Chapter 3

Multiple Alignment of DNA Sequences with MAFFT

Kazutaka Katoh, George Asimenos, and Hiroyuki Toh

Abstract

Multiple alignment of DNA sequences is an important step in various molecular biological analyses. As a large amount of sequence data is becoming available through genome and other large-scale sequencing projects, scalability, as well as accuracy, is currently required for a multiple sequence alignment (MSA) program. In this chapter, we outline the algorithms of an MSA program MAFFT and provide practical advice, focusing on several typical situations a biologist sometimes faces. For genome alignment, which is beyond the scope of MAFFT, we introduce two tools: TBA and MAUVE.

Key words: Multiple sequence alignment, progressive method, iterative refinement method, consistency objective function, genome comparison.

1. Introduction

Multiple sequence alignment (MSA) is used in various phases of biological studies, such as in SSU rRNA-based phylogeny inference for organismal classification *(1)*, or in informant sequence analysis for comparative gene finding *(2)*. Most current MSA programs successfully create an accurate alignment in a reasonable amount of time for easy cases, in which the number of input sequences is small (several dozen), the length of each sequence is short (up to ~500 nucleotides), and the sequences are highly similar (percent identity of ~70) to each other. Such cases require no special consideration. If all of these conditions are not met, however, then choosing appropriate tools and configurations, while considering the tradeoff between accuracy and computational cost, can be difficult.

Here we explain the basic algorithms implemented in an MSA program, MAFFT *(3, 4)*, and then explain how to select an appropriate option. Recent benchmark studies *(5, 6)* on RNA and DNA consistently identified MAFFT as one of the most accurate MSA

David Posada (ed.), *Bioinformatics for DNA Sequence Analysis,* Methods in Molecular Biology 537

DOI 10.1007/978-1-59745-251-9_3

methods in a wide range of situations. However, MAFFT cannot process all of the DNA alignment problems we may encounter. For example, MAFFT assumes that the order of aligned sites or blocks is completely conserved among all of the sequences. This assumption can be violated by genome inversions, translocations, or duplications. That is, genome alignment is beyond the scope of MAFFT. Thus we also explain two tools specialized to genome alignment: TBA *(7, 8)* and MAUVE *(9, 10)*. In addition, it is important to try out multiple different methods in order to obtain a high-quality alignment *(6, 11)*.

1.1. Basic Concepts

A sequence alignment is a set of corresponding residues among a collection of nucleotide or amino acid sequences. Aligned residues are usually interpreted to share an evolutionary origin and/or to be functionally equivalent. We mention here the alignment of multiple (>2) DNA sequences. A pairwise sequence alignment (composed of two sequences) can be calculated by the dynamic programming (DP) algorithm *(12–14)*. An MSA is usually displayed as a table, in which corresponding residues occupy the same column, as in **Fig. 3.1c**. When a sequence has no corresponding residue because of an insertion or deletion event, the position is displayed as "-" which is called a "gap." The sequences can be protein- or RNA-coding sequences or non-coding nucleotide sequences, such as introns or spacers.

The sequences involved in an MSA are usually assumed to be homologous, that is, derived from a single common ancestral sequence. Some MSA methods assume global homology while some other methods assume only a local homology (*see* **Note 2**). Most MSA programs do not attempt to filter non-homologous sequences, leaving the decision of what sequences to include in the MSA as an external decision for the user. However, this problem is sometimes important in comparative analysis, as discussed in **Section 3.3**.

1.2. MSA Strategies in MAFFT

1.2.1. Progressive Method

The progressive method *(15)* is currently the most widely used multiple alignment algorithm. ClustalW *(16)* and most of the other packages use this strategy with various modifications. In this strategy, a guide tree is created based on all-to-all pairwise comparisons, and an MSA is constructed using a group-to-group alignment algorithm at each node of the guide tree. In principle, the DP algorithm for pairwise alignment can be extended to that for group-to-group alignment. To achieve a reasonable balance between speed and accuracy, MAFFT uses a two-cycle progressive method (FFT-NS-2) shown in **Fig. 3.2**. In this method, low-quality all-pairwise distances are rapidly calculated, a tentative MSA is constructed, refined distances are calculated from the MSA, and then the second progressive alignment is performed (*see* **Note 1**). FFT-NS-1, a one-cycle progressive method, is faster and less accurate than FFT-NS-2. There is a more scalable option, PartTree *(17)*, applicable to ~50,000 sequences.

a

```
>Sequence1
nnnnnnattccggttgatcctgccggaggccactgctatcggggtccgactaagccatgc
gagtcaaggggctcccttcggggagcaccggcgcacggctcagtaacacgtggctaacct
accctcgggtgggggataacctcgggaaactgaggctaatcccccatagggaggaggtc
tggaatgatccctcccgaaaggcgtaagctgcccgaggatggggctgcggcggattagg
agtgaggccctgtccgcaagggcagggtcgaacttgggttcagcgaggggggcgaagtcg
taacaaggtagccgtaggggaactgcggctggatcacctccnnnnn
>Sequence2
nnnnnnactccggttgatcctgccggaggccaccgctatcggggtccgactaagccatgc
aagtcgagggccgcccggcaatgggggcggcccggcggacggctcagtaacacgtgggta
acctaccctcgggacggggataaccccggcgaaagtggggctaatccccgataggcgggg
cggcctggaacggtcctccgccgaaagggcccgggcccatgccgcccgggtccgcccgag
gatgggcctgcggccgattaggtagttggcggggtaacggcccgccaagccgataatcgg
tacgggcggtgagagccggagcccggagacggggactga
>Sequence3
agtggggaatcttggacaatgggggaaaccctgatccagcgacgccgcgtgcgggacgaa
gcccttcggggtgtaaaccgctgtggcgggggaagaataaggtagggaggaaatgcccta
tgtcggatgtgaaatcccacggctcaaccgtggggctgcatccgaaactaccaggcttgg
tct
```

b

```
>Sequence1
nnnnnnattccggttgatcctgccggaggccactgctatcggggtccgactaagccatgc
gagtcaaggggctccc----ttcggggagcaccggcgcacggctcagtaacacgtggcta
acctaccctcgggtgggggataacctcgg-gaaactgaggctaatcccccatagggaggg
aggtctggaatgatccctcccgaaagg----------------cgtaagctgcccgag
gatggggctgcggcggattaggagtgaggccctgtccgcaagggcagggtcgaacttggg
ttcagcgaggggggcgaagtcgtaacaaggtagccgtag---gggaactgcggctggatc
acctccnnnnn
>Sequence2
nnnnnnactccggttgatcctgccggaggccaccgctatcggggtccgactaagccatgc
aagtcgagggccgcccggcaatgggggcggcccggcggacggctcagtaacacgtgggta
acctaccctcgggacggggataaccccggcgaaagtggggctaatccccgataggcgggg
cggcctggaacggtcctccgccgaaagggcccgggcccatgccgcccgggtccgcccgag
gatgggcctgcggccgattaggtagttggcggggt-----aacggcccgccaagccgata
atcggtacgggcggtgag-------------agccggagcccggagacggggactga---
-----------
>Sequence3
ag--------tggggaatcttg--gacaatgggggaaaccctgatccagcgacgccgcgt
gcgggacgaagccctt----cggggtgtaaaccgctgtg---------------------
--------gcgggggaagaataaggtagg--------gaggaaatgccctatgtcggatg
tgaaatcccacg-----------------------------------------gctcaac
cgtggggctgcatccga--aactaccaggcttggtct----------------------
------------------------------------------------------------
-----------
```

c

```
CLUSTAL W (1.83) multiple sequence alignment

Sequence1       --NNNNNNATTCCGGTTGATCCTGCCGGAGGCCACTGCTATCGGGGTCCGACTAAGCCAT
Sequence2       --NNNNNNACTCCGGTTGATCCTGCCGGAGGCCACCGCTATCGGGGTCCGACTAAGCCAT
Sequence3       AGTGGGGAATCTTGGACAAT------GGGGGAAACCCTGATCCAG---CGACGCCGC-GT
                         *    **   **       ** **  **    ***  *   ****   **  *

Sequence1       GCGAGTCAAGGGGCTCCCTTC----GGGGAGCACCGGCGCACGGCTCAGTAACACGTGGC
Sequence2       GCAAGTCGAGGGCCGCCCGGCAATGGGGGCGGCCCGGCGGACGGCTCAGTAACACGTGGG
Sequence3       GCGGGACGAAGCCCTTCGGGGTGT-AAACCGCTGTGGCGGG-GGAAGAATAAGGTAGGGA
                **  * * * *  *  *             *    ****   **   * ***     **

Sequence1       TAACCTACCCT---CGGGTGGGGGATAACCTCGG-GAAACTGAGGCTAATCCCCCATAGG
Sequence2       TAACCTACCCT---CGGGACGGGGATAACCCCGGCGAAAGTGGGGCTAATCCCCGATAGG
Sequence3       GGAAATGCCCTATGTCGGATGTGAAATCCCACGGCTCAACCGTGGGG------CTGCATC
                  *  * ****     **  * * *   ** ***   **  * **        *   *

Sequence1       GGAGGAGGTCTGGAATGATCCCTCCCCGAAAGGCGTAAGCTGCCCGAGGATGGGGCTGCG
Sequence2       CGGGGCGGCCTGGAACGGTCCTCCGCCGAAAGGGCCCGG--GCCCATG---CCGCCCGGG
Sequence3       CGAAACTACCAGGCTTGGTCT-----------------------------------------
                 *       * **   * **

Sequence1       GCGGATTAGGAGTGAGGCCCTGTCCGCAAGGGCAGGGTCGAACTTGGGTTCAGCGAGGGG
Sequence2       TCCGCCCGAGGATGGGCCTGCGGCCGATTAGGTA--GTTGG---CGGGGTAA---CGGCC
Sequence3       ------------------------------------------------------------


Sequence1       GGCGAAGTCGTAACAAGGTAGCCGTAG-GGGAACTGCGGCTGGATCACCTCCNNNNN
Sequence2       CGCCAAGCCGATAATCGGTACGGGCGGTGAGAGCCGGAGCCCGGAGACGGGGACTGA
Sequence3       ------------------------------------------------------------
```

Fig. 3.1. (**a**) Multiple unaligned sequences in the FASTA format. (**b**) Multiple alignment in the FASTA format. (**c**) Multiple alignment in the CLUSTAL format.

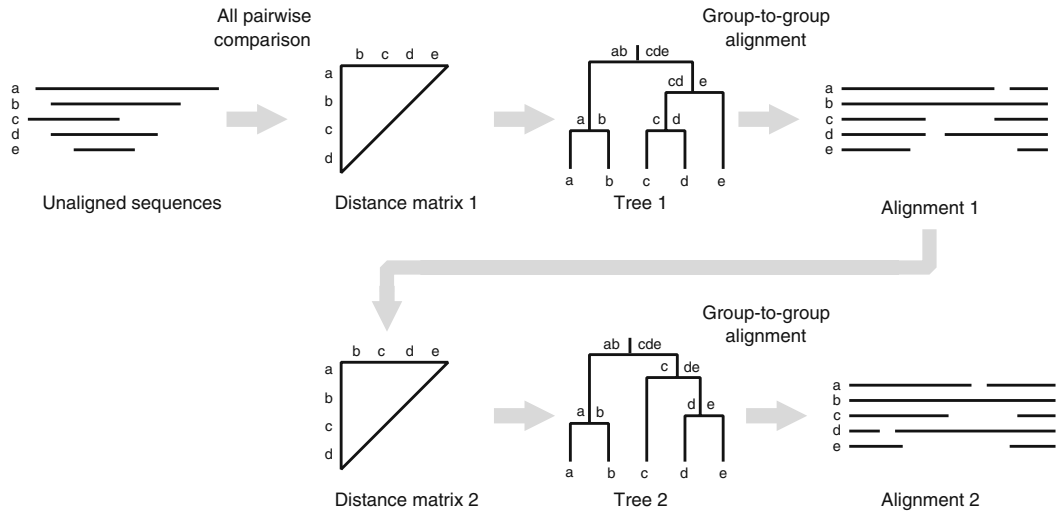

Fig. 3.2. Calculation procedure of the progressive method.

The progressive method has a drawback in that once a gap is incorrectly introduced, especially at an early step (near a leaf of the guide tree), the gap is never removed in later steps. To overcome this drawback, there are two types of solutions: the iterative refinement method and the consistency-based method. These two procedures are quite different; the former tries to correct mistakes in the initial alignment, whereas the latter tries to avoid mistakes in advance. However, both work well to improve alignment accuracy.

1.2.2. Iterative Refinement Method

In the iterative refinement method, an objective function that represents the "goodness" of the MSA is explicitly defined. An initial MSA, calculated by the progressive or another method, is subjected to an iterative process and is gradually modified so that the objective function is maximized, as shown in **Fig. 3.3**. Various

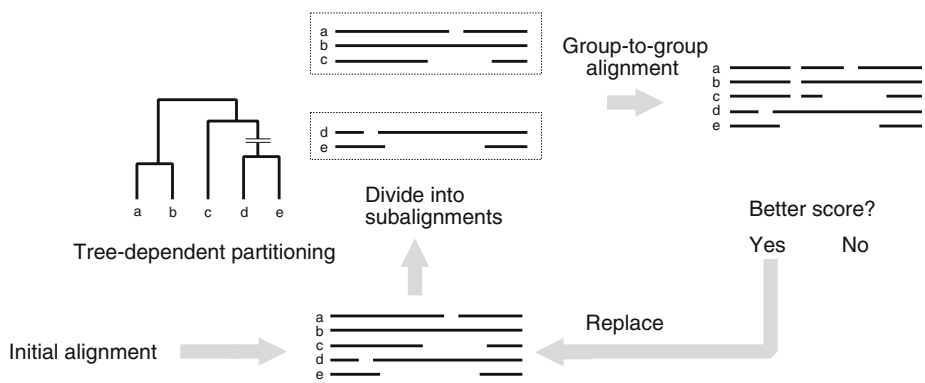

Fig. 3.3. Calculation procedure of the iterative refinement method.

combinations of objective functions and optimization strategies have been proposed to date *(18–22)*. Among them, Gotoh's iterative refinement method, PRRN with the weighted sum-of-pairs (WSP) objective function *(23–25)*, is the most successful one, and it forms the basis of recent iterative refinement methods including the FFT-NS-i option of MAFFT. In the iterative refinement method, an MSA is partitioned into two groups, and the two groups are re-aligned using a group-to-group alignment algorithm. This process is repeated until no more improvements are made. To save computation time, the partitions of the MSA are restricted to those corresponding to the branches of the guide tree *(26)* (*see* **Note 1**).

1.2.3. Consistency-Based Iterative Refinement Method

Consistency-based method is an entirely different approach to overcome the drawback of the progressive method. In consistency criteria, the accuracy of an MSA is judged with regard to how it is consistent with pairwise alignments. Several types of consistency criteria were described previously *(27–29)*, but TCoffee *(30)* achieved a great improvement in accuracy (*see* **Note 3**). MAFFT adopted a consistency criterion similar to COFFEE *(29)* and combined it with the WSP objective function. The combined objective score is iteratively maximized in the G-INS-i, L-INS-i, and E-INS-i options *(4)*. The three options should be separately used according to whether the homology is expected to be global or local (*see* **Note 2** and **Section 2.3**).

1.3. Tools for Genome Alignment

Along with the availability of genome sequences has come the need to construct genomic alignments. Since genomes undergo various evolutionary events such as inversions, translocations, and duplications, the original order and orientation of the nucleotides is no longer maintained. This makes the alignment problem harder, because genome alignment tools need to detect the homologous pieces as well as perform a nucleotide alignment of them, instead of performing one global alignment of the whole input. Since some homologous regions might not appear in all input genomes, output alignments may relate a subset of the input sequences.

Determining the homologous pieces is complicated by the presence of paralogs and segmental duplications, which lead to an increase in the number of possible combinations when many genomes are involved. There are different paradigms followed by the alignment tools in order to handle duplicated bases. **Figure 3.4** (*i*) illustrates a toy example with three genomes (A, B, and C, depicted horizontally) and two genes (1 and 2). Genome A contains two copies of gene 1 (A1 and A1′, both orthologous to B1 and C1), and genome B contains two copies

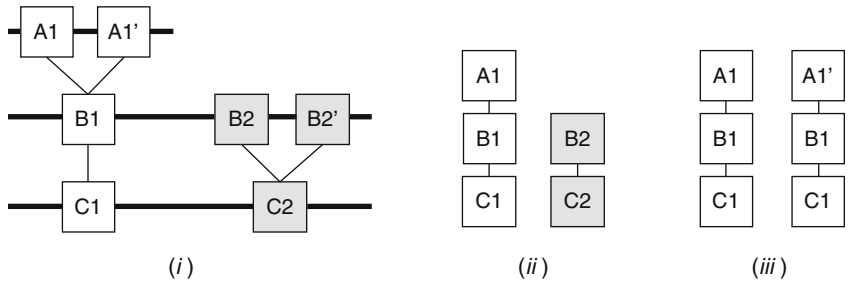

Fig. 3.4. An example of three genomic sequences with tamdemly duplicated genes.

of gene 2 (B2 and B2′, both orthologous to C2). The ideal output would include an alignment containing A1, A1′, B1, and C1, and another alignment containing B2, B2′, and C2. Traditional multiple alignment output does not allow a given species to appear more than once in the same alignment; for example, B2 and B2′ cannot appear in the same alignment. The relationship between B2, B2′, and C2 could be represented by two separate alignments (one between B2 and C2, and another between B2′ and C2), in which case the sequence of C2 would appear twice in the output, in two different alignments. What happens in practice is that most multiple alignment tools allow each base from each genome to participate in at most one alignment, and therefore provide a one-to-one mapping between the bases. An example of such an output is shown in **Fig. 3.4** (*ii*), where each gene can only appear once, and two genes from the same genome cannot appear in the same alignment; thus, one of the two copies of each duplicated gene (in this case A1′ and B2′) remains unaligned.

Other alignment tools perform a multiple alignment with respect to a reference genome: the bases of the reference genome must appear exactly once, but the bases of the other genomes are allowed to appear in more than one alignment. Alignments that do not contain the reference genome are not constructed. **Figure 3.4** (*iii*) shows an example of a multiple alignment which uses A as a reference genome; there is exactly one alignment for each of the A1 and A1′ genes and they both contain B1 and C1. In that case, gene 2 is not aligned since it is not present in A. An alignment with respect to a reference genome *G* is suitable for studies that need to address *G*-centric questions, such as "How conserved is a given base of genome *G* " or "How much of genome *G* is covered by each species," etc.

Performing a nucleotide alignment at the whole-genome level can be a resource-intensive process; the analyses that can be performed on a single desktop computer are limited by the size of the sequences involved. The alignment problem is easier for prokaryotes, which have relatively short genomes and are

less repetitive, and is more complex for eukaryotes as the input is longer and more repetitive. Most modern programs can handle (on a single computer) up to ~100 genomes that have up to ~10 M bases each, though the actual complexity depends on the similarity of the genomes and is hard to estimate. Very large alignment problems require either parallelization on a computer cluster, or some other non-trivial setup; however, for any available large genome, an alignment between that genome and related genomes may already exist in some Internet resource.

In **Section 2.6**, we explain how to use two genome alignment tools: TBA and MAUVE. Other alignment tools that are simply too slow for modern alignment problems, that are able to provide genomic alignments for only two genomes (such as BLAST and all its variants), or that require the sequences to be free from rearrangements are not covered here.

2. Program Usage

MAFFT accepts a set of unaligned sequences in FASTA format (**Fig. 3.1a**) and returns an alignment of the sequences in FASTA (**Fig. 3.1b**) or CLUSTAL (**Fig. 3.1c**) format. The following methods are implemented: progressive method (FFT-NS-1 and FFT-NS-2), iterative refinement method (FFT-NS-i), and consistency-based iterative refinement method (G-INS-i, L-INS-i, and E-INS-i). MAFFT is a command-line program with no graphical user interface. It runs on a UNIX-like environment, such as Linux, Terminal on Mac OS X, and Cygwin on Windows. For users who prefer a graphical interface, web-based interfaces are available at several MAFFT servers (**Table 3.1**), but we do not explain here how to use them.

Table 3.1
MAFFT servers

Kyushu University	http://align.bmr.kyushu-u.ac.jp/mafft/online/server/
EBI	http://www.ebi.ac.uk/mafft/
GenomeNet	http://align.genome.jp/mafft/
Max Planck Institute	http://toolkit.tuebingen.mpg.de/mafft
MyHits, SiB	http://myhits.isb- sib.ch/cgi-bin/mafft

MAFFT accepts the following command line arguments that control MSA strategy (See the MAFFT page for the full list of arguments).

- --**retree** ***n*** Progressive alignment is repeated ***n*** times renewing the guide tree. When $n = 0$, only a guide tree is calculated and no alignment is computed. $n = 2$ by default.
- --**maxiterate** ***n*** The number of cycles of iterative refinement. $n = 0$ by default.
- --**6merpair** Distance between every pair of sequences is calculated based on the number of shared 6mers *(3, 31, 32)*. Enabled by default.
- --**globalpair** All pairwise alignments are computed with the Needleman–Wunsch *(12)* algorithm. Disabled by default.
- --**localpair** All pairwise alignments are computed with the Smith–Waterman *(13)* algorithm. Disabled by default.
- --**genafpair** All pairwise alignments are computed with a local alignment algorithm with the generalized affine gap cost *(33)*. Disabled by default.

Various MSA strategies can be executed by combining the above arguments. Users have to select appropriate strategies according to the nature of input sequences, considering the trade-off between accuracy and computational cost.

As the easiest way, there is a fully automated option that selects an appropriate strategy from L-INS-i, FFT-NS-i, and FFT-NS-2, depending on the size of the data.

```
% mafft --auto input_file > output_file
```

However, we strongly recommend seeing how each strategy works and trying some different ones, particularly when a set of distantly related sequences is being aligned. We list several typical cases with possible difficulties and show what option of MAFFT can be helpful. We also show command-line arguments.

2.1. A Large Number (> ~1000) of Sequences – FFT-NS-2, FFT-NS-1, or PartTree

Only the progressive method can align many (> ~1000) sequences in a reasonable amount of time. As the distance calculation process is the time-limiting factor in this case, progressive methods with fast distance calculation are suitable. An approximate distance between a pair of sequences can be roughly estimated by the number of shared k-mers between them *(3, 31, 32)*.

```
% mafft input_file > output_file
```

This command performs the two-cycle progressive method (FFT-NS-2) shown in **Fig. 3.2**. For faster computation, try the one-cycle progressive method (FFT-NS-1)

```
% mafft --retree 1 input_file > output_file
```

which omits the re-estimation of tree, and thus is approximately two times faster than the two-cycle progressive method.

When handling larger numbers of sequences (N>∼10,000), FFT-NS-1 can be still too slow. For that case, MAFFT has a faster and more scalable option, PartTree *(17)*, which relies on an approximate guide tree construction method with a time complexity of $O(N \log N)$. For two-cycle progressive alignment,

```
% mafft --parttree input_file > output_file
```

and for one-cycle progressive alignment

```
% mafft --parttree --retree 1 input_file > output_file
```

Note that these methods are less accurate by 2 or 3% than the corresponding methods with a full guide tree.

2.2. Long Sequences (> ∼10,000 nt.) – FFT-NS-2 or FFT-NS-i

When handling long sequences, space complexity should be considered. FFT-NS-2 and FFT-NS-i automatically switch the DP algorithm to a memory saving one *(34)* requiring only an $O(L)$ RAM space when the length of the alignment exceeds a threshold, where L is sequence length. The FFT-based alignment algorithm *(3)* plays an important role to reduce the calculation time in cases of highly conserved and long sequences. This algorithm is also enabled by default. Thus no special argument is needed. For the progressive method,

```
% mafft input_file > output_file
```

and for the iterative refinement method,

```
% mafft --maxiterate 10 input_file > output_file
```

The maximum number of the iterative refinement cycle is limited to 10 in this command.

2.3. Low Similarity – E-INS-i, L-INS-i, or G-INS-i

First of all, check whether the set of sequences is globally alignable or locally alignable: will the resulting alignment be like that in **Fig. 3.5a** (needs long internal and terminal gaps), **b** (needs long terminal gaps), or **c** (globally alignable)? In general, we recommend

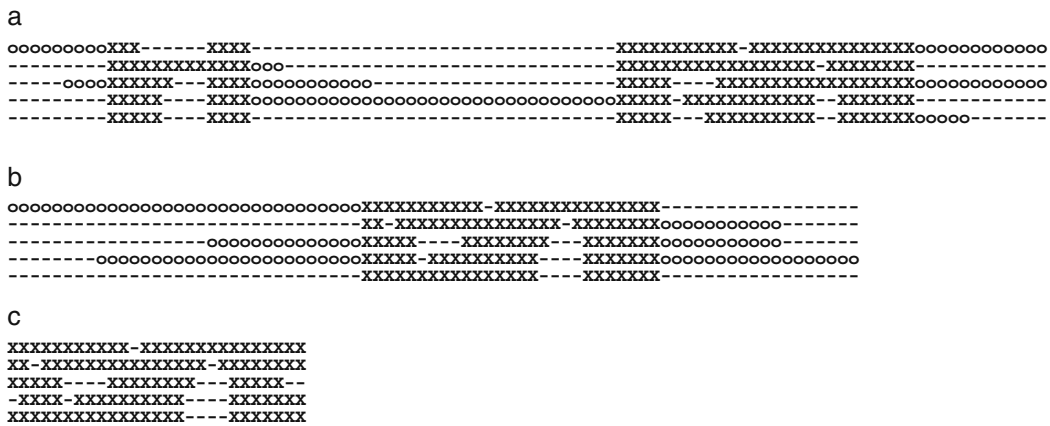

Fig. 3.5. Globally alignable, locally alignable, or long internal gaps? "-" represents a gap, "X" represents an aligned nucleotide and "o" is an un-alignable nucleotide.

assuming case **a** if the type of alignment is unclear, because the assumption of case **a** is the safest of the three. Once an initial alignment is obtained, then trimming the alignment to include only the relevant homologous parts can be done manually, and then the methods designed for case **c** can be applied. In addition, consider the tradeoff between computational costs and accuracy: highly accurate methods tend to be time- and space-consuming. The three methods (G-INS-i, L-INS-i, and E-INS-i) explained here can process only $< \sim 500$ sequences on a standard desktop computer.

Case a – E-INS-i When long internal gaps are expected. This case corresponds to an SSU rRNA alignment composed of distantly related organisms, such as from the three different domains (Eukarya, Bacteria, and Archaea), or a cDNA alignment with splicing variants, both of which need long gaps to be inserted. E-INS-i is suitable for such data. It employs a generalized affine gap cost *(33)* in the pairwise alignment stage, in which the un-alignable regions are left unaligned.

```
% mafft --genafpair --maxiterate 1000 input_file >
  output_file
```

An alias is available:

```
% mafft-einsi input_file > output_file
```

Case b – L-INS-i Locally alignable. When only one alignable block surrounded by long terminal gaps is expected, L-INS-i is recommended.

```
% mafft --localpair --maxiterate 1000 input_file >
  output_file
```

or

```
% mafft-linsi input_file> output_file
```

Case c – G-INS-i Globally alignable. If the entire region of input sequences is expected to be aligned, we recommend G-INS-i that assumes global homology.

```
% mafft --globalpair --maxiterate 1000 input_file>
  output_file
```

or

```
% mafft-ginsi input_file> output_file
```

To obtain a high-quality MSA from the biological point of view, we recommend trying multiple independent methods (*see* **Note 5**), different parameter sets (*see* **Note 4**), and comparing various alignments by eye (*see* **Note 6**).

2.4. Adding New Sequence(s) to an Existing Alignment

In this section, we explain how to align a group of aligned sequences with another group of aligned sequences or with unaligned sequences. The tools explained here can be useful for a semi-automatic alignment or for combining "eye-ball" alignments and automated alignments, although they are not needed for fully automated sequence analyses.

2.4.1. Profile–Profile Alignment

An alignment between two alignments or between an alignment and a single sequence is basically performed by converting each alignment/sequence to a profile *(35)* and then applying a pairwise profile–profile alignment algorithm. Many packages including MAFFT offer this utility.

```
% mafft-profile aligned_group1 aligned_group2> output_file
```

where ***aligned_group1*** and ***aligned_group2*** are alignment files in the FASTA format. Each of them is converted to a profile, and then an alignment between ***aligned_group1*** and ***aligned_group2*** is calculated. Note that this method assumes that the phylogenetic relationship is like that in **Fig. 3.6a** or **3.6b**.

2.4.2. Constrained Alignment

We sometimes encounter a situation like that in **Fig. 3.6c**, in which an in-group sequence is to be added. A simple profile–profile alignment should not be applied to such a case, because one may overlook a close relatedness between the new sequence and a sequence (marked with an asterisk in **Fig. 3.6c**) in the existing alignment. Instead, we recommend another option:

```
% mafft-linsi --seed aligned_sequences new_sequence> output_file
```

The alignment within ***aligned_sequences*** is fixed, and an MSA consisting of both the ***aligned_sequences*** and ***new_sequence*** is created. This can be used even if many new sequences are to be added into an existing "seed" alignment, as shown in **Fig. 3.6d**.

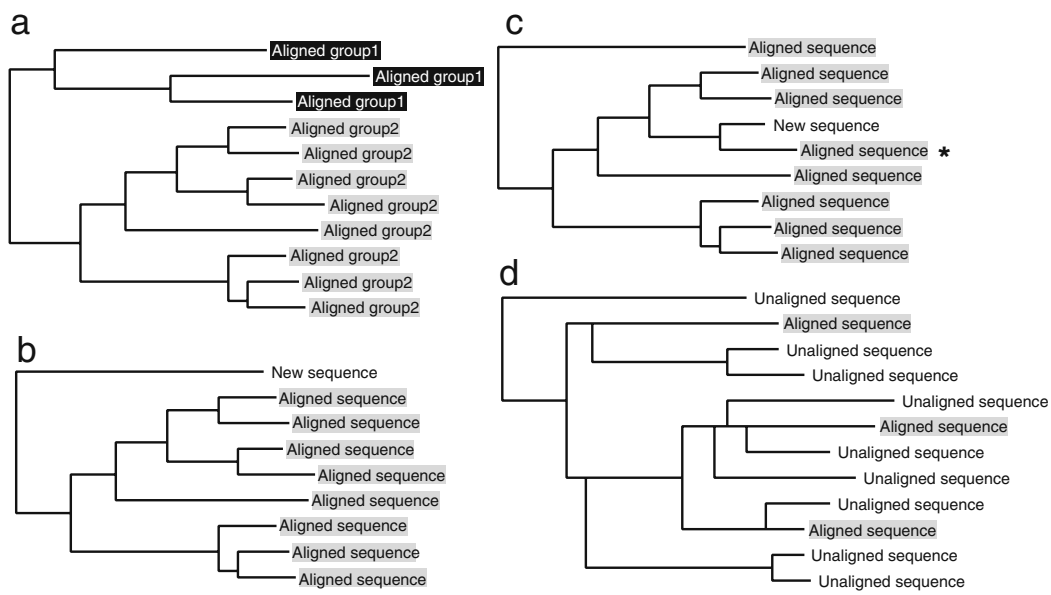

Fig. 3.6. Possible relationships between a group of aligned sequences and new sequence(s). (**a**) and (**b**) can be subjected to a profile–profile alignment, but (**c**) and (**d**) should not.

```
% mafft-linsi --seed aligned_sequences unaligned_
  sequences \ > output_file
```

Furthermore, two or more fixed groups are acceptable.

```
% mafft-linsi --seed group1 --seed group2 --seed
  group3\ unaligned_sequences> output_file
```

Users can select an appropriate algorithm, such as L-INS-i, E-INS-i, and G-INS-i, according to the characteristics of the sequences (global similarity or local similarity; *see* **Section 2.3**).

2.5. Outputting a Guide Tree

The guide tree is output by the --**treeout** argument.

```
% mafft --retree 0 --treeout --globalpair input_file>
  output_file
```

This command creates the ***input_file*.tree** file, which shows the guide tree based on pairwise alignments. The --**retree 0** option suppresses the progressive alignment calculation. Either of --**localpair**, --**globalpair** or --**genafpair** argument is recommended when the number of sequences is less than ~500.

For larger dataset (number of sequences > ~500), an approximate method based on 6mer counting is recommended, keeping in mind that the resulting tree is more approximate than that generated by the above methods.

```
% mafft --retree 0 --treeout --6merpair input_file>
  output_file
```

If the --**retree 0** option is omitted, the second guide tree (Tree 2 in **Fig. 3.2**) is output.

```
% mafft --treeout input_file> output_file
```

This tree is generally expected to be more accurate than the initial guide tree (Tree 1) as explained in **Section 1.2**. However, this is not the case when evolutionary unrelated sequences are included. This is because the second guide tree is built based on an MSA of unrelated sequences and such MSA may contain erroneous information, since the basic assumption that all of the sequences in the alignment are homologous is violated (*see* **Section 1.1**). In such a case, the use of the --**retree 0** argument is recommended. *See* **Section 3.3** for detailed discussion.

2.6. Genome Alignment

If the input sequences have translocations, inversions, or duplications, MAFFT is not suitable; MSA tools specially designed for such data should be used instead. In this section we focus on two genome alignment tools: TBA and MAUVE.

TBA *(7, 8)* is an MSA tool that can also be used to align whole genomes, though this is tractable on a single computer only for genomes that are at most a few million bases long. The latest revision (available on the web) can detect all types of genomic rearrangements. It outputs alignment blocks, where each block can contain any subset of the input species. An alignment block

cannot contain a species more than once and, by default, any genomic base of the input can appear in at most one alignment block in the output.

The first step in utilizing TBA is to align all pairs of genomes with BLASTZ *(36)*, a BLAST-like pairwise alignment tool. The TBA software package provides a wrapper program that calls BLASTZ for all pairs and post-processes the results to resolve overlapping alignments. The resulting pairwise alignments are used as input to the TBA program, along with a user-supplied binary tree of the genomes that dictates the order in which progressive alignment shall be performed. The algorithm works recursively: for each node of the tree, a set of alignment blocks corresponding to the multiple alignment of the genomes of the left subtree, and a set of alignment blocks corresponding to the multiple alignment of the genomes of the right subtree are recursively obtained. The two sets are then combined using pairwise alignments between genomes of the left subtree and genomes of the right subtree as guides (this step is called MULTIZ). The result is a set of alignment blocks corresponding to the multiple alignment of all the genomes under that node. Once the algorithm reaches the root, the final output is returned in a single file using the MAF format *(37)* (**Fig. 3.7**).

TBA can also be used to perform a multiple alignment with respect to a reference genome. The UCSC Genome Browser *(38)* utilizes MULTIZ to create and display reference-based whole genome alignments, such as a human-centric alignment of various vertebrates and a *D. melanogaster*-centric alignment of various insects. All such whole-genome MULTIZ alignments can be viewed in graphic and text form and downloaded directly from the UCSC Genome Browser website.

Both the TBA and BLASTZ packages can be downloaded from the CCGB website. To make an alignment with TBA, the input sequences should be in the FASTA format, and each species should be placed in a file of its own, named after the species (with no filename extension). The FASTA header of each sequence needs to have the following format:

>***species_name***:***chromosome_name***:***start_position***:
orientation: ***chr_size***

```
a score=4622.0
s crescentus  277993 61 + 4016947 GCAAAACGTCGCTATAAGGCGCCATGTCAAACCAAGAAAAAGCAATCGCCGTCGTCGAACG
s k31         365323 61 + 5876911 GCGGAACGCTGATATAGGGCGCCATGTCAAACCAAGAAAAAGCAATCGCCGCTGTCGAGCG

a score=7121.0
s crescentus  278054 53 + 4016947 GCTCAATGACGAATACGAACGCGCCGTCGGCGCTCTGCGCGACGCGCTCCGCG
s maris      1082062 53 - 3364850 GCTCAATGACCGCTTTGCCCGCGCCATCGACACGCTGCTGGCCCCGCTGCACG
s k31         365384 53 + 5876911 GCTAAACCAGGAATACGAACGCGCCGTCGACGCCCTGCGGACCGCGCTTCGCG

a score=2608.0
s crescentus  278107 47 + 4016947 CCTATCTCGAAGATGGAACCCGCCCGGATCCGGCCGCACGCTTTGAC
s k31         365437 47 + 5876911 CCTATCTCGAACATGGAACCCGGCCCGATCCGCAGACGCGGTTCGAC
```

Fig. 3.7. An example of alignment blocks generated by TBA from three bacterial genomes.

In this format, ***species_name*** and ***chromosome_name*** are the names of the species and the chromosome to which the sequence corresponds, and ***chr_size*** is the original size of that chromosome. In case the sequence is a piece of a chromosome (for example, if a smaller region has been extracted from a larger chromosome in order to make an alignment of a specific locus), ***start_position*** and ***orientation*** refer to the beginning position (1-indexed, inclusive) and the orientation (the ASCII character + or −) of the piece (otherwise they should be given as **1** and +, respectively). If a species contains only one sequence in its respective FASTA file, then its header can be simplified to just the species name; otherwise, if a species consists of many sequences, the special header format must be used.

To speed up BLASTZ, and also to remove spurious pairwise alignments from the BLASTZ output, low-complexity regions as well as repeats should be *softmasked* (that is, converted to lowercase characters, with the rest of the genome in uppercase) in the input sequences. This can be achieved by running RepeatMasker *(39)* and TRF *(40)* on the genomes. Repeats cover almost 50% of the human genome, so this step is essential for repeat-rich sequences. Bacterial genomes are less prone to this problem as they are not so repetitive, but it is still a good practice to mask any low complexity regions they might contain.

With the input sequences softmasked and in their proper form, the following two commands should be used to make an alignment with default parameters:

```
% all_bz "(guide tree)"
% tba "(guide tree)" *.maf output_file
```

The "(***guide tree***)"is a parenthesized expression that describes a binary tree of the species, such as "**(((human chimp) (mouse rat)) chicken)**"; it also serves as a list of filenames, since each genome is expected to reside in a file named after its name. The generated ***output_file*** will contain the resulting alignments in MAF format (**Fig. 3.7**). These can be visualized with Gmaj *(41)*, an interactive viewer (written in Java) for MAF files.

To make an alignment with respect to a reference genome, **F**=***genome*** should be added to the all bz call, and **E**=***genome*** **P**=**multic** to the tba call (where ***genome*** is the name of the genome used as reference):

```
% all_bz F=genome"(guide tree)"
% tba E=genome P=multic "(guide tree)" *.maf output_file
```

The all_bz call can also take an extra argument: the name of a file that describes the parameters of the underlying BLASTZ calls for each pair of genomes. By specifying such a file, different parameters can be used for each genomic pair, allowing, for example, more sensitive parameters for distant pairs. More documentation on this feature is available on the web *(42)*.

MAUVE *(9, 10)* is a multiple genome alignment and visualization tool available for various platforms. It does not require an external BLAST-like program, as it employs its own method for finding matching words (seeds) between the genomes, and extending them into larger anchors. Collinear anchors between genomes are combined into locally collinear blocks (LCBs). A multiple global alignment is then performed for each significant LCB, and the results are output to a file using the XMFA format *(43)* and visualized in-place with a bundled Java application. Following the traditional paradigm, MAUVE aligns each genomic base at most once in the output. Although the initially published version *(9)* detected genomic rearrangements, it also required all of the species to be present in an alignment block; a newer version called Progressive MAUVE (available on the web) allows for the generation of alignment blocks of any subset of species.

MAUVE can be easily run through its graphical front-end. Unless the sequences compared are very closely related, the progressive version of the algorithm should be used in order to achieve greater sensitivity and relax the constraint requiring the presence of *all* species in a block. The progressive alignment dialog box can be used to select the FASTA sequences to be aligned, and also to tweak the alignment parameters. The *Match Seed Weight* refers to the word size (more precisely, the number of matching positions in the seed) used to look for anchors. If the program cannot find any anchors with the default parameters due to low sequence identity, and assuming

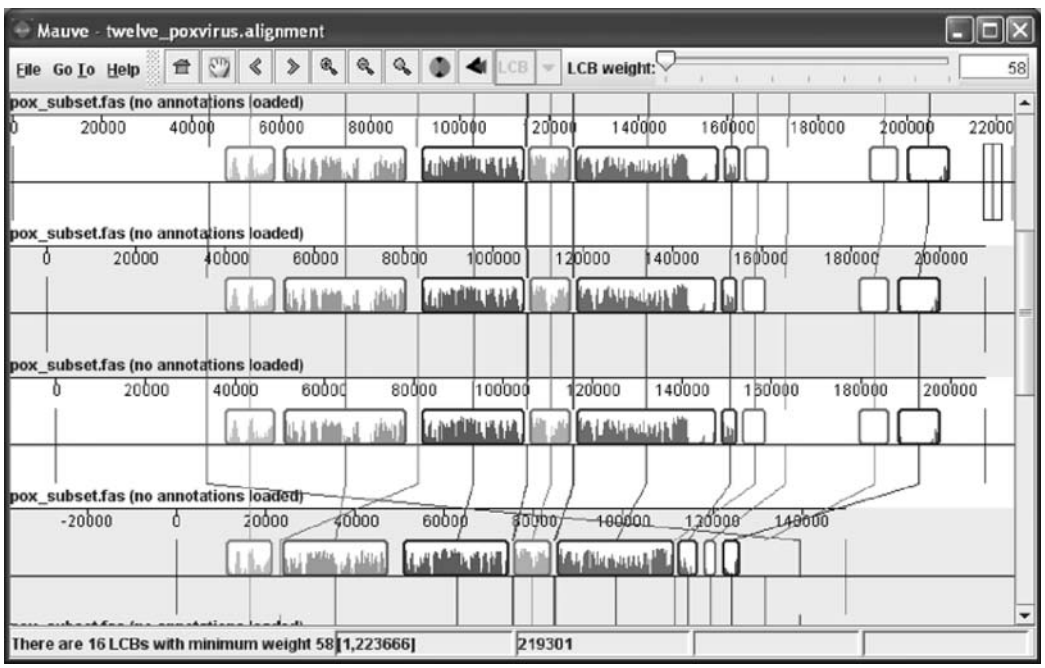

Fig. 3.8. A screenshot of MAUVE.

the genomes are small enough and not repetitive, the seed weight can be gradually lowered to permit a more sensitive anchoring. The *Minimum LCB Weight* refers to the minimum required pairwise score of an LCB. The default value (which is calculated on the fly and reported in the console output) may need to be manually increased for very closely related (nearly identical) species, or decreased for distant genomes with a lot of rearrangements and fragmented assemblies. Once the alignment is done, it will be automatically visualized (**Fig. 3.8**). The Java application can be also used to visualize previous alignments by loading the respective XMFA file.

3. Examples

3.1. Constructing an MSA of 17 Eukaryotic LSU rRNA Sequences

Here we show how to use MAFFT to construct an MSA of LSU rRNA sequences as an example. The dataset is downloadable from http://align.bmr.kyushu-u.ac.jp/mafft/examples/lsu_euk. The lsu_euk file contains 17 LSU rRNA sequences from various Eukaryotes, including animals, fungi, plants, and protists, with sequence lengths of ~3500 (*Tetrahymena*) to ~5000 (human) residues. The sequence file was subjected to the FFT-NS-2 option of MAFFT. A part of the resulting alignment is shown in **Fig. 3.9**. Most of other options, such as FFT-NS-i and L-INS-i, are applicable to this dataset, but the difference in alignment quality is small in such a highly conserved dataset. The CPU times of FFT-NS-2 and L-INS-i were ~9 s and ~2 min, respectively, on a 3 GHz Intel Xeon processor with Mac OS X. A rough phylogenetic tree of the LSU rRNA sequences by the NJ method is shown in **Fig. 3.10**.

3.2. Adding a Sequence to an Existing MSA

Assume that a partial LSU rRNA sequence (1439 residues) of chimpanzee has been newly determined. The chimpanzee sequence is at http://align.bmr.kyushu-u.ac.jp/mafft/examples/lsu_chimp. We explain here how to add the chimpanzee sequence into the existing alignment of 17 Eukaryotes. Some consideration is needed if the new sequence is expected to be closely related to one of existing sequences, like human sequence in this case (**Fig. 3.10**).

The **mafft-profile** program, which performs a group-to-sequence alignment or a group-to-group alignment, is very fast (requires less than 1 s) but not appropriate for this case. This is because **mafft-profile** converts the alignment consisting of 17 sequences into a profile ignoring the close relationship between chimpanzee and human sequences. **Figure 3.9b** shows an example of misalignment introduced by an incorrect use of **mafft-profile**. To avoid such misalignment, either of the following two procedures is recommended.

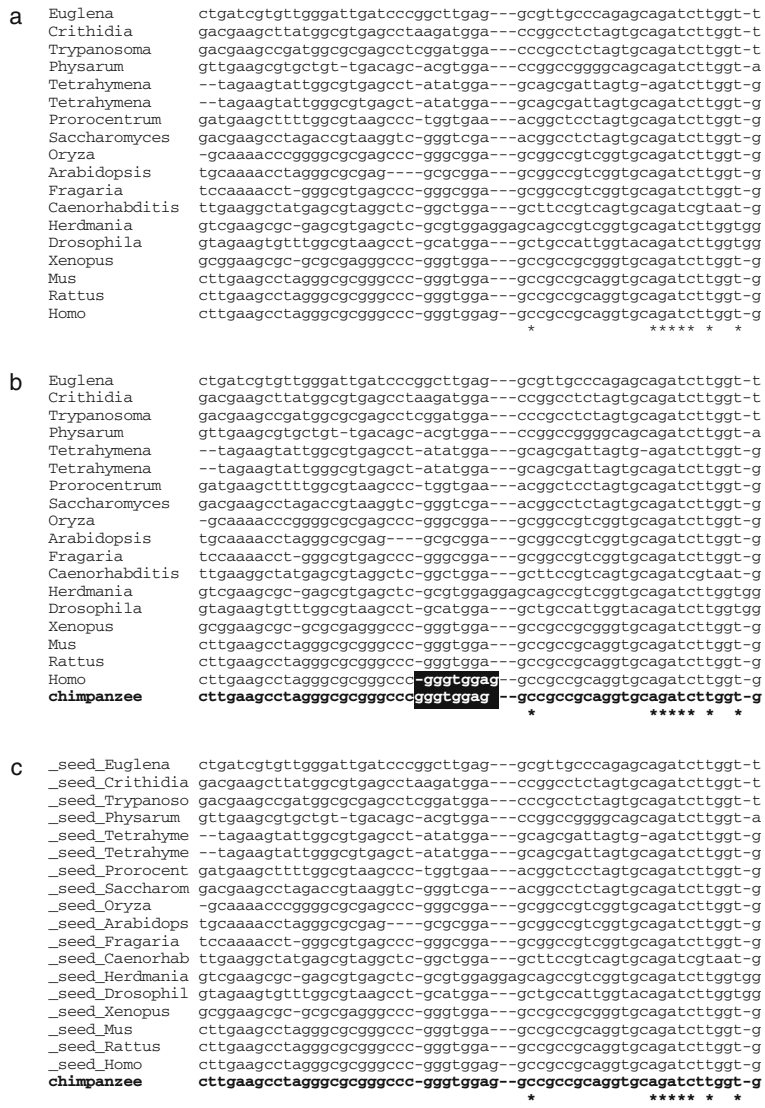

Fig. 3.9. Alignments of eukaryotic LSU rRNA sequences. (**a**), a part of the FFT-NS-2 alignment consisting of 17sequences; (**b**), an example of misalignment (highlighted) when adding the chimpanzee sequence (in bold letters) with an incorrect use of mafft-profile; (**c**), the chimpanzee sequence was added by the **--seed** option.

- If the original alignment was fully automatically constructed, remove all of the gaps from the 17 sequences and then apply the same program to the 17 (original) + 1 (chimpanzee) dataset to entirely re-construct an alignment consisting of the 18 sequences. The CPU times of FFT-NS-2 and L-INS-i are ~10 s and ~2 min, respectively, for this example.
- If the original alignment has already been manually inspected, the entire re-construction is not practical. Instead, the constrained alignment, explained in **Section 2.4**, can be useful. **Figure 3.9c** shows a resulting alignment in which a

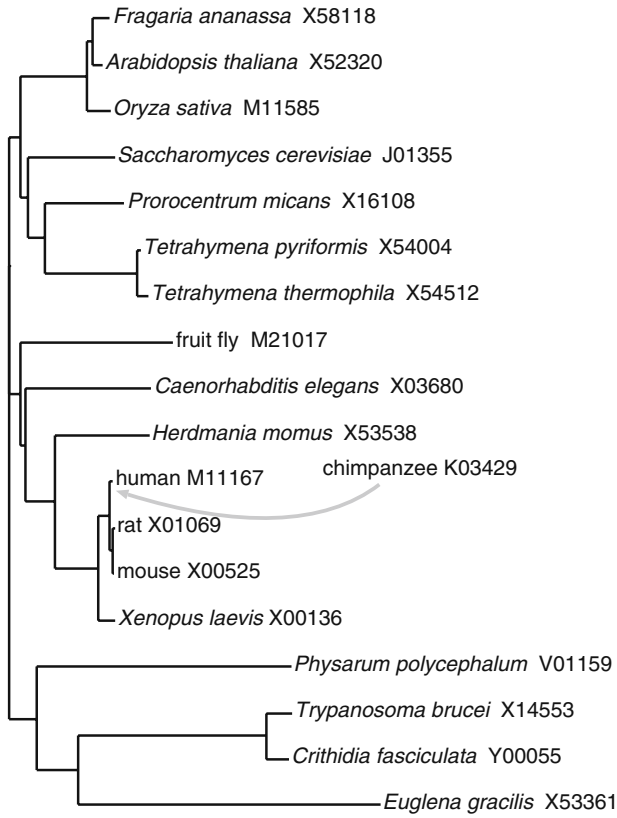

Fig. 3.10. Eukaryotic LSU rRNA sequences used in the example. Genbank accession numbers are shown after each species name. Note that this is a rough NJ tree just to show what organisms are included and is expected to reflect the phylogenetic relationship to some extent, but not completely.

chimpanzee sequence was added into the 17-species alignment by applying the constrained alignment option (**mafft --seed**). The FFT-NS-2 option was applied using the original alignment of the 17 sequences as a "seed." CPU time was ~17 s. The L-INS-i option can also be applied and it took ~1 min.

3.3. Determining Which Sequences Should be Included in an MSA

The third example is related to a problem as to which sequences should be included in an MSA and which sequences should not. We use here an artifactual example of heterogeneous data consisting of 32 sequences, including unrelated ones, SSU rRNA, tRNA, and randomly selected cDNA sequences. The dataset is available at http://align.bmr.kyushu-u.ac.jp/mafft/examples/unrelated. The SSU rRNA sequences used here (~1500 to ~2000 residues in length) were taken from all three domains, Eukarya, Bacteria, and Archaea, and they have sequence similarity. The tRNA sequences (~70 residues in length) have very weak sequence similarity to each

other and no homology to the SSU rRNA sequences. The rest were randomly taken from mouse cDNA sequences with sequence lengths of ~100 to ~700 residues. How can we graphically display the relatedness, or correctly extract the two homologous groups (SSU rRNA and tRNA)?

The most general criterion to determine whether a sequence should be included into an MSA is *E*-value calculated by BLAST *(44)* or other search tools. The sequences showing lower *E*-values to a query than a given threshold (for example, $E \leq 10^{-5}$) can be presumed as homologs and then be included into an MSA. However, such criterion is not relevant when we need an MSA of distantly related homologs, such as tRNA sequences. In this example, the *E*-values between tRNA sequences are 1 or larger and it cannot discriminate tRNA sequences from non-homologous sequences as far as standard similarity search tools are used.

In such a case, a guide tree calculated at the initial step of MSA is sometimes useful. A guide tree can show the relationship among many sequences at a glance, regardless of whether the sequences are evolutionarily related. It can be used to roughly classify (unrelated and related) sequences or to identify closely related subsets. **Figure 3.11a** shows a guide tree used in the initial step of the G-INS-i method. It was calculated with the **--treeout --globalpair --retree 0** arguments. CPU time was ~11 s. Two clusters, SSU rRNA and tRNA, are successfully discriminated. More approximate guide tree based on the 6mer counting method (**--treeout --6merpair --retree 0**) is shown in **Fig. 3.11b**. This method is very fast (requires CPU time of ~0.4 s for this example) and thus has a merit in processing larger number (~1000 to ~10,000) of sequences at the cost of accuracy.

Note that these two procedures do not include the calculation of MSA of unrelated sequences, and thus are applicable to a dataset containing evolutionarily unrelated sequences. On the other hand, the second guide tree in the progressive method (Tree 2 in **Fig. 3.2**) is not suitable for the present case with unrelated sequences because the resulting tree is built based on an MSA of unrelated sequences. Therefore, when the input sequences include evolutionary unrelated sequences, the **--tree out** option should be used always with the **--retree 0** option. **Figure 3.11c** shows the second guide tree calculated with an incorrect use of the **--treeout** option.

The guide trees explained in this section reflect the phylogenetic relationship to some extent, but it should be noted that their detailed branching order is not reliable at all. This is partly because the compared regions are not identical for pairs; almost an entire region is compared for a pair of closely related sequences, whereas only highly conserved regions are compared for a pair of distantly related sequences. Such a comparison can be useful at the early stage of a comparative study for roughly classifying sequences, but the guide tree should not be interpreted as an evolutionary tree. Only the

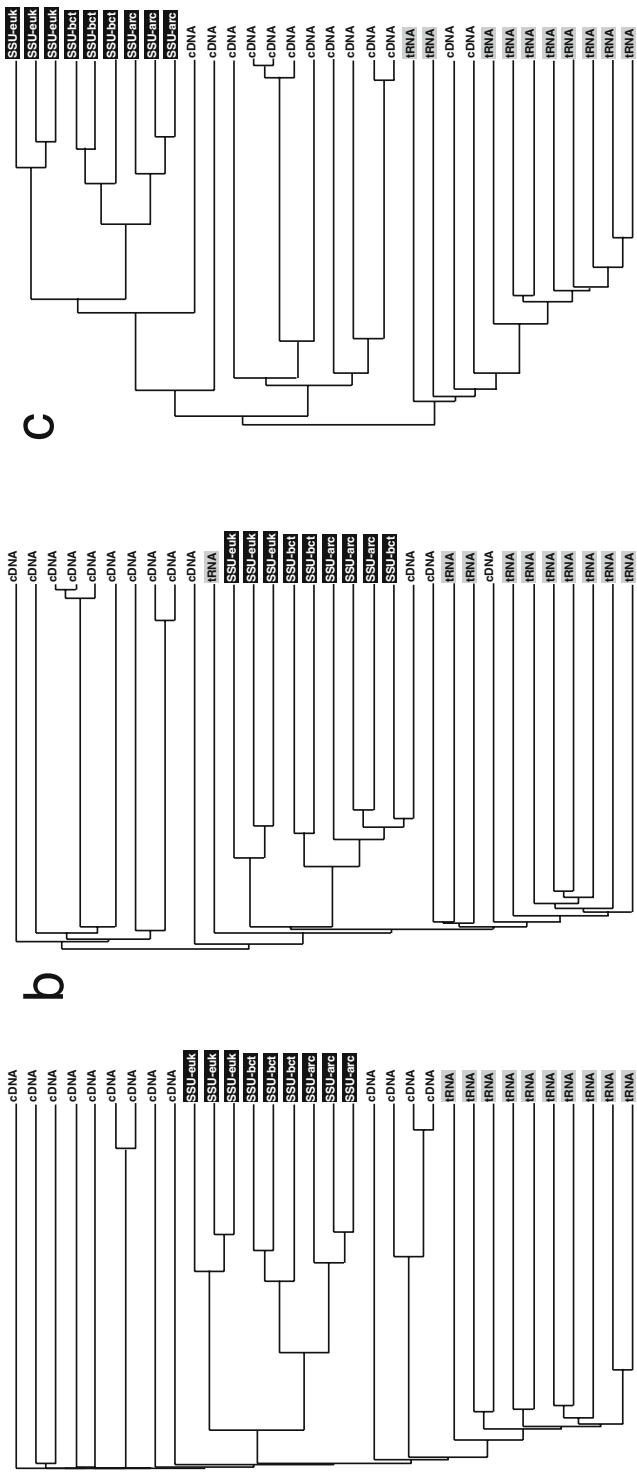

Fig. 3.11. Three types of guide trees calculated from an artifactual dataset containing unrelated sequences. SSU rRNAs are highlighted in black, tRNAs are shadowed and the rest are randomly selected cDNA sequences. (**a**), a tree based on pairwise Smith-Waterman alignments (**--globalpair**; recommended); (**b**), a tree based on 6mer distances (**--6 merpair --retree 0**; recommended only for a large dataset); (**c**), the second guide tree (**--6 merpair**; not recommended).

members of a distinct cluster, like each of the SSU and tRNA clusters in **Fig. 3.11a** and **b**, can be subjected to the MSA methods explained in **Sections 2.1, 2.2, and 2.3**. Once an MSA of homologous sequences is obtained, the MSA has much more relevant phylogenetic information than the guide tree or pairwise alignments, and the MSA can be subjected to phylogenetic inferences and/or other analyses.

4. Notes

1. Looking at **Fig. 3.2**, users who want to use an MSA for phylogeny inference may think this procedure is somewhat problematic, because guide trees, which can be thought of as an approximate distance-based phylogenetic tree, are used before an MSA is created. Will the final phylogenetic tree be affected by the initial guide tree? In our opinion, this problem makes little bias on the phylogenetic inference, as long as only unambiguously aligned sites are subjected to the phylogenetic analysis. One way to determine unambiguously aligned sites is as follows: (1) create two or more MSAs by independently applying different tools to the same set of unaligned sequences, (2) compare the MSAs using a tool such as ALTAVIST *(45)* and Mumsa *(46)*, and then (3) extract the regions of agreement. The resulting sites are more likely to be correctly aligned than those generated by an MSA with a single tool. This procedure improves the specificity of the alignment, but of course, at the cost of sensitivity. Iterative refinement methods also use a guide tree as shown in **Fig. 3.3**, and thus have the same problem discussed earlier: the alignment and the final phylogenetic tree might be affected by the guide tree. The doubly nested iterative refinement method of PRRN *(25)* is another possible solution to this problem.
2. Some MSA methods assume that all of the input sequences are globally alignable; that is, the overall sequences (5′ to 3′) are assumed to be homologous, but this assumption does not necessarily agree with real analyses. To overcome this problem, some other MSA methods incorporate a local alignment algorithm *(4, 30, 47)* to allow large successive gaps at both termini. Local alignment methods avoid the assumption of global homology by identifying only short patches of strong sequence similarity. Some methods maintain the assumption of global homology but impose different penalties for terminal gaps in order to allow for some degree of local alignment character *(16, 48–50)*.
3. TCoffee *(30)* and ProbCons *(49)* use consistency information as follows. This process is called "library extension" in TCoffee. Given three sequences, A, B, and C, all of the pairwise alignments

A-B, B-C, and A-C are calculated. The collection of these alignments is called a "primary library." An alignment between A-B can also be calculated by synthesizing A-C and B-C, and it may differ from an alignment generated by a direct comparison between A and B. This process is carried out for all possible combinations of the three sequences from the input data. The resulting collection of pairwise alignments is called an "extended library." The next step is progressive alignment, in which a group-to-group alignment is performed using a DP matrix whose elements are calculated as the summation of the scores of pairwise alignments in the library that has the matching corresponding to the element. As a result, in a group-to-group alignment step, two (groups of) sequences, say A and B, are aligned while taking into account the information of other sequences, say C.

4. Most alignment methods have several parameters, including a scoring matrix that describes the relative frequency of each substitution type and gap penalties that are related to the frequency of insertion/deletion events. The parameters, especially the gap penalties, greatly affect the alignment quality. The default parameters of most MSA methods are tuned by trial-and-error, but some methods systematically determine the parameters from unaligned sequences in an unsupervised manner *(49)*. As for MAFFT, we recommend, if possible, trying out several sets of entirely different gap penalties, such as

   ```
   % mafft-linsi --op 1.53 --ep 0.123 input_file> output_file
   ```

   (equivalent to default of mafft-linsi)

   ```
   % mafft-linsi --op 2.4 --ep 0.0 --fmodel input_file> output_file
   % mafft-ginsi --op 2.4 --ep 0.5 input_file> output_file
   ```

5. A large number of MSA programs have been developed as listed in **Table 3.2**. It is highly recommended to use multiple different methods to obtain a high quality alignment *(11*, *51*, *52)*. A decision tree for selecting appropriate tools (**Fig. 3.5** in ref. *53)* in various situations may be useful. The alignments generated by different methods can be combined into a single MSA by meta-aligners, such as MCoffee *(51)* in the TCoffee package. A combined MSA is generally more reliable than each MSA calculated by a stand-alone program *(51)*. In addition, Mumsa *(46)* and similar tools use multiple independent MSAs to improve the specificity (*see* **Note 1**).

6. There are also many visualization tools (**Table 3.3**) that allow interactive alignment editing and manual identification of reliably aligned regions. For alignments of sequences with

low-similarity, manual inspection is extremely important. Morrison *(54)* addressed various situations in which the mathematically optimal alignment is not a biologically correct alignment.

Table 3.2
MSA programs

ClustalW *(16)*	ftp://ftp.ebi.ac.uk/pub/software/clustalw2/
PRRN *(25)*	http://www.genome.ist.i.kyoto-u.ac.jp/~aln_user
DIALIGN *(45)*	http://bibiserv.techfak.uni- bielefeld.de/dialign/
TCoffee *(30)*	http://www.tcoffee.org/
MAFFT *(3)*	http://align.bmr.kyushu- u.ac.jp/mafft/software/
MUSCLE *(48)*	http://www.drive5.com/muscle/
ProbConsRNA *(49)*	http://probcons.stanford.edu/
Kalign *(50)*	http://msa.cgb.ki.se/
ProbAlign *(55)*	http://www.cs.njit.edu/usman/probalign/
PRIME *(56)*	http://prime.cbrc.jp/
MLAGAN *(57)*	http://lagan.stanford.edu/lagan_web/index.shtml
MAVID *(58)*	http://baboon.math.berkeley.edu/mavid/
MUMmer [59]	http://mummer.sourceforge.net/
CCGB (TBA and BLASTZ)	http://www.bx.psu.edu/miller_lab
MAUVE *(9)*	http://gel.ahabs.wisc.edu/mauve/

Table 3.3
Alignment viewers

Jalview	http://www.jalview.org/
Kalignvu	http://msa.cgb.ki.se/
ClustalX	ftp://ftp.ebi.ac.uk/pub/software/clustalw2
UCSC	http://genome.ucsc.edu/
Gmaj	http://globin.cse.psu.edu/dist/gmaj/

Acknowledgments

We thank Chuong B. Do for critical reading of the manuscript.

References

1. Woese, C. R., and Fox, G. E. (1977) Phylogenetic structure of the prokaryotic domain: the primary kingdoms. *Proc Natl Acad Sci USA* **74**, 5088–90.
2. Flicek, P., Keibler, E., Hu, P., Korf, I., and Brent, M. R. (2003) Leveraging the mouse genome for gene prediction in human: from whole-genome shotgun reads to a global synteny map. *Genome Res* **13**, 46–54.
3. Katoh, K., Misawa, K., Kuma, K., and Miyata, T. (2002) MAFFT: a novel method for rapid multiple sequence alignment based on fast Fourier transform. *Nucleic Acids Res* **30**, 3059–66.
4. Katoh, K., Kuma, K., Toh, H., and Miyata, T. (2005) MAFFT version 5: improvement in accuracy of multiple sequence alignment. *Nucleic Acids Res* **33**, 511–8.
5. Wilm, A., Mainz, I., and Steger, G. (2006) An enhanced RNA alignment benchmark for sequence alignment programs. *Algorithms Mol Biol* **1**, 19.
6. Carroll, H., Beckstead, W., O'connor, T., Ebbert, M., Clement, M., Snell, Q., and McClellan, D. (2007) DNA reference alignment benchmarks based on tertiary structure of encoded proteins. *Bioinformatics* **23**, 2648–49.
7. Blanchette, M., Kent, W. J., Riemer, C., Elnitski, L., Smit, A. F., Roskin, K. M., Baertsch, R., Rosenbloom, K., Clawson, H., Green, E. D., Haussler, D., and Miller, W. (2004) Aligning multiple genomic sequences with the threaded blockset aligner. *Genome Res* **14**, 708–15.
8. http://www.bx.psu.edu/miller_lab
9. Darling, A. C., Mau, B., Blattner, F. R., and Perna, N. T. (2004) Mauve: multiple alignment of conserved genomic sequence with rearrangements. *Genome Res* **14**, 1394–403.
10. http://gel.ahabs.wisc.edu/mauve/
11. Edgar, R. C., and Batzoglou, S. (2006) Multiple sequence alignment. *Curr Opin Struct Biol* **16**, 368–73.
12. Needleman, S. B., and Wunsch, C. D. (1970) A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J Mol Biol* **48**, 443–53.
13. Smith, T. F., and Waterman, M. S. (1981) Identification of common molecular subsequences. *J Mol Biol* **147**, 195–7.
14. Gotoh, O. (1982) An improved algorithm for matching biological sequences. *J Mol Biol* **162**, 705–8.
15. Feng, D. F., and Doolittle, R. F. (1987) Progressive sequence alignment as a prerequisite to correct phylogenetic trees. *J Mol Evol* **25**, 351–60.
16. Thompson, J. D., Higgins, D. G., and Gibson, T. J. (1994) 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**, 4673–80.
17. Katoh, K., and Toh, H. (2007) Parttree: an algorithm to build an approximate tree from a large number of unaligned sequences. *Bioinformatics* **23**, 372–4.
18. Barton, G. J., and Sternberg, M. J. (1987) A strategy for the rapid multiple alignment of protein sequences. confidence levels from tertiary structure comparisons. *J Mol Biol* **198**, 327–37.
19. Berger, M. P., and Munson, P. J. (1991) A novel randomized iterative strategy for aligning multiple protein sequences. *Comput Appl Biosci* **7**, 479–84.
20. Gotoh, O. (1993) Optimal alignment between groups of sequences and its application to multiple sequence alignment. *Comput Appl Biosci* **9**, 361–70.
21. Ishikawa, M., Toya, T., Hoshida, M., Nitta, K., Ogiwara, A., and Kanehisa, M. (1993) Multiple sequence alignment by parallel simulated annealing. *Comput Appl Biosci* **9**, 267–73.
22. Notredame, C., and Higgins, D. G. (1996) Saga: sequence alignment by genetic algorithm. *Nucleic Acids Res* **24**, 1515–24.
23. Gotoh, O. (1994) Further improvement in methods of group-to-group sequence alignment with generalized profile operations. *Comput Appl Biosci* **10**, 379–87.
24. Gotoh, O. (1995) A weighting system and algorithm for aligning many

phylogenetically related sequences. *Comput Appl Biosci* **11**, 543–51.

25. Gotoh, O. (1996) Significant improvement in accuracy of multiple protein sequence alignments by iterative refinement as assessed by reference to structural alignments. *J Mol Biol* **264**, 823–38.
26. Hirosawa, M., Totoki, Y., Hoshida, M., and Ishikawa, M. (1995) Comprehensive study on iterative algorithms of multiple sequence alignment. *Comput Appl Biosci* **11**, 13–18.
27. Vingron, M., and Argos, P. (1989) A fast and sensitive multiple sequence alignment algorithm. *Comput Appl Biosci* **5**, 115–21.
28. Gotoh, O. (1990) Consistency of optimal sequence alignments. *Bull Math Biol* **52**, 509–25.
29. Notredame, C., Holm, L., and Higgins, D. G. (1998) COFFEE: an objective function for multiple sequence alignments. *Bioinformatics* **14**, 407–22.
30. Notredame, C., Higgins, D. G., and Heringa, J. (2000) T-coffee: a novel method for fast and accurate multiple sequence alignment. *J Mol Biol* **302**, 205–17.
31. Higgins, D. G., and Sharp, P. M. (1988) CLUSTAL: a package for performing multiple sequence alignment on a microcomputer. *Gene* **73**, 237–44.
32. Jones, D. T., Taylor, W. R., and Thornton, J. M. (1992) The rapid generation of mutation data matrices from protein sequences. *Comput Appl Biosci* **8**, 275–82.
33. Altschul, S. F. (1998) Generalized affine gap costs for protein sequence alignment. *Proteins* **32**, 88–96.
34. Myers, E. W., and Miller, W. (1988) Optimal alignments in linear space. *Comput Appl Biosci* **4**, 11–17.
35. Gribskov, M., McLachlan, A. D., and Eisenberg, D. (1987) Profile analysis: detection of distantly related proteins. *Proc Natl Acad Sci USA*, **84**, 4355–58.
36. Schwartz, S., Kent, W. J., Smit, A., Zhang, Z., Baertsch, R., Hardison, R. C., Haussler, D., and Miller, W. (2003) Human-mouse alignments with BLASTZ. *Genome Res* **13**, 103–7.
37. http://genome.ucsc.edu/FAQ/FAQformat
38. http://genome.ucsc.edu/
39. Smit, A. F. A., Hubley, R., and Green, P. Repeatmasker. http://www.repeatmasker.org/
40. Benson, G. (1999) Tandem repeats finder: a program to analyze DNA sequences. *Nucleic Acids Res* **27**, 573–80.
41. http://globin.cse.psu.edu/dist/gmaj/
42. http://www.bx.psu.edu/miller_lab/dist/tba_howto.pdf
43. http://gel.ahabs.wisc.edu/mauve/mauve-user- guide/
44. Altschul, S. F., Madden, T. L., Schaffer, A. A., Zhang, J., Zhang, Z., Miller, W., and Lipman, D. J. (1997) Gapped BLAST and PSI-BLAST: a new generation of protein database search pro grams. *Nucleic Acids Res* **25**, 3389–402.
45. Morgenstern, B., Goel, S., Sczyrba, A., and Dress, A. (2003) Altavist: comparing alternative multiple sequence alignments. *Bioinformatics* **19**, 425–6.
46. Lassmann, T., and Sonnhammer, E. L. (2007) Automatic extraction of reliable regions from multiple sequence alignments. *BMC Bioinformat* **8** Suppl 5, S9.
47. Morgenstern, B., Dress, A., and Werner, T. (1996) Multiple DNA and protein sequence alignment based on segment-to-segment comparison. *Proc Natl Acad Sci USA* **93**, 12098–103.
48. Edgar, R. C. (2004) MUSCLE: multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Res* **32**, 1792–7.
49. Do, C. B., Mahabhashyam, M. S., Brudno, M., and Batzoglou, S. (2005) ProbCons: probabilistic consistency-based multiple sequence alignment. *Genome Res* **15**, 330–40.
50. Lassmann, T., and Sonnhammer, E. L. (2005) Kalign – an accurate and fast multiple sequence alignment algorithm. *BMC Bioinformat* **6**, 298.
51. Wallace, I. M., O'Sullivan, O., Higgins, D. G., and Notredame, C. (2006) M-Coffee: combining multiple sequence alignment methods with t-coffee. *Nucleic Acids Res* **34**, 1692–9.
52. Golubchik, T., Wise, M. J., Easteal, S., and Jermiin, L. S. (2007) Mind the gaps: Evidence of bias in estimates of multiple sequence alignments. *Mol Biol Evol* **24**, 2433–42.
53. Do, C. B., and Katoh, K. (2008) Protein multiple sequence alignment Functional Proteomics, *Methods Mol Biol* **484**, 379–413.
54. Morrison, D. (2006) Multiple sequence alignment for phylogenetic purposes. *Aust Syst Bot* **19**, 479–539.

55. Roshan, U., and Livesay, D. R. (2006) Probalign: multiple sequence alignment using partition function posterior probabilities. *Bioinformatics* **22**, 2715–21.
56. Yamada, S., Gotoh, O., and Yamana, H. (2006) Improvement in accuracy of multiple sequence alignment using novel group-to-group sequence alignment algorithm with piecewise linear gap cost. *BMC Bioinformat* **7**, 524.
57. Brudno, M., Do, C. B., Cooper, G. M., Kim, M. F., Davydov, E., Green, E. D., Sidow, A., and Batzoglou, S. (2003) LAGAN and multi-LAGAN: efficient tools for large-scale multiple alignment of genomic DNA. *Genome Res* **13**, 721–31.
58. Bray, N., and Pachter, L. (2004) MAVID: constrained ancestral alignment of multiple sequences. *Genome Res.* **14**, 693–9.