow adding extra incoming edges to the vertex immediately following the mononucleotide run. For runs of three or four bases, we add an edge that skips over the last vertex and for runs longer than that a further edge that skips over the last two vertices. As the length errors are rather infrequent, the edges have unequal weights and the full length run is favoured. Weights are further adjusted based on the quality of the last base of the run.

NGS reads are not expected to be of full length and the creation of terminal gaps is automatically penalised less heavily than in normal global alignment. By default, terminal gaps have no gap open cost and have a lower gap extension cost. For paired-end data, the same lowered gap costs are applied to the spacers of read pairs. Overlapping paired-end reads are aligned pairwise using the same alignment algorithm but setting the terminal gap penalty very low. If the alignment indicates significant overlap with a high base identity, the pair is merged into one longer sequence. For each position, the higher-quality base call from the two reads is selected retaining its original quality score.

Placement of sequences: PAGAN supports guided and unsupervised placement of sequences. If the origin of data is known, the sequences can be assigned to specific nodes in the reference phylogeny using an extension of New Hampshire eXtended (NHX) tree format [26] and defining either a single target node or a set of target nodes for each sequence. If multiple nodes are given, the read is aligned against each of them and the proportion of identities inferred between the read and the target is used to select the best placement. If the origin is unknown, PAGAN can search for the optimal placement. This can be exhaustively using PAGAN's own alignment algorithm or, as that can be time-consuming, using the fast local alignment of Exonerate [23]. The two approaches can also be combined and Exonerate can either select one target node or provide a list of potential target nodes from which PAGAN chooses the optimal one. If the sequence matches multiple locations equally well, PAGAN can add it to all locations or just one of them.

4 Extension of real alignments

We tested PAGAN on the extension of real multiple alignments with new protein and DNA/NGS sequences.

4.1 Extension of EnsemblCompara GeneTrees alignments with protein fragments

We first used PAGAN to extend EnsemblCompara GeneTrees alignments with protein fragments. A recent update of Ensembl included Northern white-cheeked gibbon (*Nomascus leucogenys*) and all alignments and corresponding phylogenies had to be updated. Re-computation of thousands of large data sets is not only laborious but also technically difficult as many gene models built from low-coverage genome sequences are incomplete and the short protein fragments contain little information for their accurate placement and alignment. PAGAN can do unsupervised placement of sequences but also provides an option to limit the placement to certain locations. The use of such guided placement guarantees that the short fragments will not be placed to phylogenetically implausible locations and thus avoids the problem of "split genes". However, it also assumes that all possible target locations for the new sequences are known in advance; in the case of Ensembl, this probably is true for many well-sampled clades.

An example of PAGAN alignment of gibbon protein fragments to a reference alignment is shown in Supplementary Figure 4. We defined a guide tree where the ancestors of Hominidae for the different paralogues were tagged as potential target locations for the alignment and let PAGAN to find the best placement for each fragment. As the sequences were known to be fragmented, we also used the option to merge fragments placed at the same target node into longer contigs, the amount of missing data estimated from the target ancestor (Supplementary Figure 4c, bottom). The alignments created by PAGAN look good but, as they are based on real data, their correctness cannot be confirmed.

One should note that the relative alignment of the sequences in the reference alignment does not change in the alignment extension and only new data is added. In the case of large repositories such as Ensembl, this ensures that the new release will not contain unwanted surprises and the possible dependencies on the previous versions will not be broken.

4.2 Extension of EnsemblCompara GeneTrees alignments with NGS data

PAGAN includes several features especially targeted for the modelling and pre-processing of NGS data. We tested it in the extension of EnsemblCompara GeneTrees alignments with real Roche 454 data as well as fragmented true sequences with added sequencing noise.

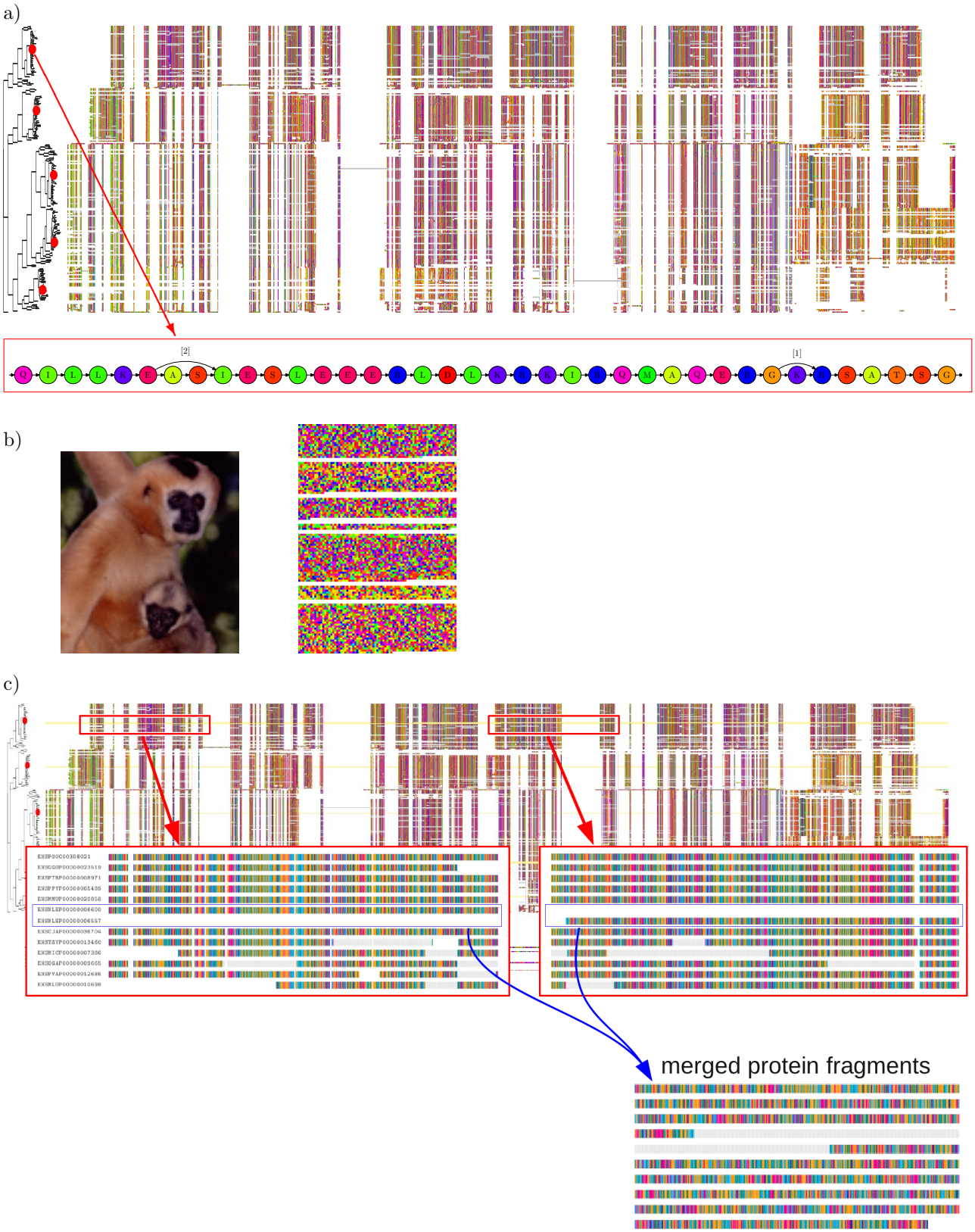
We downloaded Roche 454 reads for prairie vole (*Microtus ochrogaster*) from the NCBI SRA [21] and found a set of reads to be similar to the myosin genes. We named eight internal nodes in the phylogenetic tree, the ancestors of the rodent paralogues, as the potential target nodes (indicated with blue dots in the tree of Supplementary Figure 5) and let PAGAN find the best placement for the reads. Although the read coverage was relatively low, the placement created distinct paralogous clusters with the aligned reads showing high similarity to the surrounding sequences. When we analysed the same data using Newbler's cdna assembler (v.2.3) [17], the paralogue information was lost and all reads were collapsed into one contig, non-matching reads being discarded.

The correct placement and alignment for the prairie vole sequences are not known. To have more control on that, we created pseudo-NGS data by removing one of the species in the reference alignment, fragmenting its full-length sequences to short reads and adding empirical sequencing noise in the data. More precisely, we considered the same EnsemblCompara GeneTrees alignment for the myosin gene family and removed the five sequences from kangaroo rat (*Dipodomys ordii*). Based on these sequences, 125-base overlapping paired-end Illumina reads were created using simNGS [18] and then merged (overlapping pairs) and trimmed using

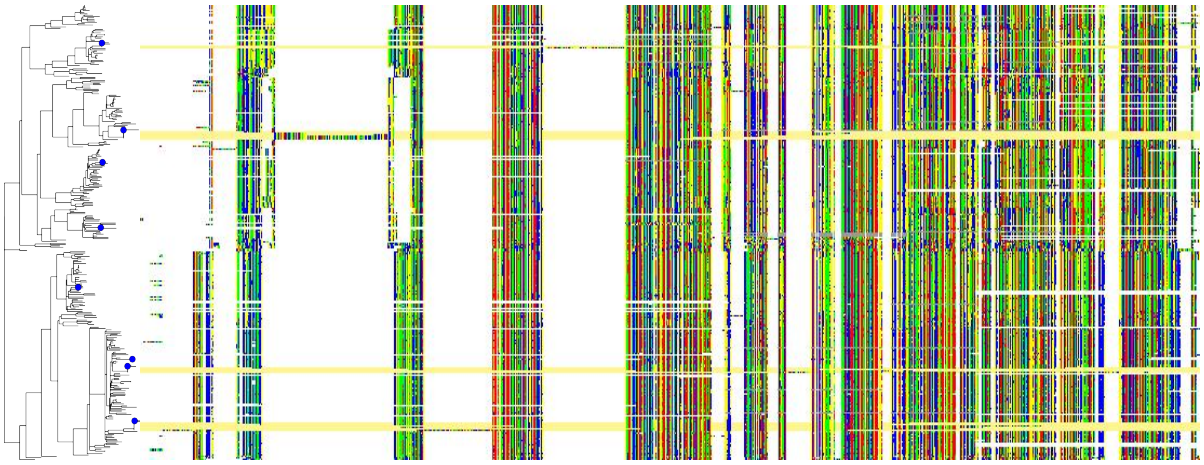
PAGAN. We named the same eight internal nodes in the myosin phylogeny as the potential target nodes and let PAGAN find the best placement for each read among those and then progressively align the reads to the chosen target nodes.

PAGAN placed most reads into clusters of related reads (Supplementary Figure 6a, background in different colours), and for the two top-most clusters (pink and cyan), the consensus contigs built of the aligned reads are nearly complete (523/528 and 504/510 of the sites, respectively) and contain no errors (Supplementary Figure 6c). However, two of the clusters (green and orange) are partially mixed and one cluster (yellow) is split in two places. Interestingly, seven and eight paralogous copies of myosin gene are known in mouse and rat, respectively, whereas only five copies are found in the low-coverage genome of kangaroo rat and not all of these group together with other rodent sequences in the EnsemblCompara GeneTrees phylogeny (Supplementary Figure 6a). The split of the yellow cluster between two locations suggests that that particular kangaroo rat gene has either been mis-assembled in the current gene model or some of its exons have been affected by a gene conversion or genomic arrangement event and are now more similar to a different gene in related species. Similar events may explain the incorrect placement of two other kangaroo rat sequences (green and orange) in the original alignment and the subsequent mixing of the read clusters.

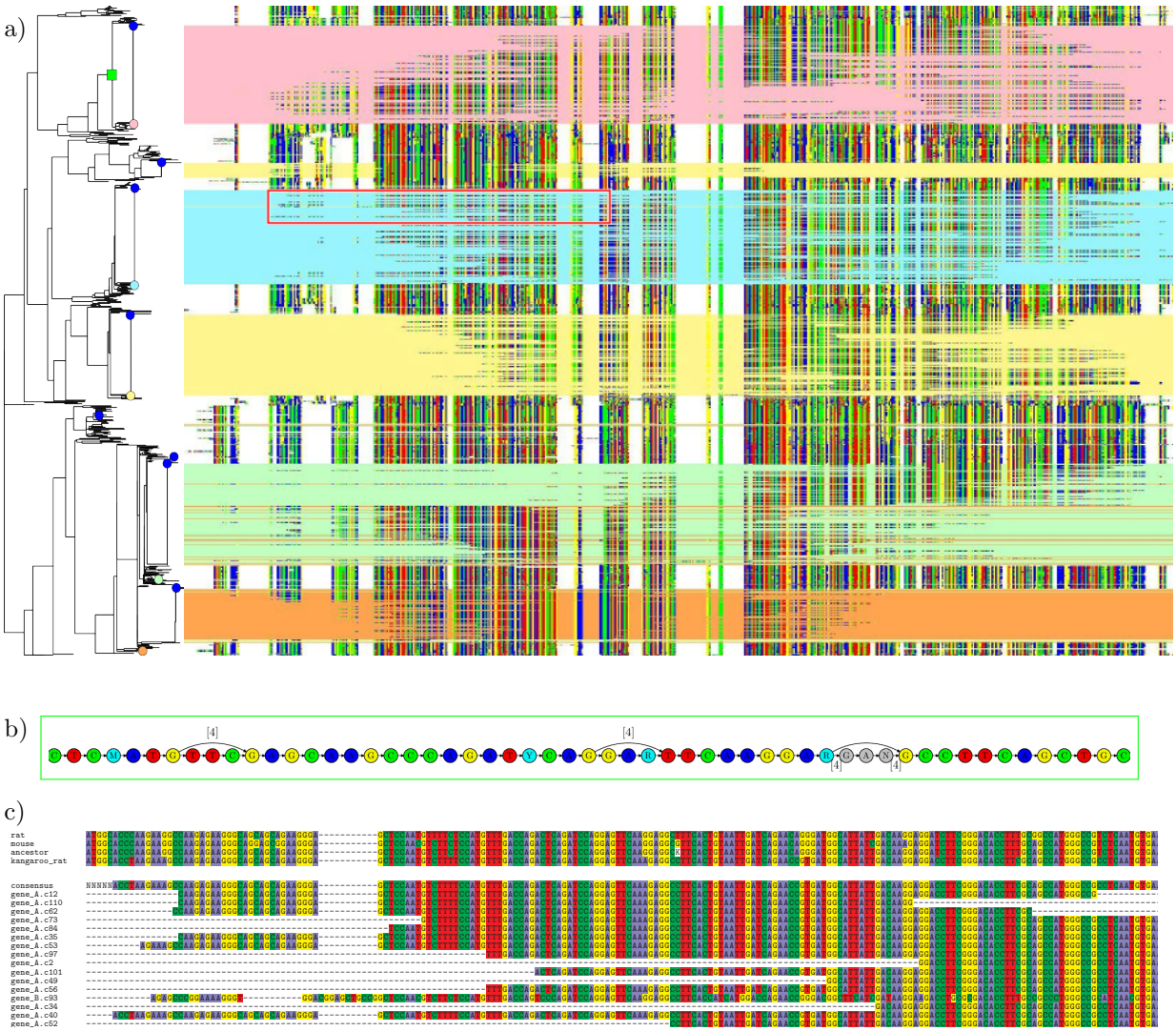
The examples show that PAGAN can accurately align very short fragments of transcripts, as short as potentially some of the raw Illumina reads that are being produced. Although we don't envision aligning read by read using PAGAN at this point in time, we expect that using very conservative pre-assemblies of raw reads will produce unambiguous short contigs, devoid of artifacts like chimaeric sequences from different transcripts. We consider these pre-assembled contigs as very suitable input for PAGAN in RNA-seq projects. Also, given that a FASTG standard format for sequence graphs is being finalised at the time of this writing (http://assemblathon.org), we expect that with minimal modifications, PAGAN will be able to use FASTG graphs from de novo pre-assembly of raw reads as suitable input.



Supplementary Figure 4: (a) The EnsemblCompara GeneTrees alignment ENSGT00620000087714 (version 1, e!v62) without the gibbon sequences (top). The ancestor of Hominidae for the different paralogues is indicated in the tree with red dots. The graph shows a part of a reconstructed ancestor with two uncertain gaps (bottom). (b) The gibbon genome is fragmented and the gene models are often incomplete. Seven protein fragments show high similarity to the gene family ENSGT00620000087714. (c) PAGAN finds the target node for each fragment and adds them to the reference alignment (top). PAGAN can also merge fragments placed to the same target node into a longer contig, estimating the amount of missing data from the closest ancestral sequence (bottom).



Supplementary Figure 5: We downloaded Roche 454 reads for prairie vole (*Microtus ochrogaster*) from the NCBI SRA [21] and found a set of reads to be similar to the rodent myosin genes. We named eight internal nodes in the EnsemblCompara GeneTrees phylogeny for the myosin gene family, the ancestors of the rodent paralogues, as the potential target nodes (blue dots) and let PAGAN find the best placement for the reads among those nodes and then progressively align the reads mapping to each target node to the reference alignment. This created distinct paralogous clusters (yellow background) and, with the exception of their 5' end, the aligned reads showed high similarity to the surrounding sequences. The long gaps in the 5' end of the alignment suggest that the added reads contain first exons that are different from the one in the Ensembl canonical transcript or include 5' UTRs not present in the reference alignment.



Supplementary Figure 6: (a) PAGAN places most of the simulated Illumina reads, based on five different myosin sequences from kangaroo rat (*Dipodomys ordii*; different background colours), in clusters of related sequences. The eight alternative locations (the node for mouse or rat or mouse/rat ancestor) to add the reads are indicated with blue dots and the phylogenetic positions of the original kangaroo rat CDS sequences are shown with dots whose colour matches the background of the aligned reads. (b) The graph representing an inferred ancestor marked by the green square in (a) shows positions with uncertain character state and additional edges indicating insertions or deletions. For the uncertain positions, the number of alignments since the edge was last used is shown in square brackets. (c) The kangaroo rat sequence from the cyan clade is more similar to the inferred ancestor than to the rat and mouse sequences (top). The reconstructed kangaroo rat contig ('consensus') covers 504 of the total 510 sites in the true sequence and has no errors (bottom, first sequence). A misplaced read ('gene_B') causes a gap in the alignment but does not affect the resulting contig sequence. The part of the alignment highlighted is indicated with a red box in (a); not all reads contributing to the contig are shown.

5 Comparison of alternative methods for alignment extension

We tested five alignment methods for both data types and additional two methods that only support DNA or protein data. The methods tested for both were PAGAN/guided, PAGAN/free, HMMER [3], MAFFT [9] and ClustalW [11]; for DNA we also used PaPaRa [1] and for protein ClustalO [22]. For PAGAN/guided, the species of origin (but not the correct paralogous copy) for the QS were provided, giving two and three target nodes for the Primate and Rodent QS. We assessed the accuracy of alignment extension by measuring the proportion of true homologies recovered between the QS and the closest human/mouse reference sequence.

Data simulation Test data were simulated using programs INDELible [5] and simNGS [18]. The model section in the INDELible configuration file was the following:

```
[MODEL] codonmodel
  [submodel]     2.5  0.15   //  M0 with kappa=2.5, omega=0.15
  [insertmodel]  POW  1.7 15 //  Power law insertion length distrib. (a=1.7, M=15)
  [deletemodel]  POW  1.8 15 //  Power law deletion length distrib. (a=1.8, M=15)
  [indelrate]    0.05        //  insertion rate = deletion rate = 0.05
```

and the length of the root sequences was 500 codons.

The query sequences (QS) for the extension were created by sampling fragments of the full length query using an in-house-built tool that could also handle amino-acid data.

The command used to simulate Illumina NGS reads was:

```
cat QS | simNGS --ncycles NC RF > QSQ
```

where the following abbreviations are used:

NC=number of sequencing cycles (30, 60 or 120)
RF=empirical runfile (s_3_4x.runfile, provided with simNGS)
QS=query sequences
QSQ=QS (fastq)

Reference phylogeny The reference phylogeny was inferred using the following command:

```
raxmlHPC -m GTRGAMMA -s RAP -n RT -o Ornithorhynchus_anatinus
```

where the following abbreviations are used:

RAP=reference alignment (phylip)
RT=reference phylogeny (newick)

Alignment extension The reference alignments were extended with new sequences using several different alignment methods. The commands used were:

PAGAN guided (v. 0.33)

```
pagan --ref-seqfile RA --ref-treefile RTX --queryfile QS --outfile OA --one-placement-only
```

PAGAN free (v. 0.33)

```
pagan --ref-seqfile RA --ref-treefile RT --queryfile QS --outfile OA --fast-placement
--test-every-internal-node --exhaustive-placement
```

HMMER (v. 3.0)

```
hmmbuild --enone --dna RAH RAS [ or hmmbuild --enone --amino RAH RAS ]
hmmalign --mapali RAS RAH QS > OAS
```

MAFFT (v. 6.860b)

```
mafft --add QS RA > OA
```

ClustalW (v. 2.1)

```
clustalw -profile1=RA -profile2=QS -sequences -output=fasta -outfile=OA
```

ClustalO (v. 1.0.3)

```
clustalo --p1=RA --in=QS --out=OA
```

PaPaRa (v. 7.2.6; RAxML)

```
papara -f X -n OAP -m GTRGAMMA -t RT -s RAP -X QS
```

where the following abbreviations are used:

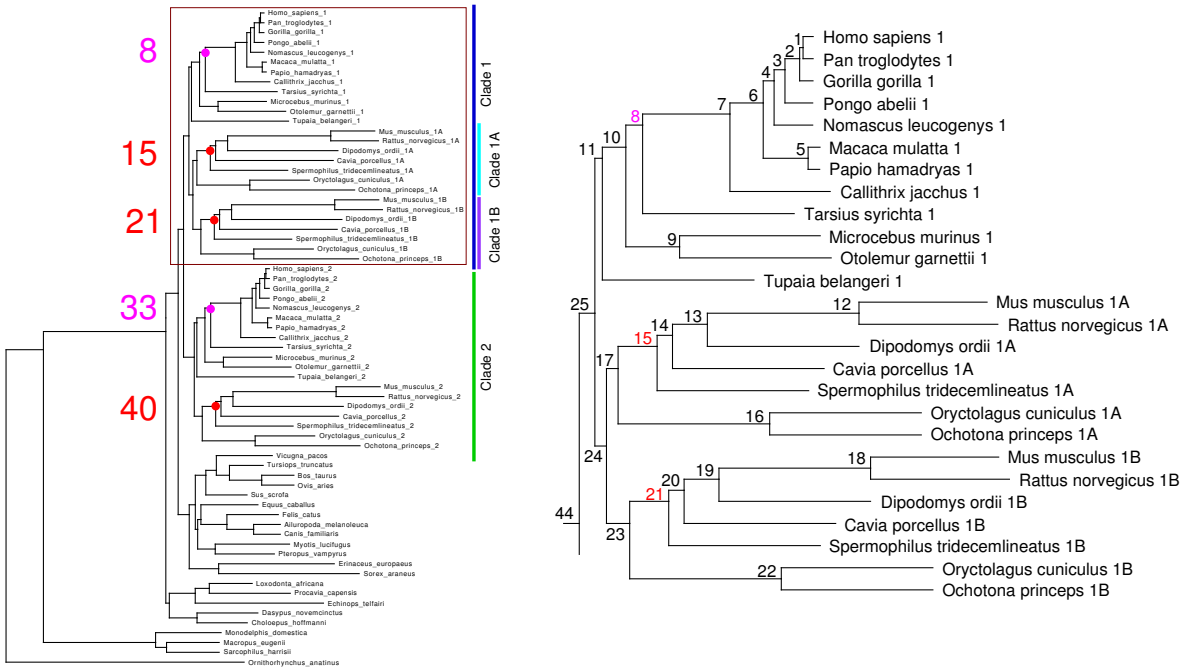
RA=reference alignment (fasta); RAS=RA (stockholm); RAP=RA (phylip); RAH=RA (hmm)
RT=reference phylogeny (newick); RTX=RT (nhx)
QS=query sequences
OA=output alignment (fasta); OAS=OA (stockholm); OAP=OA (phylip)

Additional tools were used to convert between different alignment formats.

HMMER was also tested without option `--enone` and MAFFT with option `--localpair` but these did not improve the results.

Correction for deleted characters The accuracy of PaPaRa-generated alignments was assessed after inferring the positions were it had created insertions (and subsequently deleted characters) in the query sequences. To do this, we walked through the two sequences (i.e. in the input and output) and, whenever the character states did not match, inserted gaps in the aligned query sequence. In the cases that a sub-string deleted by PaPaRa starts with the same character(s) that immediately follow(s) it, this approach cannot infer the position of the gap with certainty and may overestimate the error. However, this is true in any other analysis trying to re-create full sequences from PaPaRa alignments: once characters have been deleted, the sequences cannot be reconstructed without ambiguity.

Memory usage, run times The memory usage was tested on a Linux system and is based on the maximum value of `VmSize` in `/proc/$pid/status` during the run of the program. The execution times are the real times used on an otherwise idle system.



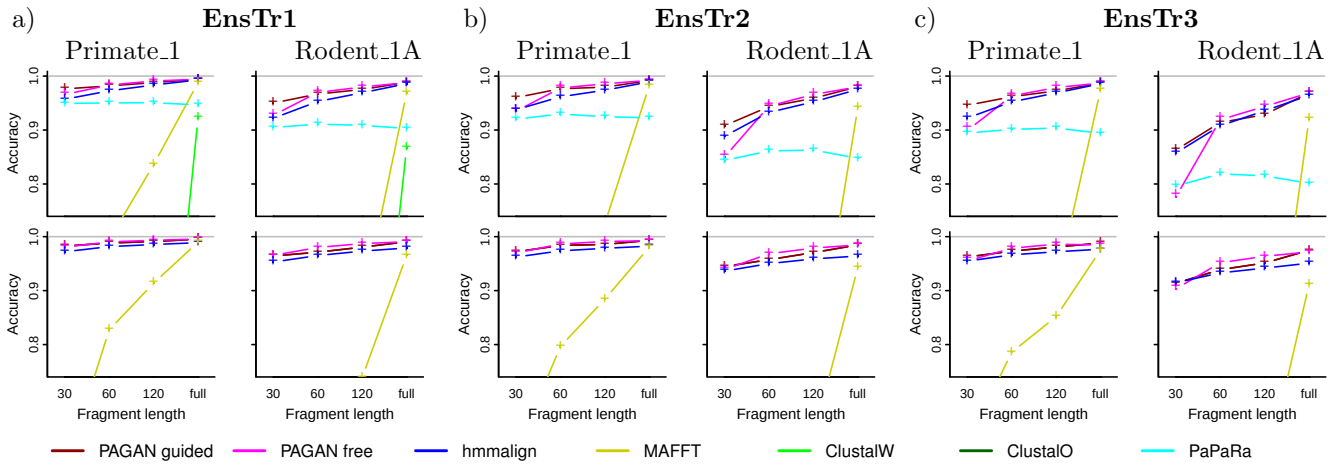
Primate_1

PAGAN	node	EnsTr1 30	EnsTr1 60	EnsTr1 120	EnsTr1 full	EnsTr2 30	EnsTr2 60	EnsTr2 120	EnsTr2 full	EnsTrT3 30	EnsTrT3 60	EnsTrT3 120	EnsTrT3 full
guided	8	93	93	97	100	89	94	97	100	90	95	99	100
	33	7	7	3	0	11	6	3	0	10	5	1	0
free	1	7	4	1	0	7	3	0	0	5	2	0	0
	6	1	1	1	0	1	2	0	0	2	1	0	0
	7	5	6	3	0	5	4	3	0	6	4	2	0
	8	19	38	56	76	25	43	56	70	31	43	51	68
	9	1	2	1	0	2	2	1	0	2	1	1	0
	10	4	8	17	19	8	12	21	26	10	16	26	30
	11	1	4	6	2	2	4	5	3	3	5	5	1
	17	2	3	1	0	2	2	0	0	2	1	1	0
	23	3	2	1	0	2	1	0	0	1	1	0	0
	24	5	4	2	1	3	2	1	0	2	2	1	0
	25	1	3	5	1	1	3	4	1	2	5	6	1
	44	6	5	2	1	4	4	2	0	3	3	1	0

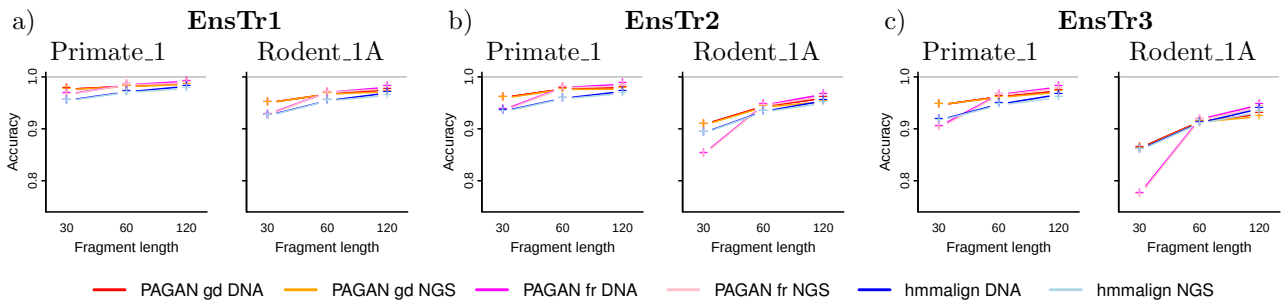
Rodent_1A

PAGAN	node	EnsTr1 30	EnsTr1 60	EnsTr1 120	EnsTr1 full	EnsTr2 30	EnsTr2 60	EnsTr2 120	EnsTr2 full	EnsTr3 30	EnsTr3 60	EnsTr3 120	EnsTr3 full
guided	15	85	87	93	99	78	83	93	100	74	82	92	100
	21	9	8	5	0	14	12	6	0	15	13	7	0
	40	6	5	2	1	8	5	1	0	11	6	1	0
free	10	2	1	1	0	3	2	1	0	3	2	1	0
	11	1	2	3	1	3	4	3	0	4	4	3	0
	13	14	9	4	0	10	6	3	0	8	4	2	0
	14	12	18	20	6	13	14	15	7	14	13	12	5
	15	12	25	44	78	16	29	43	81	18	30	44	80
	16	2	1	0	0	2	1	0	0	2	1	0	0
	17	4	7	10	7	4	7	11	2	4	9	12	6
	21	2	1	0	0	1	1	0	0	1	1	0	0
	23	2	2	1	0	2	2	1	0	2	2	1	0
	24	3	4	5	4	3	5	6	7	2	5	8	6
	25	1	3	3	3	2	3	5	3	2	5	6	3
	44	4	3	1	0	2	2	1	0	2	3	2	0

Supplementary Figure 7: The target nodes for the extension using the PAGAN guided approach are shown in the full reference tree (left; magenta for Primate, red for Rodent). With PAGAN free, most QS for Primate_1 and Rodent_1A are expected to fall in the top part of the tree (right). The tables show the percentage of protein QS placed at the corresponding node for Primate_1 and Rodent_1A. Placement to a wrong node typically indicates high sequence conservation and does not necessarily affect the accuracy of the homology inference. The results are based on analyses using the true simulated reference phylogeny; other results are based on analyses with inferred phylogenies.



Supplementary Figure 8: The accuracy of alignment of DNA (top row) and protein (bottom row) query sequences against the corresponding reference alignment using different alignment methods. The x-axis indicates the length of the fragments aligned and the sub-panels show two of the five query species analysed. Columns a–c correspond to trees with branch lengths multiplied by 1.5, 2.0 and 2.5, respectively. The accuracy is measured as the correctness of the site-wise homology inference with respect to the closest human/mouse reference sequence.



Supplementary Figure 9: The accuracy of alignment of query sequences with (NGS) and without (DNA) sequencing noise. Sub-panels a–c correspond to trees with branch lengths multiplied by 1.5, 2.0 and 2.5, respectively.

6 Extension of large alignments

The data were downloaded from http://www.cs.utexas.edu/users/phylo/software/sepp/submission/sims.tar.gz. We analysed half of each test set (M2-M4; replicates R0-R9) using PAGAN and hmmalign. We used the true simulated reference alignments and, for the PAGAN analyses, the true reference phylogeny with branch lengths estimated with RAxML. For hmmalign, we tested also option `--enone` but found it producing slightly less accurate results. The commands used were:

PAGAN fast heuristics (v. 0.37)

```
pagan --ref-seqfile RA --ref-treefile RT --queryfile QS --outfile OA --very-fast-placement
--test-every-node
```

HMMER (v. 3.0)

```
hmmbuild --dna RAH RAS
hmmalign --mapali RAS RAH QS > OAS
```

The accuracy of the resulting extended alignment was measured as the proportion of true homologies recovered between the QS and its closest reference sequence (based on the original simulation phylogeny). False homologies were not penalised and correctness of insertions inferred were not measured. The reported values are the mean accuracies and the proportion of fragments aligned by each method.

7 Impact of reference alignment composition to alignment accuracy

Data simulation Test data were simulated using programs INDELible [5] and simNGS [18]. The model section in the INDELible configuration file was the following:

```
[MODEL] codonmodel
  [submodel]     2.5  0.5    //  M0 with kappa=2.5, omega=0.5
  [indelmodel]   POW  1.8 30 //  Power law ins-del length distrib. (a=1.8, M=30)
  [indelrate]    0.05        //  insertion-deletion rate = 0.05
```

and the length of the root sequences was 500 codons.

Synthetic NGS reads were created with simNGS using the commands:

```
cat FQ | simLibrary --insert 1 --readlen 90 --coverage 5 --bias 1 > QS
cat QS | simNGS --ncycles 125 --paired paired -o fastq RF > QSQ
```

where the following abbreviations are used:

FQ=full query sequence
RF=empirical runfile (s_4_0033.runfile)
QS=query sequences
QSQ=QS (fastq)

The target length for the fragment library was 181 bases. The simulation produced approximately 41–43 fragments per sequence, their lengths varying from 79 to 448 bases (mean 180.3). Fragments shorter than 130 bases were discarded. Sequencing was simulated with 5x coverage and using 125-base pair-ended reads. In the end, we obtained approximately 74–76 reads per RA.

Alignment extension The reference alignments were extended with new sequences using PAGAN guided and HMMER. The commands used were:

PAGAN guided (v. 0.33)

```
pagan --ref-seqfile RA --ref-treefile RTX --queryfile QSQ --outfile OA
```

HMMER (v. 3.0)

```
hmmbuild --dna RAH RAS
hmmalign --mapali RAS RAH QS > OAS
```

where the following abbreviations are used:

RA=reference alignment (fasta); RAS=RA (stockholm); RAH=RA (hmm)
RTX=reference phylogeny (nhx)
QSQ=query sequences (fastq); QS=QSQ reduced to fasta
OA=output alignment (fasta); OAS=OA (stockholm);

The true simulation tree was used as RTX in PAGAN analyses. Additional tools were used to convert between different alignment formats. HMMER was also tested with option `--enone` but that did not improve the results.

Re-alignment The reference alignments were re-aligned with PAGAN and MAFFT. The commands used were:

PAGAN (v. 0.33)

```
pagan --seqfile RS --treefile RT --outfile OA
```

MAFFT (v. 6.860b)

```
mafft --treein RTM RS > OA
```

where the following abbreviations are used:

RS=reference sequences;
RT=reference phylogeny
RTM=RT (mafft)
OA=output alignment (fasta)

For PAGAN, $r = 0.01$ and $\varepsilon = 0.9$ were used. The true simulation tree was used as RT in PAGAN alignments and as the basis for RTM in MAFFT alignments. Additional tools were used to convert between different tree formats.

Supplementary Table 1: The accuracy of extending re-aligned reference alignments.

Simulation		PAGAN		HMR/full		HMR/clade		PAGAN		HMR/full		HMR/clade	
		Ingroup		Ingroup		Ingroup		Ingroup		Ingroup		Ingroup	
dpth	query	large	small	large	small	large	small	large	small	large	small	large	small
		PAGAN-generated reference alignments						MAFFT-generated reference alignments					
0.30	close	0.976	0.976	0.932	0.927	0.943	0.966	0.976	0.976	0.927	0.915	0.944	0.966
	interm.	0.965	0.963	0.934	0.924	0.943	0.941	0.963	0.960	0.926	0.914	0.940	0.939
	distant	0.954	0.947	0.932	0.921	0.931	0.913	0.948	0.943	0.926	0.911	0.928	0.910
0.45	close	0.977	0.977	0.914	0.904	0.930	0.966	0.977	0.977	0.900	0.875	0.927	0.965
	interm.	0.959	0.956	0.918	0.899	0.930	0.931	0.957	0.952	0.900	0.867	0.924	0.928
	distant	0.944	0.917	0.919	0.897	0.914	0.877	0.932	0.904	0.897	0.864	0.903	0.868
0.60	close	0.976	0.974	0.880	0.855	0.905	0.956	0.974	0.972	0.843	0.798	0.897	0.955
	interm.	0.945	0.933	0.886	0.847	0.906	0.895	0.938	0.925	0.845	0.795	0.890	0.890
	distant	0.913	0.850	0.878	0.832	0.875	0.806	0.887	0.842	0.841	0.790	0.855	0.803

References

[1] SA Berger and A Stamatakis. Aligning short reads to reference alignments and trees. *Bioinformatics*, 27:2068–2075, Aug 2011.

[2] R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. *Biological sequence analysis: Probabilistic models of proteins and nucleic acids*. Cambridge university press Cambridge, UK, 1998.

[3] S Eddy. HMMER 3.0 (`http://hmmer.org`), 2010.

[4] W Fitch. Toward defining the course of evolution: minimum change for a specific tree topology. *Syst Zoology*, 20:406–416, 1971.

[5] W. Fletcher and Z. Yang. INDELible: a flexible simulator of biological sequence evolution. *Mol Biol Evol*, 26:1879–1888, 2009.

[6] B Flusberg, D Webster, J Lee, K Travers, E Olivares, T Clark, J Korlach, and S Turner. Direct detection of DNA methylation during single-molecule, real-time sequencing. *Nat Meth*, 7:461–465, 2010.

[7] O. Gotoh. An improved algorithm for matching biological sequences. *J Mol Biol*, 162:705–708, 1982.

[8] S. Huse, J. Huber, H. Morrison, M. Sogin, and D. Welch. Accuracy and quality of massively parallel DNA pyrosequencing. *Genome Biol*, 8:R143, 2007.

[9] 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.

[10] C Kosiol, I Holmes, and N Goldman. An empirical codon model for protein sequence evolution. *Mol Biol Evol*, 24:1464–1479, 2007.

[11] M. Larkin, G. Blackshields, N. Brown, R. Chenna, P. McGettigan, H. McWilliam, F. Valentin, I. Wallace, A. Wilm, R. Lopez, J. Thompson, T. Gibson, and D. Higgins. Clustal W and Clustal X version 2.0. *Bioinformatics*, 23:2947–2948, 2007.

[12] H Li and R Durbin. Fast and accurate short read alignment with Burrows-Wheeler transform. *Bioinformatics*, 25:1754–1760, 2009.

[13] H Li, J Ruan, and R Durbin. Mapping short DNA sequencing reads and calling variants using mapping quality scores. *Genome Res*, 18:1851–1858, 2008.

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

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

[16] A. Löytynoja and N. Goldman. Uniting alignments and trees. *Science*, 324:1528–1529, 2009.

[17] M. Margulies, M. Egholm, W. Altman, S. Attiya, J. Bader, L. Bemben, J. Berka, M. Braverman, Y. Chen, Z. Chen, et al. Genome sequencing in open microfabricated high density picoliter reactors. *Nature*, 437:376–380, 2005.

[18] T Massingham and N Goldman. simNGS and simLibrary (`http://www.ebi.ac.uk/goldman-srv/simNGS`), 2012.

[19] SB Needleman and CD Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J Mol Biol*, 48:443–453, 1970.

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

[21] EW Sayers, T Barrett, DA Benson, E Bolton, SH Bryant, K Canese, V Chetvernin, DM Church, M DiCuccio, S Federhen, M Feolo, IM Fingerman, LY Geer, W Helmberg, Y Kapustin, D Landsman, DJ Lipman, Z Lu, TL Madden, T Madej, DR Maglott, A Marchler-Bauer, V Miller, I Mizrachi, J Ostell, A Panchenko, L Phan, KD Pruitt, GD Schuler, E Sequeira, ST Sherry, M Shumway, K Sirotkin, D Slotta, A Souvorov, G Starchenko, TA Tatusova, L Wagner, Y Wang, WJ Wilbur, E Yaschenko, and J Ye. Database resources of the National Center for Biotechnology Information. *Nucleic Acids Res*, 39:D38–51, 2011.

[22] F Sievers, A Wilm, D Dineen, TJ Gibson, K Karplus, W Li, R Lopez, H McWilliam, M Remmert, J Söding, JD Thompson, and DG Higgins. Fast, scalable generation of high-quality protein multiple sequence alignments using clustal omega. *Mol Syst Biol*, 7:539, 2011.

[23] GS Slater and E Birney. Automated generation of heuristics for biological sequence comparison. *BMC Bioinformatics*, 6:31, 2005.

[24] K Tamura and M Nei. Estimation of the number of nucleotide substitutions in the control region of mitochondrial DNA in humans and chimpanzees. *Mol Biol Evol*, 10:512–526, 1993.

[25] S Whelan and N Goldman. A general empirical model of protein evolution derived from multiple protein families using a maximum-likelihood approach. *Mol Biol Evol*, 18:691–699, 2001.

[26] C Zmasek. New Hampshire eXtended 2.0 (`http://phylosoft.org/NHX`), 2008.