Chapter 3
Multiple alignments for structural, functional, or phylogenetic analyses of homologous sequences

L. Duret
Laboratorie de Biométrie, Génétique et Biologie des Populations, Université Claude Bernard, France.

S. Abdeddaim
Département d'Informatique de Rouen, Universite de Rouen, France.

1 Introduction

Understanding the structure, function and evolution of genes is one of the main goals of genome sequencing projects. Classically, gene function has been investigated experimentally through the analysis of mutant phenotypes. More recently, comparative analysis of homologous sequences has proved to be a very efficient approach to study gene function (this approach has been coined 'comparative genomics' or 'phylogenomics'). Indeed, the evolution of living organisms may be considered as an ongoing large-scale mutagenesis experiment. For more than three billion years, genomes have continuously undergone mutations (substitutions, insertion, deletions, recombination, and so on). Deleterious mutations are generally rapidly eliminated by natural selection, while mutations that have no phenotypic effect (neutral mutations) may, by random genetic drift, eventually become fixed in the population. Globally, advantageous mutations are very rare, and hence residues that are poorly conserved during evolution generally correspond to regions that are weakly constrained by selection (1). Thus, studying mutation patterns through the analysis of homologous sequences is useful not only to study evolutionary relationships between sequences, but also to identify structural or functional constraints on sequences (DNA, RNA, or protein).

The alignment of homologous sequences consists of trying to place residues (nucleotides or amino acids) in columns that derive from a common ancestral residue. This is achieved by introducing gaps (which represent insertions or deletions) into sequences. Thus, an alignment is a hypothetical model of mutations (substitutions, insertions, and deletions) that occurred during sequence evolution. The best alignment will be the one that represents the most likely evolutionary scenario. Generally, this best alignment cannot be unambiguously established. Firstly, because of the computational complexity of this problem, alignment algorithms that are usable in practice cannot guarantee to find the best solution (see *Section 4.1*). Secondly, even with an ideal algorithm, finding the best alignment would not be guaranteed because current knowledge of the probability of occurrence of the different types of mutation is still limited (see *Section 2.3*). However, as long as homologous sequences are not too divergent, fast approximate algorithms may be used to provide reliable alignments. In practice, such alignments are commonly used in molecular or evolutionary biology. Typical examples of usage of multiple alignments are indicated in *Table 1*.

Table 1 Examples of usage of multiple alignments

• Identification of functionally important sites Multiple alignments allow the identification of highly conserved residues are likely to correspond to essential sites for the structure or function of the sequence and may thus be useful to design mutagenesis experiments.
• Demonstration of homology between sequences (see Section 2.1)
• Molecular phylogeny Molecular phylogenetic trees rely on multiple alignments (protein or DNA) to infer mutation events from which it is possible to retrace evolutionary relationships between sequences. Such trees are useful to reconstruct the history of species or multigenic families, and notably to identify gene duplication events to distinguish orthologues from paralogues. It is important to note that unreliable parts of alignments should not be used to build phylogenetic trees since they do not reflect the real pattern of mutations that occurred during evolution and may lead to artifactual results.
• Search for weak but significant similarities in sequence databases The sensitivity of sequence similarity search may be improved by weighting sites according to their degree of conservation. Thus, multiple alignments of homologous sequences are used by methods such as profile searches (see the chapter by Henikoff in this volume) or PSI-BLAST (24) to identify distantly related members of a family.
• Structure prediction The use of multiple alignments increases significantly the efficiency of protein secondary structure prediction. Moreover, the identification of covariant sites (or compensatory mutations) in alignments (protein or RNA) is a strong argument to suggest that these sites interact in the molecule *in vivo*. Finally, alignments are commonly used for homology modeling, i.e. for the structure prediction of sequences by comparison with homologues of known structure.
• Function prediction The three-dimensional (3D) structure of homologous proteins or RNA is often much more conserved than their primary sequence. Similar shape usually implies similar function. Thus, if a new gene is found to be homologous to an already characterized gene it is possible to infer the likely function of the new gene from the known one. Such inferences should however be used with great caution.
• Design of primers for PCR (polymerase chain reaction) identification of related genes

The general procedure to compute a multiple alignment of homologous sequences consists of three steps:

(a) Search for homologues in sequence databases.

(b) Compute alignments.

(c) Check and edit alignments.

In this chapter, we will focus, essentially, on steps (b) and (c). Firstly, we will define some general concepts underlying multiple alignment methodology. We will then describe and compare different methods that have been developed to align sequences. As far as possible, we will indicate WWW sites where these tools are available, so that they may be used from any computer with an appropriate WWW browser software and internet connection. The list of WWW sites that we provide here is also available at the following address:

http://pbil.univ-lyon1.fr/alignment.html

Some problems such as contig assembly, related to multiple alignment will not be treated in this chapter. We will only describe methods intended for the alignment of homologous sequences and, in particular, we will not deal with the problem of finding common motifs in a set of unrelated sequences. Note that there is not an absolute difference between motif search and multiple alignments: when homologous sequences have diverged too much there may remain only a few short conserved fragments, separated by regions of variable length. Motif-based methods have been developed to identify and align such conserved fragments within highly divergent sequences. In *Section* 4.4 we will mention some of these methods. However, for a more exhaustive review on this topic, see Chapter 7 by Jonassen in this volume.

2 Basic concepts for multiple sequence alignment

2.1 Homology: definition and demonstration

Two sequences are said to be homologous if they derive from a common ancestor. Generally, homology is inferred by sequence similarity. It should be stressed, however, that similarity does not necessarily reflect homology: similarity between short sequence fragments may result from evolutionary convergence (2), or may simply occur by chance. Moreover many sequences contain relatively long fragments of very biased nucleotide or amino acid composition (e.g. CA-repeats in DNA, proline-rich domains in proteins) (3). Generally, similarities between such 'low complexity regions' do not reflect evolutionary relationship. However, in the absence of such compositional bias, similarity over an extended region usually implies homology. Statistical tests can be used to evaluate the chances that an observed similarity occurred purely by chance and thus accept or reject the hypothesis of homology (4). Such tests are now generally provided by similarity search programs.

Multiple alignments may be useful to help demonstrate homology: a weak similarity which would be considered as non-significant in a pairwise sequence comparison may prove to be highly significant if the same residues are conserved in other distantly related sequences. It should be emphasized that if sequences have diverged too much, homology may not be recognizable on the basis of sequence similarity alone.

2.2 Global or local alignments

In the above paragraph, we implicitly considered sequences that are homologous over their entire length. However, in many cases, homology is restricted to a limited region of the sequences. Indeed, many proteins consist of a combination of discrete 'modules' that have been shuffled during evolution. It is clear that many protein-coding genes result from recombination between different fragments of other genes. This modular evolution has played a major role in protein evolution and has been particularly facilitated in eukaryotes thanks to the presence of introns within genes (5).

Multiple copies of a given module may be repeated within a sequence, and a set of modules may occur at different relative positions in different genes. In such cases, it is not possible to align sequences over their whole length (global alignment) and it is thus necessary to perform alignments only on homologous modules (local alignment) See *Figure 1* for an illustration.

2.3 Substitution matrices, weighting of gaps

As indicated earlier, searching for the best alignment consists of searching for the one that represents the most likely evolutionary scenario. Thus, the prob-

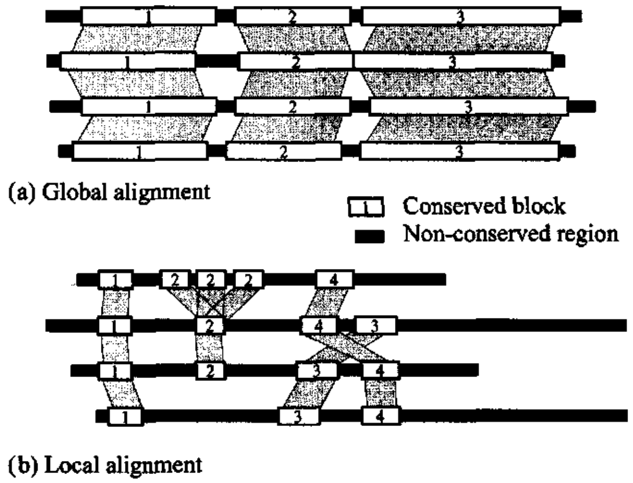


Figure 1 Global versus local alignment. (a) Conserved regions occur in the same order in all sequences. They can be represented in a single global alignment. (b) Some conserved regions are duplicated or occur in a different order along sequences. It is necessary to perform local alignments to display similarities between all conserved regions.

ability of occurrence of the different mutational events during evolution must be taken into consideration when computing a multiple alignment. In alignments, three types of mutations are considered: substitutions, insertions or deletions (the two latter events are often indistinguishable, and are commonly referred as 'indels').

2.3.1 Substitutions

The probability of substitution of one amino acid by another depends on the structure of the genetic code (i.e. on the number of mutations necessary to pass from one codon to another) and also on the phenotypic effect of that mutation. Substitutions of one amino acid by another with similar biochemical properties generally do not greatly affect the structure and hence the function of the protein. Thus, during evolution, such conservative substitutions are relatively frequent compared to other substitutions. It is important to note that the probability of substitution of one amino acid by another depends on the evolutionary distance between sequences. At short evolutionary distances, probabilities of substitution mainly reflect the structure of the genetic code, whereas at larger distances, probabilities of substitution depend essentially on biochemical similarities between amino acids. Various methods have been proposed to build series of matrices that give estimates of probabilities of all possible substitutions for different evolutionary distances (6–8). The most commonly used are the PAM and BLOSUM substitution matrices. PAM matrices suitable for increasing evolutionary distances are indicated by increasing indices (e.g. PAM80, PAM120, and PAM250). The opposite convention has been used for the BLOSUM series (e.g. BLOSUM80 for short evolutionary distances, BLOSUM45 for large evolutionary distances). Generally, alignment programs allow users to choose which substitution matrix to use. In the CLUSTAL W program (9) (see *Section 4.2*) substitution matrices are automatically selected and varied at different alignment stages according to the divergence of the sequences to be aligned.

Probabilities of substitutions also vary along sequences according to the local environment of amino acids in the folded protein. Thus, several environment-specific substitution matrices have been developed (e.g. for α-helix, or β-sheet) (10). However, to our knowledge, these matrices are rarely used for multiple alignments.

At the DNA level, probabilities of substitution vary according to the bases. Notably, transitions (substitutions between two purines—A, G—or two pyrimidines—C, T) are generally more frequent than transversions (substitutions between a purine and a pyrimidine). Thus, multiple alignment programs generally propose a parameter to weight more heavily transversions than transitions. Probabilities of nucleotide substitution also depend on neighbouring bases (e.g. in vertebrates, C in CG dinucleotides is hypermutable) (11, 12). However, currently available alignment programs do not make use of such information.

2.3.2 Insertions, deletions

The probability of occurrence of an indel depends on its length. Thus, when computing an alignment, penalties (p) associated with gaps are often estimated using a linear or 'affine' model such as:

$$p = a + bL$$

where L is the length of the gap, a the gap opening penalty, and b the gap extension penalty. However, analyses of alignment of homologous sequences have shown, both for protein and nucleic sequences, that this model underestimates the probability of long indels (7, 13, 14). Indeed, more realistic indel penalties can be estimated with models of the following form:

$$p = a + b\log(L)$$

However, because of computational complexity, such models have not been implemented in commonly used alignment programs. Fortunately, other approaches have been proposed to align sequences with large indels (see *Section 4.3*).

The probability of occurrence of indels in proteins also depend on the degree of divergence between sequences (7, 13). Thus, as for amino-acids substitution matrices, indel penalty parameters should ideally be varied according to the divergence of the sequences to be aligned. The probability also depends on the nature of the sequences: protein, structural RNA, non-coding DNA (in which transposable elements may be inserted), etc. Moreover, probabilities of indel may vary along sequences. In proteins notably, indels are more frequent within external loops than in the core of the structure. Thus, knowledge on the structure of proteins can be used to weight indels. For example, the CLUSTAL W program uses residue specific indel penalties and locally reduced indel penalties to encourage new gaps in potential loop regions rather than in regular secondary structure. In cases where secondary structure information is available, indel-penalty masks can also be used to guide the alignment.

It is important to note that, in most programs, default parameters for gap penalties have been set for typical globular proteins. These may not be optimal for other sequences.

3 Searching for homologous sequences

The first step in the analysis of a family of homologous sequences consists of searching for all available members of that family. Published sequences are stored in databases: GenBank (15) or EMBL (16) for nucleic acid sequences and SWISSPROT-TREMBL (17) or PIR (18) for protein sequences. Retrieval systems such as Entrez (19), SRS (20), or ACNUC (21) have been developed to query those databases and extract sequences according to the associated annotation (e.g. keywords, taxon, authors). Some WWW addresses for commonly used database retrieval systems are shown in *Table 2*. Unfortunately, it is not possible to rely on the annotation to identify in a database all homologous sequences belonging to a given family. Presently, the most efficient way to identify those homologues

Table 2 Websites for text-based searches in sequence databases

Entrez at NCBI	http://www.ncbi.nlm.nih.gov/Entrez/
SRS at EBI	http://srs.ebi.ac.uk/
WWW-QUERY at PBIL	http://pbil.univ-lyon1.fr/
ExPASy	http://www.expasy.ch/sprot/
DBGET at GenomeNet	http://www.genome.ad.jp/

Table 3 Websites for sequence similarity searches in databases

BLAST at NCBI[a]	http://www.ncbi.nlm.nih.gov/BLAST/
WU-BLAST at EMBL[b]	http://dove.embl-heidelberg.de/Blast2e/
FASTA at EBI	http://www2.ebi.ac.uk/fasta3/
Smith-Waterman search at EBI	http://www2.ebi.ac.uk/bic_sw/
BCM search launcher	http://gc.bcm.tmc.edu:8088/search-launcher.html
BLAST at PBIL[c]	http://pbil.univ-lyon1.fr/BLAST/blast.html

[a] Possibility to select BLAST output results by taxa.

[b] Performs multiple alignment on homologous sequences detected by BLAST.

[c] Possibility to select BLAST output results by taxa or keyword.

consists in taking one member of the family and comparing it to the entire database with a similarity search program such as FASTA (22) or BLAST (23, 24). To guarantee a more exhaustive search, one may repeat this procedure with several distantly related homologues identified in the first step. See the review by Altschul, *et al.* (25) for a comprehensive discussion of sequence similarity searches.

The sensitivity of a sequence similarity search may be improved by weighting sites according to their degree of conservation. Thus, once several homologous sequences have been identified, it is possible to use methods such as profile searches (see Chapter 5 in this volume) or PSI-BLAST (24) that rely on a multiple alignment to identify more distantly related members of the family. A list of some similarity search WWW servers is presented *Table 3*.

4 Multiple alignment methods

Once homologous sequences have been identified, which program should be preferentially used to align them? Several multiple alignment methods (algorithms) have been developed, but none of them is ideal. Thus, it is important to have an idea of what these algorithms try to solve, in order to make an informed choice of the most appropriate method(s) for a particular problem. The multiple alignment problem is algorithmically hard: methods that guarantee to find the best alignment (for a given measure of alignment score and for a given set of substitution matrix and gap penalty parameters) require so much time and space (memory) that they cannot be used in practice with, say, more than 10 to 15 sequences of length 100. Thus, alternative algorithms have been developed using heuristics to gain speed and limit space requirements. Although these heuristics do not guarantee to find the optimal alignment, they are very useful in practice and often give results very close to the exact solution. In the following we will focus on four families of multiple alignment algorithms:

(a) Algorithms that guarantee to find the optimal alignment for a given scoring scheme; these algorithms can be used only for a limited number of short sequences.

(b) Heuristic algorithms that are based on a progressive pairwise alignment approach.

(c) Heuristic algorithms that build a global alignment based on local alignments.

(d) Heuristic algorithms that build local multiple alignments.

It should be noted that this list is not exhaustive. Other multiple alignment methods such as those based on hidden Markov models (26) or genetic algorithms

Table 4 Websites for multiple alignments

• **Optimal global multiple alignment**	
MSA at IBC	http://www.ibc.wustl.edu/ibc/msa.html
• **Progressive global multiple alignment**	
ClustalW at EBI[a]	http://www2.ebi.ac.uk/clustalw/
ClustalW, Multalin at PBIL[b]	http://pbil.univ-lyon1.fr/
MAP, ClustalW at BCM	http://kiwi.imgen.bcm.tmc.edu:8088/search-launcher/launcher.html
Multalin at INRA[b]	http://www.toulouse.inra.fr/multalin.html
ClustalW at Pasteur[c]	http://bioweb.pasteur.fr/seqanal/alignment/intro-uk.html
ClustalW at DDBJ	http://www.ddbj.nig.ac.jp/searches-e.html
MAP	http://genome.cs.mtu.edu/map.html
• **Block-based global multiple alignment**	
DCA at BiBiServ	http://bibiserv.techfak.uni-bielefeld.de/dca/
DIALIGN2 at BiBiServ	http://bibiserv.TechFak.Uni-Bielefeld.DE/dialign/
DCA at Pasteur[c]	http://bioweb.pasteur.fr/seqanal/alignment/intro-uk.html
DIALIGN2 at Pasteur[c]	http://bioweb.pasteur.fr/seqanal/alignment/intro-uk.html
ITERALIGN at Stanford	http://giotto.stanford.edu/~luciano/iteralign.html
• **Motif-based local multiple alignment**	
MEME at SDSC	http://www.sdsc.edu/MEME/
MEME at Pasteur	http://bioweb.pasteur.fr/seqanal/motif/meme/
MATCH-BOX	http://www.fundp.ac.be/sciences/biologie/bms/matchbox_submit.html
BLOCK Maker at FHCRC	http://www.blocks.fhcrc.org/blockmkr/make_blocks.html
PIMA at BCM	http://kiwi.imgen.bcm.tmc.edu:8088/search-launcher/launcher.html
PIMA II at BMERC	http://bmerc-www.bu.edu/protein-seq/pimaII-new.html

[a] Possibility to display and edit alignment with the JALVIEW JAVA applet.

[b] Coloured alignments.

[c] In combination with many WWW tools for molecular phylogeny.

Table 5 Software for multiple alignments

ClustalW (UMPV)[a]	ftp://ftp-igbmc.u-strasbg.fr/pub/ClustalW/
ClustalX (UMPV)[a] (ClustalW + graphical interface)	ftp://ftp-igbmc.u-strasbg.fr/pub/ClustalX/
Multalin	http://www.toulouse.inra.fr/multalin.html
MSA (U)[a]	http://www.ibc.wustl.edu/ibc/msa.html
DIALIGN (U)[a]	http://www.gsf.de/biodv/dialign.html
DCA (U)[a]	http://bibiserv.techfak.uni-bielefeld.de/dca/
RIW/DNR (U)[a]	ftp://ftp.genome.ad.jp/pub/genome/saitama-cc/
MACAW (MP)[a]	ftp://ftp.bio.indiana.edu/molbio/align/macaw/

[a] Availability: U = UNIX , M = Macintosh, P = PC, V = VMS.

(27) can also be used. For a review of multiple alignment algorithms see reference (28).

Many of the programs reviewed here can be used directly through the WWW (see *Table* 4) or downloaded over the Internet to be installed on a local computer (see *Table* 5).

4.1 Optimal methods for global multiple alignments

In this section, we will mention several methods that are said to be optimal, because they guarantee to find the 'best' multiple alignment among all possible solutions for a given scoring scheme. It should be stressed that the term 'optimal' is taken here in its mathematical meaning. Whether a mathematically optimal alignment corresponds or not to the biologically correct alignment (i.e. the alignment that represent the most likely evolutionary scenario) will depend on the choice of parameters (weighting of substitutions and of indels, see *Section* 2.3) and on the way the multiple alignment is scored.

4.1.1 Scoring schemes for multiple alignments

In principle, the score of a multiple alignment should reflect its likelihood (according to a given evolutionary model). There are different ways to measure the score (or cost) of a multiple alignment. In the following we consider that a sequence is an ordered set of letters taken from an alphabet Σ. An alignment of n sequences $S1, \ldots, Sn$ can be defined as a matrix $a(S1, \ldots ,Sn) = A$, where each entry Aij is either a letter from Σ or a null symbol (the gap symbol, usually denoted by –). The row i from A is the sequence S_i, after gaps are removed.

In the simplest model, the cost of an alignment of n sequences is defined as the sum of the cost of its columns. However, this model is crude because each column of the alignment is considered independently of its context (i.e. a gap of length L is considered as corresponding to L independent indels).

In more realistic models, a gap is interpreted as one single mutational event (a deletion or an insertion of L residues) and associated with a cost that depends on its length (see *Section* 2.3.2). With such models, pairwise alignment costs are defined as the sum of substitution and gap costs. However, the definition of the multiple alignment cost is more complex. One possible solution, known as the Sum of Pairs (SP) alignment cost (29), consists of calculating the multiple alignment cost from pairwise alignment costs. A multiple alignment *a(S1, … ,Sn)* contains *n(n-1)/2* pairwise alignments *a(Si.,Sj)* where $1 \leq i < j \leq n$. Each *projection* *a(Si.,Sj)* is the pairwise alignment built from *a(S1, … ,Sn)* by removing all the rows except the rows i and j, and then by removing all the columns that contains two null letters. The SP multiple alignment cost is defined as the sum of all its projections costs (29).

Simple SP alignment cost may, however, be inappropriate when some groups of sequences are heavily over or under-represented in a family. This drawback may be corrected by introducing a proper weighting system (30, 31) which assigns a weight to each sequence. This can be used to give less weight to sequences from overrepresented groups. Another solution consists of using a cost function based on an evolutionary tree. The tree leaves are the sequences we want to align, and the internal nodes are their hypothetical ancestral sequences. For a given tree, the cost of an alignment *a(S1, … ,Sn)* is the sum of all its projections *a(Si.,Sj)* on adjacent sequences *Si* and *Sj* in the tree (32).

4.1.2 Algorithmic complexity of optimal multiple alignment methods

The optimal alignment is the one with the maximal score (or the minimal cost). Needleman and Wunsh (33) proposed an efficient algorithm, based on dynamic programming, to compute this minimal cost for pairwise alignments. This dynamic programming approach can be easily generalized to more than two sequences. However, computing the minimal alignment cost of *n* sequences, each of length *l* requires $o(2^n l^n)$ time and $o(l^n)$ space (i.e. time proportional to $2^n l^n$ and computer memory proportional to l^n) and the complexity is even higher if gap cost are not linear (see *Section 2.3.2*). Such an algorithm cannot be used, in practice, for much more than three sequences. For example, to align ten sequences of length 100, on a very fast computer that would need 10^{-9} sec to compute the score for one column of a multiple alignment, it would take approximately three million years ($2^{10}\ 100^{10}\ 10^{-9} \cong 10^{14}$ sec) to compute the alignment. This assumes we have approximately ten billion giga-bytes of memory.

Carrillo and Lipman (29) proposed a branch and bound algorithm to compute a minimal SP cost alignment. This algorithm uses an upper bound of the alignment SP cost to limit the space and time used by dynamic programming. This approach is implemented in the program MSA (34). A new version of MSA with substantial improvements in time and space usage is available (35). Despite these improvements, MSA cannot easily be used for more than about 10 short sequences.

As stated previously, cost functions based on an evolutionary tree are, in principle, better than SP alignment costs to measure the likelihood of an alignment. However, the alignment problem under an evolutionary tree is even harder than the SP alignment problem, as the algorithm has to find the alignment, the tree, and the ancestral sequences such that the alignment cost is minimal. The problem remains hard even if the tree is given (36).

4.2 Progressive global alignment

Progressive alignment is the most commonly used method to align biological sequences. This heuristic approach is very rapid, requires low memory space and offers good performance on relatively well-conserved, homologous sequences (37, 38).

4.2.1 Description of progressive alignment methods

Progressive alignment consists of building a multiple alignment using pairwise alignments in three steps:

(a) Compute the alignment scores (or distances) between all pairs of sequences.

(b) Build a *guide* tree that reflects the similarities between sequences, using the pairwise alignment distances.

(c) Align the sequences following the guide tree. Corresponding to each node in the tree, the algorithm aligns the two sequences or alignments that are associated with its two daughter nodes. The process is repeated beginning from the tree leaves (the sequences) and ending with the tree root.

Depending on the algorithms, steps (b) and (c) are done separately, or merged in one step where the tree topology is deduced from the progressive alignment process.

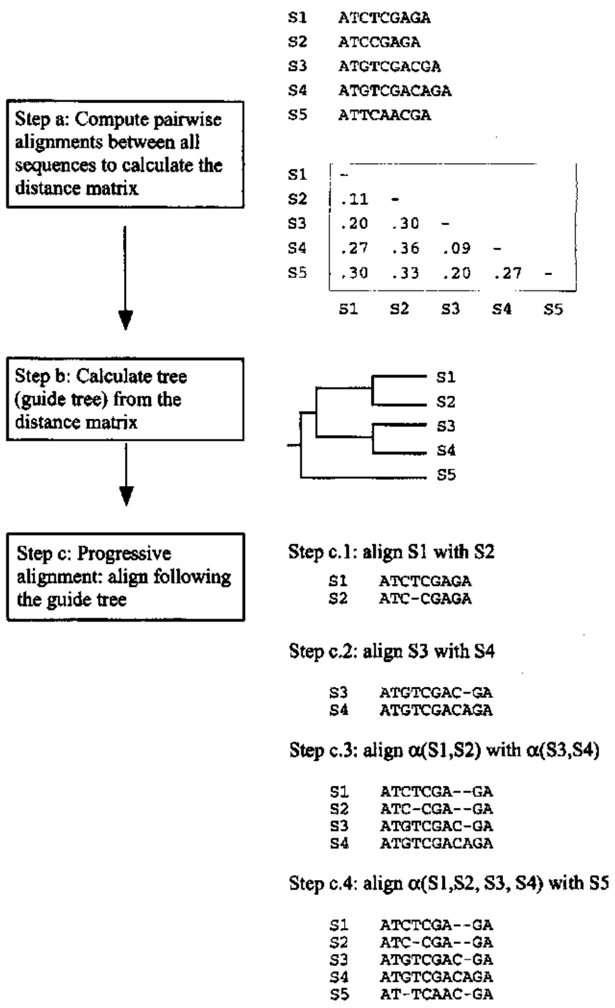
Figure 2 illustrates the progressive alignment process. *S1* is first aligned with *S2* following the given tree, *S3* is then aligned with *S4*, then the two alignments $\alpha(S1,S2)$ and $\alpha(S3,S4)$ are aligned together, and finally *S5* is aligned with $\alpha(S1,S2,S3,S4)$. Notice that even if $\alpha(S1,S2)$ and $\alpha(S3,S4)$ are optimal alignments computed by dynamic programming, the progressive alignment approach does not guarantee that $\alpha(S1,S2,S3,S4)$ is optimal for a multiple alignment cost function (SP cost or the tree cost for example).

A great number of tools that use a progressive alignment approach have been proposed, they differ by the methods used in at least one of the three steps.

In the first step (a) the pairwise alignment cost can be computed by dynamic programming, or by heuristic algorithms. The multiple alignment program CLUSTAL W (9) for example allows one to choose either dynamic programming or a heuristic method. Dynamic programming gives more accurate scores but is slower than heuristic methods.

Different algorithms can be used to build a tree (step b) given a distance matrix between sequences. Following Feng and Doolittle (37), early versions of CLUSTAL (39) used the UPGMA algorithm (40). However, UPGMA is notorious for giving incorrect branching orders when rates of substitution vary in different lineages. Therefore, CLUSTAL W (9) now uses the Neighbor-Joining (41) algorithm to build the guide tree.

The main problem in the third step (c) consists of aligning two alignments.


Step a: Compute pairwise alignments between all sequences to calculate the distance matrix

```
S1   ATCTCGAGA
S2   ATCCGAGA
S3   ATGTCGACGA
S4   ATGTCGACAGA
S5   ATTCAACGA
```

S1	-				
S2	.11	-			
S3	.20	.30	-		
S4	.27	.36	.09	-	
S5	.30	.33	.20	.27	-
	S1	S2	S3	S4	S5

Step b: Calculate tree (guide tree) from the distance matrix

Step c: Progressive alignment: align following the guide tree

Step c.1: align S1 with S2

```
S1   ATCTCGAGA
S2   ATC-CGAGA
```

Step c.2: align S3 with S4

```
S3   ATGTCGAC-GA
S4   ATGTCGACAGA
```

Step c.3: align α(S1,S2) with α(S3,S4)

```
S1   ATCTCGA--GA
S2   ATC-CGA--GA
S3   ATGTCGAC-GA
S4   ATGTCGACAGA
```

Step c.4: align α(S1,S2, S3, S4) with S5

```
S1   ATCTCGA--GA
S2   ATC-CGA--GA
S3   ATGTCGAC-GA
S4   ATGTCGACAGA
S5   AT-TCAAC-GA
```

Figure 2 Progressive alignment process. (a) All sequences are compared to each other S_2. (b) A guide tree is calculated from the pairwise distance matrix. (c) Sequences are progressively aligned following the guide tree.

The simplest method for this problem reduces each alignment to a *consensus* sequence, and uses a pairwise alignment algorithm to do the work. In the consensus sequence, each column of the alignment is represented by its most frequent letter. Consensus alignment was used in the first version of CLUSTAL. In most programs, each alignment is considered as a *profile* (see Chapter 5). In a profile, a column is reduced to a distribution giving the frequency of each letter. Two profiles are aligned as two sequences by dynamic programming without major modification of the algorithm. The alignment of two profiles of length l takes $o(a^2l^2)$, where *a* is the alphabet size. CLUSTAL W uses profile alignment with position-specific gap penalties (see *Section 2.3.2*).

4.2.2 Problems with progressive alignment methods

An important problem with this progressive alignment approach stems from the 'greedy' nature of the algorithm: any mistakes that appear during early alignments cannot be corrected later as new sequence information is added. For example, suppose that we have to align three sequences (*x*, *y*, *z*). Consider a short fragment of these sequences for which the optimal alignment is:

```
x ACTTA
y A-GTA
z ACGTA
```

Suppose that the guide tree based on pairwise comparison of entire sequences indicates that we should first align sequence *x* with sequence *y*, followed by the alignment of sequence *z* with the first two (already aligned together). At the first step, there are three possible alignments of *x* and *y* giving exactly the same score:

```
x ACTTA    x ACTTA    x ACTTA
y A-GTA    y AGT-A    y AG-TA
```

At the later step, the gap that was introduced cannot be changed. Thus adding sequence *z* could give the following three alignments:

```
x ACTTA    x ACTTA    x ACTTA
y A-GTA    y AGT-A    y AG-TA
z ACGTA    z ACGTA    z ACGTA
```

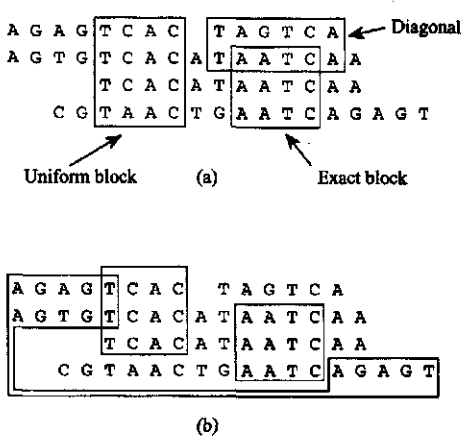
Only the first of these alignments is optimal. At the first step, only one of the three possibilities will be used. If it is the wrong one, we cannot correct this later.

To avoid that problem, iterative optimization strategies such as RIW or DNR (42) have been proposed. These methods are reported to perform better than CLUSTAL W (42). However, although these methods are much faster than optimal algorithms, they are still to slow for large dataset.

Another limitation of the progressive approach described above is that it requires computing pairwise distances between all sequences to calculate the guide tree. One may sometimes have to align set of homologous sequences that include some non-overlapping fragments (e.g. partial protein sequences). When sequences are non-overlapping they are obviously completely unrelated and thus the guide tree generated may be totally false. The alignment produced in this case can be unpredictable.

4.3 Block-based global alignment

The sequences to be compared may share conserved blocks, separated by non-conserved regions containing large indels. In such cases, the result of optimal or progressive global alignment methods will depend greatly on the choice of gap penalty parameters. An alternative to these approaches consists of searching for



A G A G T C A C T A G T C A
A G T G T C A C A T A A T C A A
T C A C A T A A T C A A
C G T A A C T G A A T C A G A G T
Diagonal
Uniform block
(a)
Exact block
(b)

Figure 3 (a) Consistent set of blocks. (b) Non-consistent set of blocks.

conserved blocks that will be used as anchors in order to align the sequences. Blocks are alignments of fragments (segments) of sequences (local alignments). Most methods consider gap-free blocks. Depending on the programs used, the blocks allowed can be *exact* (composed of identical segments) or not exact and they may be *uniform* (found in every sequence) or not. The selected set of blocks must be *consistent*, i.e. the blocks can occur together in a multiple global alignment (*Figure* 3). Once blocks have been computed, it is possible to use a classical approach to align regions between blocks (e.g. ref. 43).

The first multiple block alignment program (44) used a sorting algorithm in order to compute uniform exact blocks. Faster algorithms based on suffix trees (45), or equivalent data structures, can also be used to compute exact blocks. However, homologous regions are rarely exactly conserved. ASSEMBLE (46) performs a dot matrix analysis on all pairs of sequences and then compares these dot matrices to find uniform blocks that are not necessarily exact. In practice, it often happens that some blocks are not present in all sequences. Thus, a further improvement has consisted of developing methods that allow blocks that are not necessarily uniform. DIALIGN (47, 48) is based on computing gap-free blocks between pairs of segments (*diagonals*).

A set of uniform blocks is consistent when each pair of blocks is ordered (they do not cross each other). Using this observation, selecting an optimal consistent set of blocks can be reduced to a classic optimal-path algorithm in a graph (44). The optimal-path algorithm requires $o(M^2)$ time for M blocks. Faster algorithms (sub-quadratic) have been proposed in order to compute an optimal consistent uniform set of blocks (49, 50). However, finding an optimal consistent set of *non-uniform* blocks is an intractable problem (51). Indeed, the consistency of non-uniform blocks cannot be reduced to a binary relation between them. A set of three non-uniform blocks, such that all its three pairs of blocks are consistent, is not necessarily consistent. To compute a 'good' consistent set of diagonals, DIALIGN uses a heuristic algorithm in which diagonals are incorporated by decreasing score order into a consistent set of diagonals. Diagonals not consistent with the set of selected diagonals are rejected. In order to check if a new diagonal is consistent or not with the set of selected diagonals, DIALIGN maintains a data structure in $o(kL^2)$ time, for k sequences of total length L. This makes it slower than progressive alignment programs. This computation time can however be reduced to $o(k^2L + L^2)$ (52) and even, thanks to recent developments, to $o(L^2)$. Thus, faster versions of block-based alignment methods should be available in the near future.

4.4 Motif-based local multiple alignments

The sequences to compare may share similar regions, without necessarily being globally related. These homologous modules may occur in different relative positions and may be duplicated in different sequences. In such cases it is not possible to compute a global alignment, but one may look for 'good' local alignments of segments taken in the sequences. Calculating local alignments consists of finding approximate repeated patterns in a set of sequences. Dynamic programming has been adapted in order to find the maximal diagonal score for pairwise comparison (53). For more than two sequences the problem is hard and heuristics are needed as for the global multiple alignment problem.

PRALIGN (54) computes consensus words for a given word length. For each possible word w of length k one may define the *neighbourhood* of w as the set of k length words whose score with w is higher than a given cut-off. The score of w is then the sum of all the scores with his neighbours that occur in the given sequences. PRALIGN tries to compute the best score words (consensus words) of fixed length. The main problem with this program is its space requirement: for a fixed length k the space used is proportional to 20^k (for proteins). This space requirement could be much reduced using automatons as it is done in BLAST.

The MACAW method (55) combines pairwise comparisons in order to compute multiple local alignments. In a first step, MACAW marks, for each pair of sequences, all the diagonals with significant scores. The diagonals are then merged into local alignments. MACAW is generally considered too time-consuming for a general local alignment method, as it needs $o(L^2)$ time for the first step (L is the sum of the sequence lengths).

Most recent local alignment programs are based on statistical methods. Statistical methods use computationally efficient heuristics in order to solve optimization problems. GIBBS (56) uses iterative Gibbs sampling in order to find blocks. The computation time of this approach grows linearly with the number of input sequences. GIBBS is available in the programs MACAW and Block Maker (57). The tool MEME (58) uses an expectation-maximization (EM) algorithm (59, 60) to locate repeated patterns.

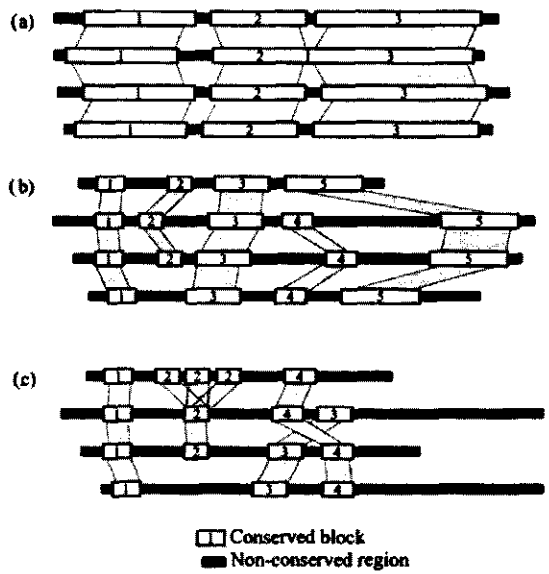
4.5 Comparison of different methods

When sequences are similar (say more that 50% pairwise identity for proteins, 70% for DNA) and are homologous over their entire length, all global alignment methods give more or less correct results. Moreover, in such cases, any reasonable set of parameters (substitution matrix, and usually, gap opening and gap extension penalties) will give similar alignments. However, when at least two sequences in a given family share less identity, or if homologous regions are interrupted by large gaps of different sizes, the result of alignment may vary considerably according to programs and parameters used.

Several comparative analyses of multiple alignment programs have been published (42, 48, 61, 62). These comparisons are based on the ability to detect motif patterns on several protein families or based on reference alignments derived from three-dimensional protein structures. Comparative analysis can also be based on the effect of the multiple alignment programs on phylogeny. Such a study was done on 18S rDNA from 43 protozoan taxa (63). These comparisons must be taken only as indications. Indeed, the parameter values (substitution matrices, gap penalty, etc.) used in these comparisons may not be optimal for other sequence families (61). In addition these parameters are not really comparable, even if the programs use the same strategies. For example a gap opening score of 5 does not have the same meaning in CLUSTAL W (9) as it does in MULTAL (38), as the value 5 will be modified in the programs (multiplied by constants for example). For these reasons and because no known method guarantees to find the *correct* alignment, it is still necessary to combine different methods from different families of algorithms and human expertise to obtain satisfactory alignments.

Figure 4 summarizes indications to guide users in their choice according to the sequences they have to align. For the alignment of two sequences, one should use an optimal pairwise alignment method (for example LALIGN or SIM (64), see *Table* 6). For more than two sequences, one generally has to use heuristic approaches. As a first step, the user should try to compute the multiple alignment with a progressive alignment program. These programs are rapid, do not demand large memory capacity and may thus be run on large dataset even on micro-computers. Among programs using this approach, we recommend CLUSTAL W (or its graphical user interface version: CLUSTAL X) (65, 66). This includes useful features such as automatic selection of amino-acid substitution matrix during alignment and lower weighting of gaps in potential protein loops. If this first alignment shows that all sequences are related to each other over their entire lengths, it is unlikely that any other method will give a better result (*Figure 4a*).

However, if there are some highly divergent sequences, large gaps, or poorly conserved regions it is recommended to compare the results of different methods and/or sets of parameters. *Figure 4b* shows homologous sequences sharing conserved blocks separated by non-conserved regions of varying size. This situation, which is frequently observed in practice (e.g. in genomic DNA sequences and in many protein families), is particularly error prone for progressive alignment methods, notably because the linear weighting of gaps tends to over-penalize long indels. Block-based global methods (e.g. DIALIGN, ITERALIGN) (47, 48, 67) are not sensitive to these long gaps and are particularly appropriate for such



	Comment	Appropriate approach
(a)	Sequences are related over their entire length.	Progressive global alignment method (e.g. CLUSTAL W).
(b)	Sequences share conserved blocks, separated by non-conserved regions containing large indels. Blocks are consistent (i.e. in the same order) but not necessarily uniform (i.e. some blocks may be missing in some sequences).	Block-based global alignment method (e.g. DIALIGN, ITERALIGN). Compare alignments produced by different programs (including progressive methods).
(c)	Sequences contain a non-consistent set of conserved blocks (i.e. some blocks are duplicated or occur in a different order along sequences).	Motif-based local alignment method (e.g. MEME). Compare alignments produced by different programs.

Figure 4 Choice of multiple alignment methods according of the nature of the sequence set.

Table 6 Websites for pairwise alignments

LFASTA at PBIL[a]	http://pbil.univ-lyon1.fr/lfasta.html
SIM at ExPASy[a]	http://www.expasy.ch/sprot/sim-prot.html
BLAST two sequences at NCBI	http://www.ncbi.nlm.nih.gov/gorf/bl2.html
LALIGN at CRBM	http://www2.igh.cnrs.fr/bin/lalign-guess.cgi
SIM, GAP, NAP, LAP	http://genome.cs.mtu.edu/align/align.html

[a] Possibility to visualize pairwise alignments with LALNVIEW (83).

cases. Moreover, one drawback of progressive methods (but also of optimal global alignment methods) is that an alignment is produced even if sequences are not related, possibly of random origin. DIALIGN and ITERALIGN on the contrary do not attempt to generate a global alignment if sequences are only locally related. Another interesting feature of these programs is that they indicate the significance of the alignment: in DIALIGN for example, regions that are not considered to be aligned (e.g. a non-conserved region between two aligned blocks) are printed as lower-case letters whereas aligned residues are in upper-case.

Global methods (optimal, progressive, or block-based) are appropriate only if all conserved blocks are consistent (see *Figure* 3). If, as presented in *Figure 4c*, some domains are duplicated, or ordered differently along sequences it is necessary to use a local multiple alignment method to align all related domains. The WWW version of the MEME tool (see *Table* 4) provides a graphical representation of the motifs found in sequences which proves to be very helpful to analyse the domain organization of proteins.

4.6 Particular case: aligning protein-coding DNA sequences

It is sometimes necessary to align protein-coding DNA sequences rather than proteins. Two examples are the design of primers to identify related genes by PCR or for molecular phylogenies relying on the measure of substitution rates at synonymous (Ks) or non-synonymous (Ka) sites of codons. Due to the degeneracy of the genetic code, it is generally more difficult to align coding DNA sequences than their protein translation. Moreover, some ambiguities in DNA alignments may be solved when considering the protein translation. For example, the two DNA alignments below have exactly the same similarity score:

```
     L    F          L    F
    CTT  TTC        CTT  TTC
    CTC  ---        ---  CTC
     L    -          -    L
 (a)            (b)
```

However, the second alignment can be rejected unambiguously taking into account the protein translation. Thus, the procedure commonly used to align protein-coding DNA sequences is the following:

(a) Extract coding DNA sequences and the corresponding protein translation.

(b) Align protein sequences.

(c) Back-translate the protein alignment into a nucleic acid alignment.

The program PROTAL2DNA (C. Letondal, unpublished) has been written for this purpose, and is available at the Pasteur WWW server:

http://bioweb.pasteur.fr/seqanal/interfaces/protal2dna.html

Note that the WWW-QUERY server (see *Table 2*) may be used to extract both coding DNA sequences and their corresponding protein translation (taking into account species- or organelle-specific genetic codes).

5 Visualizing and editing multiple alignments

Results of multiple alignment programs are generally saved simply as text files. There is presently no standard format for multiple alignments. However, the MSF output format (*Figure 5*) is provided by most of popular alignment programs and is recognized by many programs that require alignments as an input (e.g. molecular phylogeny, profile searches). The MASE format presents the advantage of allowing the inclusion of annotation regarding the whole alignment or specific to each sequence (*Figure 6*). Textual representation of multiple alignments is,

```
PileUp


   MSF:  171  Type: P    Check:  8689   ..

 Name: BTG1_BOVIN      oo  Len:  171  Check:  4676  Weight:  1.00
 Name: BTG1_CHICK      oo  Len:  171  Check:  3006  Weight:  1.00
 Name: BTG2_HUMAN      oo  Len:  171  Check:  5090  Weight:  1.00
 Name: BTG2_MOUSE      oo  Len:  171  Check:  5917  Weight:  1.00

//


BTG1_BOVIN        MHPFYSRAAT MIGEIAAAVS FISKFLRTKG LTSERQLQTF SQSLQELLAE
BTG1_CHICK        MHPALYTRAS MIREIAAAVA FISKFLRTKG LMNERQLQTF SQSLQELLAE
BTG2_HUMAN        ..MSHGKGTD MLPEIAAAVG FLSSLLRTRG CVSEQRLKVF SGALQEALTE
BTG2_MOUSE        ..MSHGKRTD MLPEIAAAVG FLSSLLRTRG CVSEQRLKVF SRALQDALTD


BTG1_BOVIN        HYKHHWFPEK PCKGSGYRCI RINHKMDPLI GQAAQRIGLS SQELFRLLPS
BTG1_CHICK        HYKHHWFPEK PCKGSGYRCI RINHKMDPLI GQAAQRIGLS SQELFQLLPS
BTG2_HUMAN        HYKHHWFPEK PSKGSGYRCI RINHKMDPII SRVASQIGLS QPQLHQLLPS
BTG2_MOUSE        HYKHHWFPEK PSKGSGYRCI RINHKMDPII SKVASQIGLS QPQLHRLLPS


BTG1_BOVIN        ELTLWVDPYE VSYRIGEDGS ICVLYEASPA GGSTQNSTNV QMVDSRISCK
BTG1_CHICK        ELTLWVDPYE VSYRIGEDGS ICVLYEAAPA GGS.QNNTNM QMVDSRISCK
BTG2_HUMAN        ELTLWVDPYE VSYRIGEDGS ICVLYEEAPL AAS....... ...CGLLTCK
BTG2_MOUSE        ELTLWVDPYE VSYRIGEDGS ICVLYEEAPV AAS....... ...YGLLTCK


BTG1_BOVIN        EELLLGRTSP SKNYNMMTVS G
BTG1_CHICK        EELLLGRTSP SKSYNMMTVS G
BTG2_HUMAN        NQVLLGRSSP SKNYVMAVSS .
BTG2_MOUSE        NQMMLGRSSP SKNYVMAVSS .
```

Figure 5 Example of alignment in MSF format.

```
;; Header (at least one line)
; Sequence-specific annotation  (at least one line)
sequence name 1
ATA-GGGA ....
; Sequence-specific annotation  (at least one line)
sequence name 2
ATA-GGGA ....
; Sequence-specific annotation  (at least one line)
sequence name 3
ATAGGGGA ....
etc.

Example:


;;Aligned by clustal on Wed Dec  2 09:12:04 1998
;;Block: conserved domain: 49..89
;;Group of species = 2 BTG1: 1, 2
;AC   P53348; O18950;
;DE   BTG1 PROTEIN (B-CELL TRANSLOCATION GENE 1 PROTEIN) (MYOCARDIAL
;DE   VASCULAR INHIBITION FACTOR) (VIF).
BTG1_BOVIN
MHPFYSRAATMIGEIAAAVSFISKFLRTKGLTSERQLQTFSQSLQELLAEHYKHHWFPEK
PCKGSGYRCIRINHKMDPLIGQAAQRIGLSSQELFRLLPSELTLWVDPYEVSYRIGEDGS
ICVLYEASPAGGSTQNSTNVQMVDSRISCKEELLLGRTSPSKNYNMMTVSG
;AC   P34743;
;DE   BTG1 PROTEIN (B-CELL TRANSLOCATION GENE 1 PROTEIN).
BTG1_CHICK
MHPALYTRASMIREIAAAVAFISKFLRTKGLMNERQLQTFSQSLQELLAEHYKHHWFPEK
PCKGSGYRCIRINHKMDPLIGQAAQRIGLSSQELFQLLPSELTLWVDPYEVSYRIGEDGS
ICVLYEAAPAGGS-QNNTNMQMVDSRISCKEELLLGRTSPSKSYNMMTVSG
;AC   P78543;
;DE   BTG2 PROTEIN PRECURSOR (NGF-INDUCIBLE ANTI-PROLIFERATIVE
;DE   PROTEIN PC3).
BTG2_HUMAN
--MSHGKGTDMLPEIAAAVGFLSSLLRTRGCVSEQRLKVFSGALQEALTEHYKHHWFPEK
PSKGSGYRCIRINHKMDPIISRVASQIGLSQPQLHQLLPSELTLWVDPYEVSYRIGEDGS
ICVLYEEAPLAAS----------CGLLTCKNQVLLGRSSPSKNYVMAVSS-
;AC   Q04211;
;DE   BTG2 PROTEIN PRECURSOR (NGF-INDUCIBLE PROTEIN TIS21).
BTG2_MOUSE
--MSHGKRTDMLPEIAAAVGFLSSLLRTRGCVSEQRLKVFSRALQDALTDHYKHHWFPEK
PSKGSGYRCIRINHKMDPIISKVASQIGLSQPQLHRLLPSELTLWVDPYEVSYRIGEDGS
ICVLYEEAPVAAS----------YGLLTCKNQMMLGRSSPSKNYVMAVSS-
```

Figure 6 MASE format. This format is used to store nucleotide or protein multiple alignments along with annotations relative to the whole alignment (indicated in the header), or specific to each sequence. The beginning of the file must contain a header containing at least one line (but the content of this header may be empty). The header lines begin by ';;'. The body of the file has the following structure: First, each entry begins with one (or more) annotation lines. Annotation lines begin by the character ';'. Again, this annotation line may be empty. After the annotations, the name of the sequence is written on a separate line. At last, the sequence itself is written on the following lines.

however, poorly informative. Therefore, graphical interfaces have been developed to manipulate and edit multiple alignments. Generally, these interfaces allow users to colour or shade residues (amino acids or nucleotides) according to various criteria such as physico-chemical properties, degree of conservation within the alignment, hydrophobicity or secondary structure. The use of colours is very helpful to interpret a multiple alignment. It gives a much more comprehensive view of the information embedded in a multiple alignment than a simple textual representation. Besides, these interfaces propose several interesting facilities detailed below. A list of such graphical interfaces is given in *Table 7*.

Table 7 Multiple alignment viewers and editors

Jalview (J)[ab]	http://www2.ebi.ac.uk/~michele/jalview/contents.html
CINEMA 2.1 (J)[ac]	http://www.biochem.ucl.ac.uk/bsm/dbbrowser/CINEMA2.1/
SEAVIEW (U)[a]	http://pbil.univ-lyon1.fr/software/seaview.html
MPSA (UM)[a]	http://www.ibcp.fr/mpsa/
Se-Al (M)[a]	http://evolve.zps.ox.ac.uk/Se-Al/Se-Al.html
ClustalX (UMPV)[a] (ClustalW + graphical interface)	ftp://ftp-igbmc.u-strasbg.fr/pub/ClustalX/
DCSE (U)[a]	http://indigo2.uia.ac.be:80/~peter/dcse/

[a] Availability: U = UNIX , M = Macintosh, P = PC, V = VMS, J = JAVA applet.

[b] Links to sequence databases.

[c] Possibility to download alignments from the PRINTS database.

5.1 Manual expertise to check or refine alignments

Whatever the quality of the software, it is necessary to examine the alignment to check that there are no obvious errors. Alignments of sequences with large length differences, or with duplicated domains are particularly error prone, even if the sequences are not very divergent. A good control consists in verifying that local similarities detected by pairwise sequence comparisons are preserved in the multiple alignment. For such purposes, one may use the results of similarity searches (BLAST, FASTA, etc.), or run pairwise local alignment software (see *Table* 6) or use a dot-plot representation. Pairwise alignments can be computed directly from JALVIEW. SEAVIEW (68) includes a dot-plot utility that can be used to drive the alignment, semi-automatically. CINEMA (69) allows user to run a BLAST search or a dot-plot on selected sequences.

In some cases, it may be necessary to refine part of the alignment. Experienced users are often able to recognize residues that have been misaligned. In some cases, external information (e.g. known interactions or 3-D structures) may also reveal alignment errors. SEAVIEW and CLUSTALX allow users to run CLUSTALW on a specified region and/or a specified set of sequences, without changing the rest of the alignment.

Alignment editors (except CLUSTALX) also allow users to manually add or remove gaps in the alignment. In some interfaces (e.g. JALVIEW or SEAVIEW), it is possible to define groups in order to edit a subset of sequences. In the absence of objective criteria, manual alignment editing should, however, be used with caution.

5.2 Annotating alignments, extracting sub-alignments

The SEAVIEW software allows users to annotate alignments (e.g. to indicate the location of relevant features such as enzyme active sites or RNA splicing signals). The locations of annotations are correctly preserved after indels are inserted or moved. This software also allows one to define groups of sequences and blocks in the alignment and thus to extract sub-alignments. This feature is particularly useful when building phylogenetic trees where one needs to exclude unreliable parts of alignments (i.e. regions for which the alignment is ambiguous). It is also useful to select particular domains for profile searches. Definitions of groups and blocks can be saved along with the alignment in MASE format (*Figure* 6).

5.3 Comparison of alignment editors

Each of the editors presented in *Table* 7 has some specific useful features, some of which have been mentioned above. Programs written in JAVA (JALVIEW, CINEMA) present two advantages. First, they can be used from any computer and run directly from a WWW browser (although, depending on the network load, the time necessary to download the JAVA applet through the internet sometimes limits considerably their usefulness). Secondly, thanks to the network communication facilities provided by JAVA, these programs allow users to directly access information stored in sequence databases available on the internet. CLUSTALX is a graphical interface to the CLUSTALW program and not simply an alignment viewer. However, it does not allow manual editing of alignments. MPSA is dedicated to protein secondary structure prediction. SEAVIEW is particularly suited for phylogenetic analyses and can notably be used in combination with the PHYLOWIN graphical interface dedicated to molecular phylogeny (68).

5.4 Alignment shading software, pretty printing, logos, etc.

To publish the results of such analyses, it is generally useful to prepare a high quality colour figure of the multiple alignment. Some of the above editors (e.g. JALVIEW, CINEMA) can be used to save or print coloured alignments in a format suitable for publication. Other programs, some of which are available on the WWW, have been developed specifically for that purpose (see *Table* 8). The program LOGO (70) is intended to give a visual representation of a consensus sequence, along with possible variants.

6 Databases of multiple alignments

Databases of precompiled multiple alignments have been developed, essentially for protein sequences (71–79) but also for rRNA (80–82) and some other nucleic acid sequences (see *Table* 9). The approach used to cluster together homologous protein sequences varies according to databases. Some intend to classify together proteins homologous over their entire length (protein families), whereas others focus on the classification of protein domains (see *Table* 9). For example,

Table 8 Pretty printing, shading, logos, etc.

BOXSHADE	http://ulrec3.unil.ch/software/BOX_form.html
WebLogo	http://www.bio.cam.ac.uk/cgi-bin/seqlogo/logo.cgi
Mview	http://mathbio.nimr.mrc.ac.uk/nbrown/mview/
AMAS	http://barton.ebi.ac.uk/servers/amas_server.html

Table 9 Databases of multiple alignments

• Protein families	
PIRALN	http://www-nbrf.georgetown.edu/nbrf/getaln.html
HOVERGEN	http://pbil.univ-lyon1.fr/databases/hovergen.html
PROTOMAP	http://www.protomap.cs.huji.ac.il/
Megaclass	http://www.ibc.wustl.edu/megaclass/
• Protein domains	
ProDom	http://protein.toulouse.inra.fr/prodom.html
PRINTS	http://www.biochem.ucl.ac.uk/bsm/dbbrowser/PRINTS/PRINTS.html
DOMO	http://www.infobiogen.fr/~gracy/domo/
PFAM	http://genome.wustl.edu/Pfam/
BLOCKS	http://blocks.fhcrc.org/
• RNA/DNA	
Ribosomal Database Project	http://www.cme.msu.edu/RDP/
The rRNA WWW server	http://rrna.uia.ac.be/
ACUTS[a]	http://pbil.univ-lyon1.fr/acuts/ACUTS.html

[a] Ancient Conserved UnTranslated Sequences.

the HOVERGEN database compiles multiple alignments and phylogenetic trees for all families of vertebrate protein-coding genes along with the corresponding GenBank annotations (79). This database provides all the data necessary to decipher the orthology/paralogy relationships among vertebrate multigenic families and is thus particularly useful for phylogenetic studies or for comparative analysis of vertebrate genes. However, this approach is limited to relatively well-conserved sequences alignable over their entire length. Conversely, databases of protein domains may achieve to cluster very distantly related sequences and are useful to analyse the structure, function, and evolution of modular proteins. For some complex families, it may be useful to consult specialized databases such as those available for immunoglobulins or HOX proteins (for a complete list, see the WWW page maintained by Amos Bairoch: http://www.expasy.ch/alinks.html).

7 Summary

In this chapter, we describe methods commonly used to align homologous sequences. Searching for the best alignment consists of finding the one that represents the most likely evolutionary scenario (substitutions, insertion, and deletion). Different alignment algorithms have been developed, but none of them is ideal. Because of time and memory requirements, algorithms that guarantee to find the best alignment for a given evolutionary model can be used in practice only with a very limited number of short sequences. Therefore, non-optimal algorithms based on heuristics have been proposed to gain speed and limit memory requirements. We discuss the choice between these different methods (progressive global alignment, block-based global alignment, motif-based local multiple alignment) according to the nature of the sequences to align. We also describe graphical tools that have been developed to visualize and edit multiple alignments. Finally, we mention several databases that compile multiple alignments of homologous protein or nucleotide sequences. All internet addresses where the tools and resources described here are available are listed in the following WWW page:

http://pbil.univ-lyon1.fr/alignment.html

References

1. Kimura, M. (1983). *The neutral theory of molecular evolution*, Cambridge University Press, Cambridge.
2. Doolittle, R. F. (1994). *Trends Biochem. Sci.*, **19**, 15.
3. Wootton, J. C. and Federhen, S. (1993). *Computers Chem.*, **17**, 149.
4. Altschul, S. F. and Gish, W. (1996). In *Methods in enzymology* (ed. R. F. Doolittle), Vol. 266, p. 460. Academic Press, London.
5. Patthy, L. (1996). *Matrix Biol.*, **15**, 301.
6. Schwartz, R. M. and Dayhoff, M. O. (1978). In *Atlas of protein sequence and structure* (ed. M. O. Dayhoff), p. 353. Nat. Biomed. Res. Found., Washington DC.
7. Gonnet, G. H., Cohen, M. A., and Benner, S. A. (1992). *Science*, **256**, 1443.
8. Henikoff, S. and Henikoff, J. G. (1992). *Proc. Natl. Acad. Sci. USA*, **89**, 10915.
9. Thompson, J. D., Higgins, D. G., and Gibson, T. J. (1994). *Nucleic Acids Res.*, **22**, 4673.
10. Overington, J., Donnelly, D., Johnson, M. S., Sali, A., and Blundell, T. L. (1992). *Protein Sci.*, **1**, 216.
11. Bains, W. (1992). *Mutat. Res.*, **267**, 43.
12. Hess, S. T., Blake, J. D., and Blake, R. D. (1994). *J. Mol. Biol.*, **236**, 1022.
13. Benner, S. A., Cohen, M. A., and Gonnet, G. H. (1993). *J. Mol. Biol.*, **229**, 1065.
14. Gu, X. and Li, W. H. (1995). *J. Mol. Evol.*, **40**, 464.
15. Benson, D. A., Boguski, M. S., Lipman, D. J., Ostell, J., Ouellette, B. F. F., Rapp, B. A., *et al.* (1999). *Nucleic Acids Res.*, **27**, 12.
16. Stoesser, G., Tuli, M. A., Lopez, R., and Sterk, P. (1999). *Nucleic Acids Res.*, **27**, 18.
17. Bairoch, A. and Apweiler, R. (1999). *Nucleic Acids Res.*, **27**, 49.
18. Barker, W. C., Garavelli, J. S., McGarvey, P. B., Marzec, C. R., Orcutt, B. C., Srinivasarao, G. Y., *et al.* (1999). *Nucleic Acids Res.*, **27**, 39.
19. Schuler, G. D., Epstein, J. A., Ohkawa, H., and Kans, J. A. (1996). In *Methods in enzymology* (ed. R. F. Doolittle), Vol. 266, p. 141. Academic Press, London.
20. Etzold, T. and Argos, P. (1993). *CABIOS*, **9**, 49.
21. Gouy, M., Gautier, C., Attimonelli, M., Lanave, C., and Di-Paola, G. (1985). *Comp. Appl. Biosci.*, **1**, 167.
22. Pearson, W. R. and Lipman, D. J. (1988). *Proc. Natl. Acad. Sci. USA*, **85**, 2444.
23. Altschul, S. F., Gish, W., Miller, W., Myers, E. W., and Lipman, D. J. (1990). *J. Mol. Biol.*, **215**, 403.
24. Altschul, S. F., Madden, T. L., Schaffer, A. A., Zhang, J. H., Zhang, Z., Miller, W., *et al.* (1997). *Nucleic Acids Res.*, **25**, 3389.
25. Altschul, S. F., Boguski, M. S., Gish, W., and Wootton, J. C. (1994). *Nature Genet.*, **6**, 119.
26. Hughey, R. and Krogh, A. (1996). *Comput. Appl. Biosci.*, **12**, 95.
27. Notredame, C. and Higgins, D. G. (1996). *Nucleic Acids Res.*, **25**, 4570.

28. Chan, S. C., Wong, A. K. C., and Chiu, D. K. Y. (1992). *Bull. Math. Biol.*, **54**, 563.
29. Carillo, H. and Lipman D. (1988). *SIAM J. Appl. Math.*, **48**, 1073.
30. Altschul, S. F., Carroll, R. J., and Lipman, D. J. (1989). *J. Mol. Biol.*, **207**, 647.
31. Gotoh, O. (1995). *Comput. Appl. Biosci.*, **11**, 543.
32. Sankoff, D. (1975). *SIAM J. Appl. Math.*, **78**, 35.
33. Needleman, S. B. and Wunsh C. D. (1970). *J. Mol. Biol.*, **48**, 443.
34. Lipman, D. J., Altschul, S. F., and Kececioglu, J. D. (1989). *Proc. Natl. Acad.. Sci. USA*, **86**, 4412.
35. Gupta, S., Kececioglu, J. D., and Schäffer, A. (1995). *J. Comput. Biol.*, **2**, 459.
36. Jiang, T., Lawler, E. L., and Wang, L. (1994). *ACM Sympos. Theory Comput.*, **26**, 760.
37. Feng, D. F. and Doolittle, R. F. (1987). *J. Mol. Evol.*, **25**, 351.
38. Taylor, W. R. (1988). *J. Mol. Evol.*, **28**, 161.
39. Higgins, D. G. and Sharp, P. M. (1988). *Gene*, **73**, 237.
40. Sneath, H. A. and Sokal, R. R. (1973). *Numerical taxonomy*. W. H. Freeman. San Francisco.
41. Saitou, N. and Nei, M. (1987). *Mol. Biol. Evol.*, **4**, 406.
42. Gotoh, O. (1996). *J. Mol. Biol.*, **264**, 823.
43. Vingron, M. and Argos, P. (1989). *Comput. Appl. Biosci.*, **5**, 115.
44. Sobel, E. and Martinez, H. (1986). *Nucleic Acids Res.*, **14**, 363.
45. McCreight, E. M. (1976). *J. ACM*, **232**, 262.
46. Vingron, M. and Argos, P. (1991). *J. Mol. Biol.*, **218**, 33.
47. Morgenstern, B., Dress, A., and Werner T. (1996). *Proc. Natl. Acad. Sci. USA*, **93**, 12098.
48. Morgenstern, B., Atchley, W. R., Hahn, K., and Dress, A. (1998). *Ismb*, **6**, 115.
49. Zhang, Z., Raghavachari, B., Hardison, R., and Miller, W. (1996). *J. Comput. Biol.*, **1**, 217.
50. Myers, E. and Miller, W. (1995). In *Proc. 6th ACM-SIAM Symposium On Discrete Algorithms*, p. 38.
51. Zhang, Z., He, B., and Miller, W. (1996). *J. Disc. Appl. Math.*, **71**, 337.
52. Abdeddaïm, S. (1997). In *Lecture notes in computer science*, Vol. 1264, p. 167. Springer-Verlag.
53. Smith, R. F. and Waterman, M. S. (1981). *J. Mol. Biol.*, **147**, 195.
54. Waterman, M. S. and Jones, R. (1990). In *Methods in enzymology* (ed. R. F. Doolittle), Vol. 183, p. 221. Academic Press, London.
55. Schuler, G. D., Altschul, S. F., and Lipman, D. J. (1991). *Proteins*, **9**, 180.
56. Lawrence, C. E., Altschul, S. F., Boguski, M. S., Liu, J. S., Neuwald, A. F., and Wootton, J. C. (1993). *Science*, **262**, 208.
57. Henikoff, S., Henikoff, J. G., Alford, W. J., and Pietrovoski, S. (1995). *Gene-COMBIS, Gene*, **163**, 17.
58. Bailey, T. L. and Elkan, C. (1995). *Ismb*, **3**, 21.
59. Lawrence, C. E. and Reilly, A. A. (1990). *Proteins*, **7**, 41.
60. Cardon, L. R. and Stormo, G. D. (1992). *J. Mol. Biol.*, **223**, 159.
61. McClure, M. A., Vasi, T. K., and Fitch, W. M. (1994). *Mol. Biol. Evol.*, **11**, 571.
62. Briffeuil, P., Baudoux, G., Lambert, C., De Bolle, X., Vinals, C., Feytmans, E., *et al.* (1998). *Bioinformatics*, **14**, 357.
63. Morrison, D. A. and Ellis, J. T. (1997). *Mol. Biol. Evol.*, **14**, 428.
64. Huang, X. and Miller, W. (1991). *Adv. Appl. Math.*, **12**, 337.
65. Thompson, J. D., Gibson, T. J., Plewniak, F., Jeanmougin, F., and Higgins, D. G. (1997). *Nucleic Acids Res.*, **25**, 4876.
66. Jeanmougin, F., Thompson, J. D., Gouy, M., Higgins, D. G., and Gibson, T. J. (1998). *Trends Biochem. Sci.*, **23**, 403.
67. Brocchieri, L. and Karlin, S. (1998). *J. Mol. Biol.*, **276**, 249.
68. Galtier, N., Gouy, M., and Gautier, C. (1996). *Comput. Appl. Biosci.*, **12**, 543.

69. Parry-Smith, D. J., Payne, A. W., Michie, A. D., and Attwood, T. K. (1998). *Gene*, **221**, GC57.
70. Schneider, T. D. and Stephens, R. M. (1990). *Nucleic Acids Res.*, **18**, 6097.
71. Srinivasarao, G. Y., Yeh, L. S. L., Marzec, C. R., Orcutt, B. C., Barker, W. C., and Pfeiffer, F. (1999). *Nucleic Acids Res.*, **27**, 284.
72. Yona, G., Linial, N., Tishby, N., and Linial, M. (1998). *Ismb*, **6**, 212.
73. Corpet, F., Gouzy, J., and Kahn, D. (1999). *Nucleic Acids Res.*, **27**, 263.
74. Attwood, T. K., Flower, D. R., Lewis, A. P., Mabey, J. E., Morgan, S. R., Scordis, P., *et al.* (1999). *Nucleic Acids Res.*, **27**, 220.
75. Gracy, J. and Argos, P. (1998). *Bioinformatics*, **14**, 164.
76. Gracy, J. and Argos, P. (1998). *Bioinformatics*, **14**, 174.
77. Bateman, A., Birney, E., Durbin, R., Eddy, S. R., Finn, R. D., and Sonnhammer, E. L. L. (1999). *Nucleic Acids Res.*, **27**, 260.
78. Henikoff, J. G., Henikoff, S., and Pietrokovski, S. (1999). *Nucleic Acids Res.*, **27**, 226.
79. Duret, L., Mouchiroud, D., and Gouy, M. (1994). *Nucleic Acids Res.*, **22**, 2360.
80. Maidak, B. L., Cole, J. R., Parker Jr, C. T., Garrity, G. M., Larsen, N., Li, B., *et al.* (1999). *Nucleic Acids Res.*, **27**, 171.
81. Van de Peer, Y., Robbrecht, E., de Hoog, S., Caers, A., de Rijk, P., and de Wachter, R. (1999). *Nucleic Acids Res.*, **27**, 179.
82. De Rijk, P., Robbrecht, E., de Hoog, S., Caers, A., Van de Peer, Y., and de Wachter, R. (1999). *Nucleic Acids Res.*, **27**, 174.
83. Duret, L., Gasteiger, E., and Perrière, G. (1996). *Comput. Appl. Biosci.*, **12**, 507.