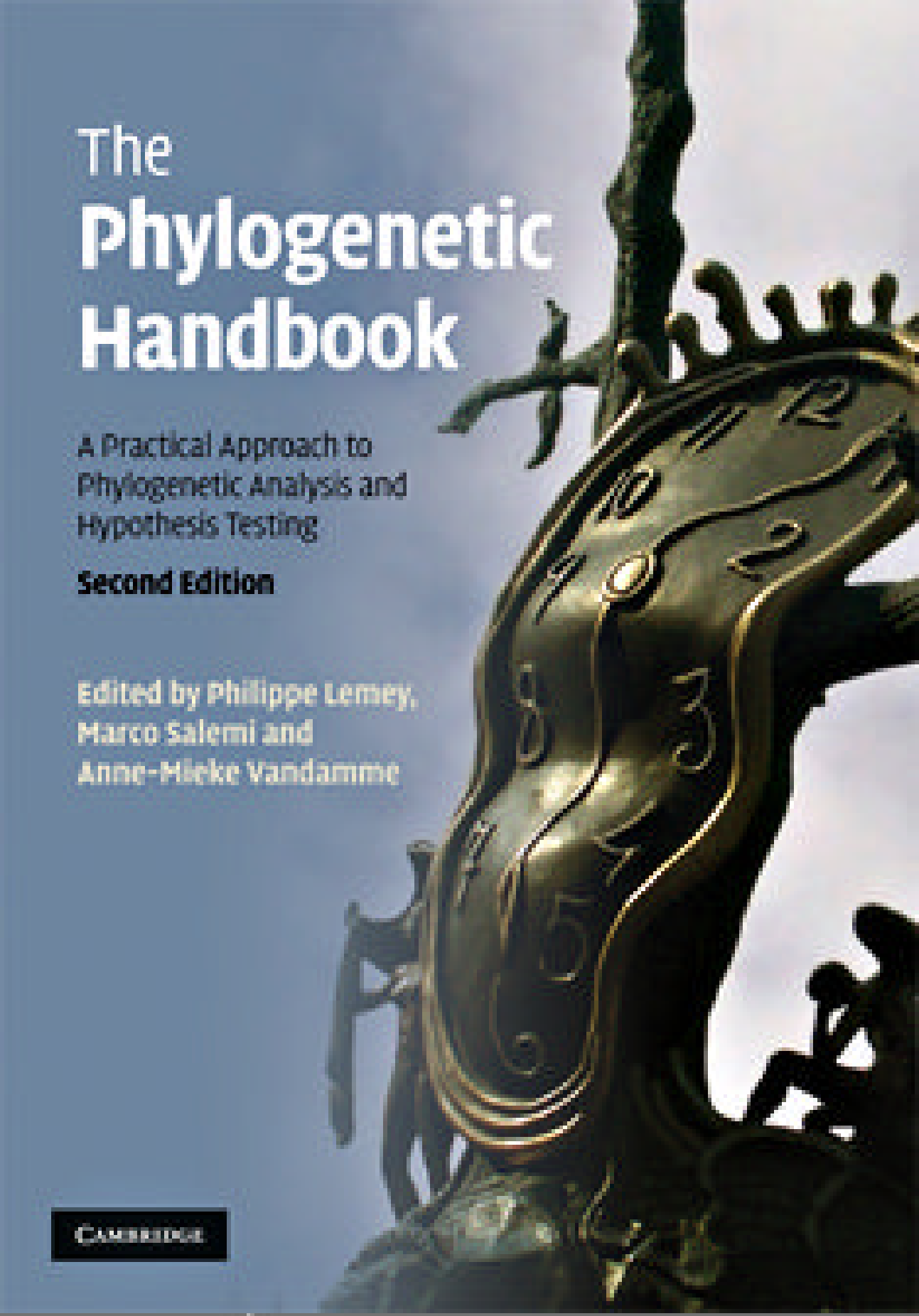
The Phylogenetic Handbook

A Practical Approach to Phylogenetic Analysis and Hypothesis Testing

Second Edition

Edited by Philippe Lemey, Marco Salemi and Anne-Mieke Vandamme



CAMBRIDGE

CAMBRIDGE www.cambridge.org/9780521877107

3

Multiple sequence alignment

THEORY

Des Higgins and Philippe Lemey

3.1 Introduction

From a biological perspective, a sequence alignment is a hypothesis about ***homology*** of multiple residues in protein or nucleotide sequences. Therefore, aligned residues are assumed to have diverged from a common ancestral state. An example of a multiple sequence alignment is shown in Fig. 3.1. This is a set of amino acid sequences of globins that have been aligned so that ***homologous*** residues are arranged in columns "as much as possible." The sequences are of different lengths, implying that gaps (shown as hyphens in the figure) must be used in some positions to achieve the alignment. The gaps represent a deletion, an insertion in the sequences that do not have a gap, or a combination of insertions and deletions. The generation of alignments, either manually or using an automatic computer program, is one of the most common tasks in computational sequence analysis because they are required for many other analyses such as structure prediction or to demonstrate sequence similarity within a family of sequences. Of course, one of the most common reasons for generating alignments is that they are an essential prerequisite for phylogenetic analyses. Rates or patterns of change in sequences cannot be analysed unless the sequences can be aligned.

3.2 The problem of repeats

It can be difficult to find the optimal alignment for several reasons. First, there may be repeats in one or all the members of the sequence family; this problem is shown

The Phylogenetic Handbook: a Practical Approach to Phylogenetic Analysis and Hypothesis Testing, Philippe Lemey, Marco Salemi, and Anne-Mieke Vandamme (eds.). Published by Cambridge University Press.

```
Human beta         --------VHLTPEEKSAVTALWGKVN--VDEVGGEALGRLLVVYPWTQRFFESFGDLST
Horse beta         --------VQLSGEEKAAVLALWDKVN--EEEVGGEALGRLLVVYPWTQRFFDSFGDLSN
Human alpha        ---------VLSPADKTNVKAAWGKVGAHAGEYGAEALERMFLSFPTTKTYFPHF-DLS-
Horse alpha        ---------VLSAADKTNVKAAWSKVGGHAGEYGAEALERMFLGFPTTKTYFPHF-DLS-
Whale myoglobin    ---------VLSEGEWQLVLHVWAKVEADVAGHGQDILIRLFKSHPETLEKFDRFKHLKT
Lamprey globin     PIVDTGSVAPLSAAEKTKIRSAWAPVYSTYETSGVDILVKFFTSTPAAQEFFPKFKGLTT
Lupin globin       --------GALTESQAALVKSSWEEFNANIPKHTHRFFILVLEIAPAAKDLFSFLKGTSE
                             *:  :   :   *  .           :  .:   * :   *  :   .

Human beta         PDAVMGNPKVKAHGKKVLGAFSDGLAHLDN-----LKGTFATLSELHCDKLHVDPENFRL
Horse beta         PGAVMGNPKVKAHGKKVLHSFGEGVHHLDN-----LKGTFAALSELHCDKLHVDPENFRL
Human alpha        ----HGSAQVKGHGKKVADALTNAVAHVDD-----MPNALSALSDLHAHKLRVDPVNFKL
Horse alpha        ----HGSAQVKAHGKKVGDALTLAVGHLDD-----LPGALSNLSDLHAHKLRVDPVNFKL
Whale myoglobin    EAEMKASEDLKKHGVTVLTALGAILKKKGH-----HEAELKPLAQSHATKHKIPIKYLEF
Lamprey globin     ADQLKKSADVRWHAERIINAVNDAVASMDDT--EKMSMKLRDLSGKHAKSFQVDPQYFKV
Lupin globin       VP--QNNPELQAHAGKVFKLVYEAAIQLQVTGVVVTDATLKNLGSVHVSKGVAD-AHFPV
                         . .:: *.  :   .                  :  *.  *  .       : .

Human beta         LGNVLVCVLAHHFGKEFTPPVQAAYQKVVAGVANALAHKYH------
Horse beta         LGNVLVVVLARHFGKDFTPELQASYQKVVAGVANALAHKYH------
Human alpha        LSHCLLVTLAAHLPAEFTPAVHASLDKFLASVSTVLTSKYR------
Horse alpha        LSHCLLSTLAVHLPNDFTPAVHASLDKFLSSVSTVLTSKYR------
Whale myoglobin    ISEAIIHVLHSRHPGDFGADAQGAMNKALELFRKDIAAKYKELGYQG
Lamprey globin     LAAVIADTVAAG---D------AGFEKLMSMICILLRSAY-------
Lupin globin       VKEAILKTIKEVVGAKWSEELNSAWTIAYDELAIVIKKEMNDAA---
                           :   :  .:      .      ..       .   :
```

Fig. 3.1 Multiple alignment of seven amino acid sequences. Identical amino acid positions are marked with asterisks (*) and biochemically conserved positions are marked with colons and periods (less conserved). The lupin sequence is a leghemoglobin and the lamprey sequence is a cyanohemoglobin. The whale sequence is from the sperm whale. The approximate positions of the alpha helices are typed in italics and bold font. The positions of two important histidine residues are underscored, and are responsible for binding the prosthetic heme and oxygen.

in the simple diagram in Fig. 3.2. It is not clear which example of a repeat unit should line up between the different members of the family. If there are large-scale repeats such as with duplications of entire protein domains, then the problem can be partly solved by excising the domains or repeat units and conducting a phylogenetic analysis of the repeats. This is only a partial solution because a single domain in one ancestral protein can give rise to two equidistant repeat units in one descendant protein and three in another; therefore, it will not be obvious how the repeat units should line up with one another. With small-grain repeats, such as those involving single nucleotides or with microsatellite sequences, the problem is even worse.

As shown in Fig. 3.2b, it is not obvious in the highlighted box if one alignment of the sequences is better than any other, except maybe cosmetically. One can only conclude that there are some repeated C residues. Fortunately, these alignment details often make no difference in a phylogenetic context and, in some cases, these

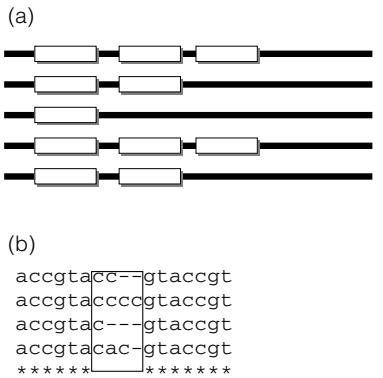


Fig. 3.2 (a) A simple diagram showing the distribution of a repeated domain (marked with a rectangle) in a set of sequences. When there are different numbers of repeats in the sequences, it can be difficult to know exactly how to arrange the alignment. Some domains will be equally similar to several domains in other sequences. (b) A simple example of some nucleotide sequences in which the central boxed alignment is completely ambiguous.

regions are simply ignored in phylogenetic inference. All computer programs will experience difficulties to unambiguously disentangle these repeats and, frequently, there is no ideal alignment. These small-scale repeats are especially abundant in some nucleotide sequences, where they can accumulate in particular positions due to mistakes during DNA replication. They tend to be localized to non-protein coding regions or to hypervariable regions; these will be almost impossible to align unambiguously and should be treated with caution. In amino acid sequences, such localized repeats of residues are unusual, whereas large-scale repeats of entire protein domains are very common.

3.3 The problem of substitutions

If the sequences in a data set accumulate few substitutions, they will remain similar and they will be relatively easy to align. The many columns of unchanged residues should make the alignment straightforward. In these cases, both manual and computerized alignment will be clear and unambiguous; in reality, sequences can and do change considerably.

For two amino acid sequences, it becomes increasingly difficult to find a good alignment once the sequence identity drops below approximately 25% (see also Chapter 9). Sequence identity, a commonly used and simple measure of sequence similarity, is the number of identical residues in an alignment divided by the total number of aligned positions, excluding any positions with a gap. Of course, a nonsensical alignment with a gap every two or three residues, for example, can result in a misleading measure of sequence similarity. In general, however,

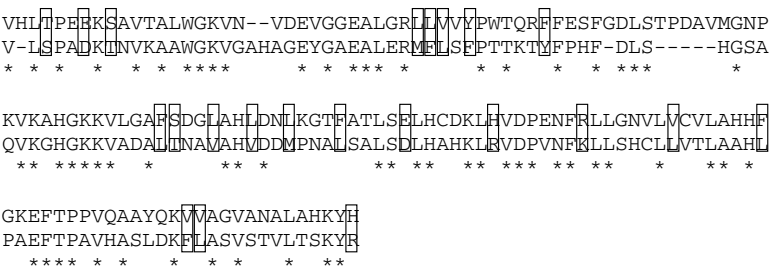


Fig. 3.3 An alignment of the amino acid sequences of human alpha globin (below) and human beta globin (above). The boxed pairs of residues are all between pairs of similar amino acids (i.e. biochemically similar side chains). The most conserved pairs are S and T (small polar); F and Y (aromatic); D and E (acidic, negatively charged); any two of H, K, and R (positively charged); and any two of F, I, L, M, or V (hydrophobic). Many of the other unmarked pairs also represent some conservation of biochemical property.

the measure is rather robust. Consider any pair of aligned homologous protein sequences with some positions where one or both residues have changed during evolution. When examining the nature of these changes, two patterns emerge (e.g. Fig. 3.3). First, the identities and differences are not evenly distributed along the sequence. Particular blocks of alignment with between 5 and 20 residues will have more identities and similar amino acids (i.e. biochemically similar; discussed in the next section) than elsewhere. These blocks will typically correspond to the conserved secondary-structure elements of α-helices and β-strands in the proteins, and are more functionally constrained than the connecting loops of irregular structure. The same pattern is observed when examining the gap locations in the sequences. This pattern of blocks is more obvious in multiple alignments of protein sequences (see Fig. 3.1). One benefit of this block-like similarity is that there will be regions of alignment which are clear and unambiguous and which most computer programs can find easily, even between distantly related sequences. However, it also means that there will be sequence regions that are more difficult to align and, if the sequences are sufficiently dissimilar, impossible to align unambiguously.

The second observed pattern is that most of the pairs of aligned, but non-identical residues, are biochemically similar (i.e. their side chains are similar). Similar amino acids such as leucine and isoleucine or serine and threonine tend to replace each other more often than dissimilar ones (see Fig. 3.3). This conservation of biochemical character is especially true in the conserved secondary-structure regions and at important sites such as active sites or ligand-binding domains. By far the most important biochemical property to be conserved is polarity/hydrophobicity. The amino acid size also matters but the exact conservation pattern will depend on the specific role of the amino acid in the protein. These patterns of conservation can significantly aid sequence alignment.

To carry out an automatic alignment, it is necessary to quantify biochemical similarity. Previous attempts focused on chemical properties and/or the genetic code; however, for both database searching and multiple alignment, the most popular amino acid scoring schemes are based on empirical studies of aligned proteins. Until the early 1990s, the most powerful method resulted from the work of Margaret Dayhoff and colleagues, who produced the famous PAM series of weight matrices (Dayhoff *et al.*, 1978) (see also Chapter 9). Protein substitution matrices are 20 × 20 tables that provide scores for all possible pairs of aligned amino acids. The higher the score, the more weight is attached to the pair of aligned residues. The PAM matrices were derived from the original empirical data using a sophisticated evolutionary model, which allowed for the use of different scoring schemes depending on how similar the sequences were. Although this was a powerful capability, most biologists used the default table offered by whatever software they were using. Currently, the PAM matrices have largely been superseded by the BLOSUM matrices of Jorja and Steven Henikoff (1992), which are also available as a series, depending on the similarity of the sequences to be aligned. The most commonly used BLOSUM62 matrix, shown in Fig. 3.4, illustrates how different pairs of identical residues get different scores. Less weight is assigned to residues which change readily during evolution, such as alanine or serine, and more weight is assigned to those that change less frequently, such as tryptophan. The remaining scores are either positive or negative, depending on whether a particular pair of residues are more or less likely to be observed in an alignment.

Given the table in Fig. 3.4, the alignment problem is finding the arrangement of amino acids resulting in the highest score summed over all positions. This is the basis of almost all commonly used alignment computer programs. One notable exception is software based on the so-called ***hidden Markov models*** (***HMMs***), which use probabilities rather than scores; however, these methods use a concept related to amino acid similarity.

For nucleotide sequences, it is much harder to distinguish conservative from non-conservative substitutions. In addition, two random sequences of equal base compositions will be 25% identical. As a consequence, it may be difficult to make a sensible alignment if the sequences have diverged significantly, especially if the nucleotide compositions are biased and/or if there are many repeats. This is one reason why, if there is a choice, *it is important to align protein coding sequences at the amino acid level.*

3.4 The problem of gaps

Insertions and deletions also accumulate as sequences diverge from each other. In proteins, these are concentrated between the main secondary structure elements,

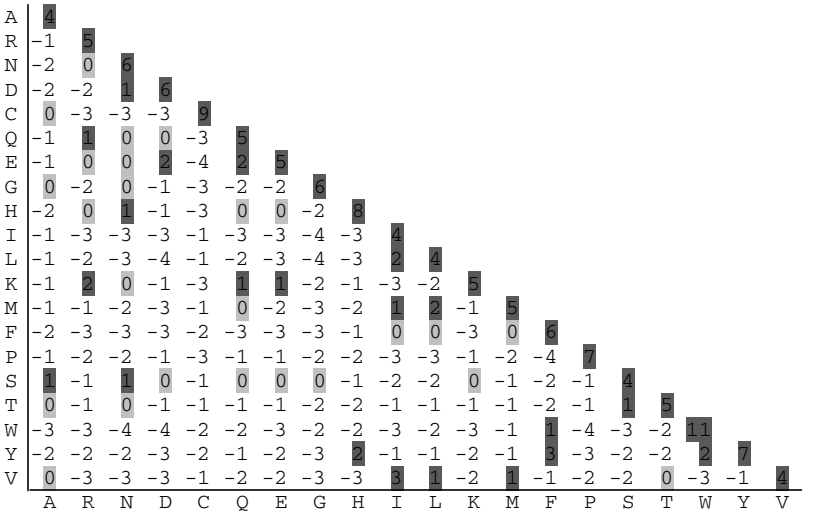


Fig. 3.4 The BLOSUM62 matrix. Similarity scores for all 20 × 20 possible pairs of amino acids, including identical pairs (the diagonal elements). Negative numbers represent pairs that are not commonly observed in real alignments; positive numbers represent commonly found pairs (shaded). These scores allow us to quantify the similarity of sequences in an alignment, when combined with gap scores (i.e. gap penalties).

just as for substitutions. Normally, no distinction is made between insertions and deletions, which are sometimes collectively referred to as *indels*. In alignments, they show up as gaps inserted in sequences in order to maximize an alignment score.

If there were no *indels*, the optimal alignment for sequences could be found by simply sliding one over the other. To choose the alignment with the highest score, the weight matrix scores are counted for each pair of aligned residues. With the presence of *indels*, a proper alignment algorithm is required to find the arrangement of gaps that maximizes the score. This is usually accomplished using a technique called ***dynamic programming*** (Needleman & Wunsch, 1970; Gotoh, 1982), which is an efficient method for finding the best alignment and its score. However, there is an additional complication: if gaps of any size can be placed at any position, it is possible to generate alignments with more gaps than residues. To prevent excessive use of gaps, *indels* are usually penalized (a penalty is subtracted from the alignment score) using so-called gap penalties (GPs). The most common formula for calculating GPs follows.

$$\mathrm{GP} = g + e(l - 1) \tag{3.1}$$

where l is the length of the gap, g is a gap opening penalty (charged once per gap) and e is a gap-extension penalty (charged once per hyphen in a gap). These penalties are often referred to as ***affine gap penalties***. The formula is quite flexible in that

```
VHLTPEEKSAVTALWGKVNVDEVGGEAL...
V-----------------NEEEVGGEAL...
```

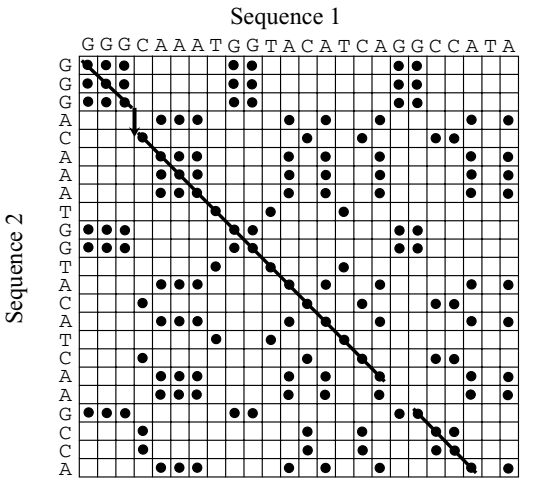
Fig. 3.5 A simple example to illustrate the problem of end gaps. The second sequence is missing a section from its N-terminus, and the end V aligns with the N-terminal V of the first sequence. This is clearly a nonsensical alignment, but it gets exactly the same score as the one with the V moved across to join the rest of the second sequence – unless end gaps are free (there is no GP for end gaps).

it allows the number and the lengths of gaps to be controlled separately by setting different values for *g* and *e*. However, there is no particular mathematical, statistical or biological justification for this formula. It is widely used because it works often well and it is straightforward to implement it in computer programs. In practice, the values of *g* and *e* are chosen arbitrarily, and there is no reason to believe that gaps simply evolve as the formula suggests. Significantly, the alignment with the highest alignment score may or may not be the correct alignment in a biological sense. Finally, it is common to make end gaps free (un-penalized), which takes into account that, for various biological and experimental reasons, many sequences are missing sections from the ends. Not making end gaps free risks getting nonsensical alignments, such as the one shown in Fig. 3.5.

3.5 Pairwise sequence alignment

3.5.1 Dot-matrix sequence comparison

Probably the oldest way to compare two nucleotide or amino acid sequences is a *dot matrix representation* or *dot plot*. In a *dot plot* two sequences are compared visually by creating a matrix where the columns are the character positions in sequence 1 and the rows are the character positions in sequence 2 (see Fig. 3.6). A dot is placed in location (*i*, *j*) if the *i*th character of sequence 1 matches the *j*th character of sequence 2. For nucleotide sequences, long protein sequences, or very divergent proteins, better resolution can be obtained by employing a sliding window approach to compare blocks of *N* residues (usually 5 to 20 amino acids or nucleotides). Two blocks are said to match if a mismatch level is not exceeded. For example, the user can look for DNA or amino acid stretches of length 10 (window size = 10) with the mismatch limit equal to 3, which means that a dot will be placed in the graph only if at least 7 out of 10 residues in the two 10-nucleotide stretches compared have identical residues. This procedure filters a considerable amount of noise that can be attributed to random matches of single residues. A high similarity region will show as a diagonal in the *dot plot*. This method can be used to explore similarity quickly in large genomic fragments, and thus evaluate homology among sequences, and can reveal features like internal repetitive elements when proteins



(a)

```
GGG-CAAATGGTACATCAGGCCATA
   *                *     **
GGGACAAATGGTACATCAAGCCA--
```

(b)

Fig. 3.6 (a). DNA dot plot representation with stringency 1 and window size 1 (see text). The nucleotides at each site of Sequence 1 and Sequence 2 are reported on the *x* (5′ → 3′ from left to right) and *y* axes (5′ → 3′ from top to bottom), respectively. Each nucleotide site of a sequence is compared with another sequence, and a dot is placed in the graph in case of identical residues. The path drawn by the diagonal line represents regions where the two sequences are identical; the vertical arrow indicates a deletion in Sequence 1 or an insertion in Sequence 2. (b) Sequence alignment resulting from following the path indicated in the dot plot representation.

are compared to themselves. However, their use has become somewhat obselete and we mainly discuss the method for historical and educational purposes.

3.5.2 Dynamic programming

For two sequences, dynamic programming can find the best alignment by scoring all possible pairs of aligned residues and penalizing gaps. This is a relatively rapid procedure on modern computers, requiring time and memory proportional to the product of the sequence lengths (N). This is usually referred to as an "N^2 algorithm" or as having complexity $O(N^2)$. In practice, unless the two sequences are enormous, pairwise alignment through dynamic programming can be achieved in a matter of seconds. Dynamic programming is based on *Bellman's principle of optimality*, which states that any subsolution of an optimal solution is itself an optimal solution. In Box 3.1, we use an amino acid example to illustrate how this principle can be used to find the best pairwise alignment.

Box 3.1 Dynamic programming

The dynamic programming algorithm guarantees us to find the optimal scoring alignment of two sequences without enumerating all possible solutions. The solution to the problem can be considered as the optimal scoring path in a matrix. The scoring system in our example is constituted by the BLOSUM62 (see Fig. 3.4) substitution matrix and a simple linear gap penalty g taking the value of −8. We will introduce affine gap penalties in a later example. A matrix for sequence X (GRQTAGL) and sequence Y (GTAYDL) filled with the relevant BLOSUM62 substitution scores and gap penalties is shown below:

Sequence X

Sequence Y	i / j	1	2	3	4	5	6	7	8
		*	G	R	Q	T	A	G	L
1	*	0	-8	-8	-8	-8	-8	-8	-8
2	G	-8	6	-2	-2	-2	0	6	-4
3	T	-8	-2	-1	-1	5	0	-2	-1
4	A	-8	0	5	-1	0	4	0	-1
5	Y	-8	-3	-2	-1	-2	-2	-3	-1
6	D	-8	-1	-2	0	-1	-2	-1	-4
7	L	-8	-4	-2	-2	-1	-1	-4	4

To allow for end gaps, a boundary condition is imposed through the use of a complete gap column ($i = 1$, denoted by *) and a complete gap row ($j = 1$). In a first step of the algorithm, we need to find optimal alignments for smaller subsequences. This can be achieved by finding the best scoring subpath for each element (i, j) in the matrix (F). For each element, we need to consider three options: a substitution event (X_i, Y_i) moving from (i−1, j−1) to (i, j), an insertion in X (or deletion in Y) moving from (i−1, j) to (i, j), and a deletion in X (or insertion in Y) moving from (i, j−1) to (i, j). Hence, the best subpath up to (i, j), will be determined by the score:

$$F(i, j) = \max \begin{cases} F(i-1, j-1) + s(X_i, Y_i), \\ F(i-1, j) - g, \\ F(i, j-1) - g \end{cases} \tag{3.2}$$

where $s(X_i, Y_i)$ is the score defined by the BLOSUM62 substitution matrix. We can apply this equation to fill the matrix recursively: column-wise from top left to bottom right.

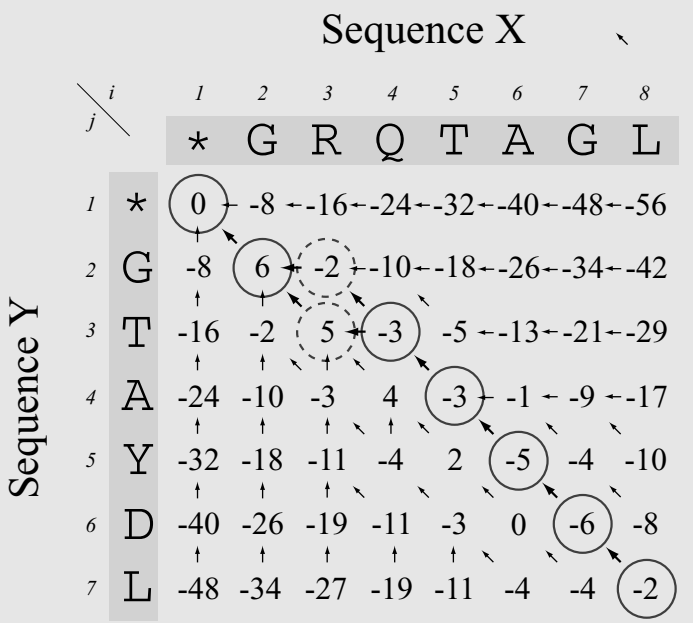
Importantly, we also will use a *pointer* to keep track of the movement in the matrix that resulted in the best score $F(i, j)$. For the first gap column ($i = 1$), only sequential gap events can be considered, reducing (3.2) to $F(i, j) = F(i, j-1) - g$. The same is true for the gap row ($j = 1$). The first element for which (1.2) needs to be fully evaluated is ($i = 2, j = 2$). For this element the best score is 6 as a result of:

$$F(2,2) = \max \begin{cases} F(1,1) + s(X_2, Y_2) = 6, \\ F(1,2) - 8 = -16, \\ F(2,1) - 8 = -16 \end{cases}$$

In this case, a pointer indicates a substitution event (X_i, Y_i) moving from ($i-1$, $j-1$) to (i, j). We continue by finding the best score for element ($i = 2, j = 3$), given by:

$$F(2,3) = \max \begin{cases} F(1,2) + s(X_2, Y_3) = -10, \\ F(1,3) - 8 = -16, \\ F(2,2) - 8 = -2 \end{cases}$$

In which case a pointer is kept to indicate a deletion in X (or insertion in Y) moving from (i, $j-1$) to (i, j). Repeating this procedure results in the matrix:



Note that for some elements, e.g. ($i = 4, j = 3$) two pointers are kept indicating paths that resulted in an equal score. Now that the matrix has been completed, we need to identify the best alignment based on all the pointers we have kept (represented by the arrows). We start from the score in the final element and follow the path indicated by the arrows, a procedure known as *traceback*. It is exactly here that we rely on Bellman's

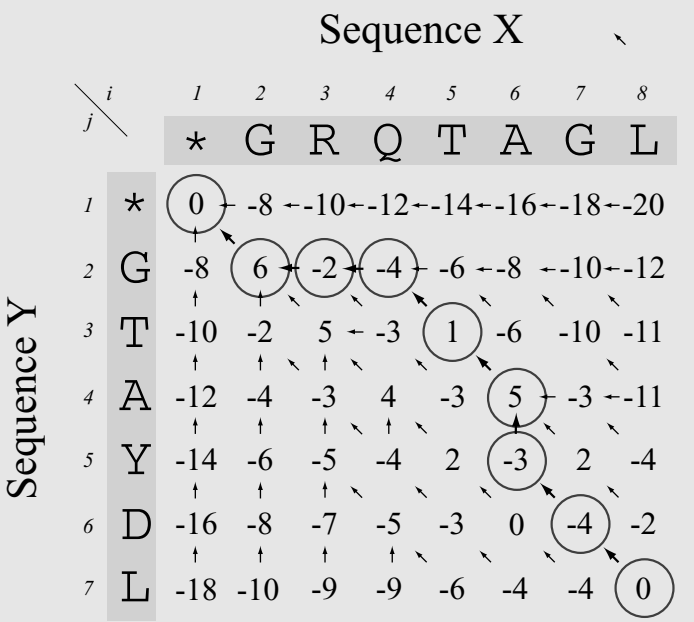
Box 3.1 (*cont.*)

optimality principle: if the final element reports the score for the best path, then the sub-solution leading to this element is also lying on the optimal path. So, if we are able to identify the optimal subpath leading to this element, than the preceding element is guaranteed to lie on the optimal path allowing us to continue the *traceback* procedure from this sub-solution. If two paths can be followed (for example, from ($i = 4$, $j = 3$) to ($i = 3$, $j = 3$) or to ($i = 3$, $j = 2$)), than an arbitrary choice must be made (indicated by the dashed circles). Each step in the *traceback* procedure can be associated with the assembly of two symbols to construct the alignment: adding symbol X_i and Y_j if the step was to ($i-1$, $j-1$), X_i and a gap if the step was to ($i-1$, j), or a gap and Y_j if the step was to (i, $j-1$). Hence, the alignment is constructed from right to left. Depending on the choice made at ($i = 4$, $j = 3$) in our example, this assembly results in the alignment:

```
GRQTAGL   or   GRQTAGL
GT-AYDL        G-TAYDL
```

It is not surprising that these alignments have an equal score since both proposed amino acid substitutions (R↔T) and (Q↔T) are equally penalized by BLOSUM62 (Fig. 3.4).

Since gaps are often observed as blocks in sequence alignments, a simple linear gap penalty will generally be biologically unreasonable in real-world data. Therefore, affine gap penalties (see 3.1) have been the preferred choice for penalizing gaps in many algorithms. Here, we will extend our example using a gap-opening penalty of −8 and a gap extension penalty of −2. In this case, the matrix recursively completed with scores for the optimal subpath to each element is given by:



Note that the use of affine gap penalties significantly reduces the cost for sequential gaps (e.g. for column $i = 1$). Affine gap penalties introduce an additional complexity of having to keep track of multiple values. To evaluate the cost for a subpath that involves inserting a gap, we need to consider that the previous step may also have required the insertion of a gap. In this case, a gap extension needs to be penalized; in the other case, we need to penalize a gap opening. Interestingly, this scoring scheme results in a different optimal path, with which only a single alignment can be associated:

```
GRQTA-GL
G--TAYDL
```

This example illustrates that different scoring schemes, for which the gap penalties values are usually chosen arbitrarily, can result in different alignments. Therefore, it is generally recommended to explore how changing default values in alignment programs influences the quality of the sequence alignments.

3.6 Multiple alignment algorithms

The dynamic programming approach described in Box 3.1 can easily be generalized to more than two sequences. In this case, the alignment maximizing the similarity of the sequences in each column (using an amino acid weight matrix as usual) is found while allowing some minimal number and lengths of gaps. The task is most commonly expressed as finding the alignment that gives the best score for the following formula, called the *weighted sum of pairs* or `WSP` *objective function*:

$$\sum_i \sum_j W_{ij} D_{ij} \tag{3.3}$$

For any multiple alignment, a score between each pair of sequences (D_{ij}) is calculated. Then, the `WSP` function is simply the sum of all of these scores, one for each possible pair of sequences. There is an extra weight term W_{ij} for each pair, that is, by default, always equal to one, but enables weighting some pairs more than others. This can be extremely useful to give more weight to pairs that are more reliable or more important than others. Alternatively, it can be used to give less weight to sequences with many close relatives because these are overrepresented in the data set. Although the weighting can be inspired by evolutionary relationships, the WSP fails to fully exploit phylogeny and does not incorporate an evolutionary model (Edgar & Batzoglou, 2006). Dynamic programming can be used to find a multiple alignment that gives the best possible score for the `WSP` function. Unfortunately, the time and memory required grows exponentially with the number of sequences as the complexity is $O(N^M)$, where M is the number of sequences and N is the sequence length. This quickly becomes impossible to compute for more than four sequences of even modest lengths.

One elegant solution to the computational complexity came from the MSA program of Lipman *et al.* (1989). It used a so-called ***branch-and-bound*** technique to eliminate many unnecessary calculations making it possible to compute the WSP function for five to eight sequences. The precise number of sequences depended on the length and similarity; the more similar the sequences, the faster the calculation and the larger the number of sequences that could be aligned. Although this is an important program, its use is still limited by the small number of sequences that can be managed. In tests with BALIBASE (see below for more information on testing multiple alignment methods), this program performs extremely well although it is not able to handle all test cases (any with more than eight sequences). The FASTMSA program, a highly optimized version of MSA, is faster and uses less memory, but it is still limited to small data sets.

In the following paragraphs, we discuss several heuristics or approximate alternatives for multiple sequence alignment and some of their most popular implementations. A comprehensive listing of multiple sequence programs and their availability can be found in Table 3.1. An updated version of this table with updated links to download pages and servers running web-based applications will be maintained on the website accompanying this book (*www.thephylogenetichandbook.org*).

3.6.1 Progressive alignment

Multiple alignments are the necessary prerequisite for phylogenetic analysis. Conversely, if the phylogenetic relationships in a set of sequences were known, this information could be useful to generate an alignment. Indeed, this mutual relationship was the basis of an early multiple alignment method (Sankoff, 1985) which simultaneously generated the tree and alignment; unfortunately, the particular method is too complex for routine use. One very simple shortcut is to make a quick and approximate tree of the sequences and use this to make a multiple alignment, an approach first suggested by Hogeweg and Hesper (1984). The method is heuristic in a mathematical sense insofar as it makes no guarantees to produce an alignment with the best score according to the optimality criterion. Nonetheless, this method is extremely important because it is the way the vast bulk of automatic alignments are generated. As judged by test cases, it performs very well, although not quite as well as those methods that use the WSP objective function. This lack of sensitivity is only visible with the most difficult test cases; for similar sequences, progressive alignment is perfectly adequate. The sheer speed of the method, however, and its great simplicity, make it extremely attractive for routine work.

A phylogenetic tree showing the relatedness of the sequences in Fig. 3.1, is shown in Fig. 3.7. A similar tree can be generated quickly by making all possible pairwise alignments between all the sequences and calculating the observed distance

Table 3.1 Multiple alignment programs

Software	Reference and URL*	Description
ABA	Raphael *et al.*, 2004 *http://aba.nbcr.net/*	A-Bruijn Aligner (ABA) is a method that represents an alignment as a directed graph, which can contain cycles accommodating repeated and/or shuffled elements
ALIGN-M	Van Walle, Lasters, & Wyns, 2004 *http://bioinformatics.vub.ac.be/software/software.html*	ALIGN-M will not align parts with low confidence, thereby focusing on specificity which is important for sequences with low similarity
ALLALL	Korostensky & Gonnet, 1999 *http://www.cbrg.ethz.ch/services/AllvsAll*	Starting from a set of related peptides, ALLALL determines the relationship of each peptide sequence versus the others and uses these results to construct multiple alignments
AMAP	Schwartz & Pachter, 2007 *http://bio.math.berkeley.edu/amap/* *http://baboon.math.berkeley.edu/mavid/*	AMAP uses a sequence annealing algorithm, which is an incremental method for building multiple sequence alignments one match at a time
BALIPHY	Suchard & Redelings, 2006 *http://www.biomath.ucla.edu/msuchard/baliphy/index.php*	BALI-PHY performs joint Bayesian estimation of alignment and phylogeny. The program considers near-optimal alignments when estimating the phylogeny
CHAOS-DIALIGN	Brudno & Morgenstern, 2002 *http://dialign.gobics.de/* *http://dialign.gobics.de/chaos-dialign-submission*	The local alignment tool CHAOS rapidly identifies chains of pairwise similarities, which are subsequently used as anchor points to speed up the multiple-alignment program
CLUSTALW	Thompson, Higgins, & Gibson, 1994 *ftp://ftp-igbmc.u-strasbg.fr/pub/ClustalW/* *http://www.ebi.ac.uk/clustalw/*	CLUSTALW is the prototypical progressive alignment program and the most widely used multiple alignment tool
COBALT	Papadopoulos & Agarwala, 2007 *ftp://ftp.ncbi.nlm.nih.gov/pub/agarwala/cobalt*	COBALT can combine a collection of pairwise constraints derived from database searches, sequence similarity, and user input, and incorporates them into a progressive multiple alignment
CONTRAALIGN	Do, Woods, & Batzoglou, 2006 *http://contra.stanford.edu/contralign/* *http://contra.stanford.edu/contralign/server.html*	CONTRAALIGN uses discriminative learning techniques, introduced in machine learning literature, to perform multiple alignment

(*cont.*)

Table 3.1 (*cont.*)

Software	Reference and URL*	Description
DCA	Stoye, 1998; Stoye, Moulton, & Dress, 1997 *http://bibiserv.techfak.uni-bielefeld.de/dca/* *http://bibiserv.techfak.uni-bielefeld.de/dca/submission.html*	The DCA (divide-and-conquer) algorithm is a heuristic approach to sum-of-pairs (SP) optimal alignment
DIALIGN	Morgenstern, 2004 *http://bibiserv.techfak.uni-bielefeld.de/dialign* *http://bibiserv.techfak.uni-bielefeld.de/dialign/submission.html*	DIALIGN does not use gap penalties and constructs pairwise and multiple alignments by comparing whole segments of the sequences, an efficient approach when sequences share only local similarities
DIALIGN-T	Subramanian *et al.*, 2005 *http://dialign-t.gobics.de/* *http://dialign-t.gobics.de/submission?type = protein*	DIALIGN-T is a reimplementation of the segment-based ("local") alignment approach of DIALIGN with better heuristics
EXPRESSO (3DCOFFEE)	Armougom *et al.*, 2006; O'Sullivan *et al.*, 2004 *http://www.tcoffee.org/*	EXPRESSO is a webserver that runs BLAST to retrieve PDB structures of close homologues of the input sequences, which are used as templates to guide the alignment
ITERALIGN	Brocchieri & Karlin, 1998 *http://giotto.stanford.edu/~luciano/iteralign.html.* Currently in the process of moving ITERALIGN to a faster more reliable server	A symmetric-iterative method for multiple alignment that combines motif finding and dynamic programming procedures. The method produces alignment blocks that accommodate indels and are separated by variable-length unaligned segments
KALIGN	Lassmann & Sonnhammer, 2005 *http://msa.cgb.ki.se*	KALIGN employs the Wu-Manber string-matching algorithm, to improve both the accuracy and speed of multiple sequence alignment
MACAW	Schuler, Altschul, & Lipman, 1991 Can be downloaded at: *http://genamics.com/software/downloads/macawnta.exe*	The Multiple Alignment Construction & Analysis Workbench (MACAW) is an interactive program that allows the user to construct multiple alignments by locating, analyzing, editing, and combining "blocks" of aligned sequence segments
MAFFT	Katoh *et al.*, 2005; Katoh *et al.*, 2002 *http://align.bmr.kyushu-u.ac.jp/mafft/software/* *http://align.bmr.kyushu-u.ac.jp/mafft/online/server/*	MAFFT offers a range of multiple alignment methods including iterative refinement and consistency-based scoring approaches, e.g. L-INS-i (accurate; recommended for <200 sequences), FFT-NS-2 (fast; recommended for >2000 sequences)

Map	Huang, 1994 *http://genome.cs.mtu.edu/map.html*	The Map program computes a global alignment of sequences using an iterative pairwise method: two sequences are aligned by computing a best overlapping alignment without penalizing terminal gaps and without heavily penalizing long internal gaps in short sequences
Match-Box	Depiereux & Feytmans, 1992 *http://www.fundp.ac.be/sciences/biologie/bms/matchbox_submit.shtml*	Match-Box software performs protein sequence multiple alignment that does not require a gap penalty and is particularly suitable for finding and aligning conserved structural motivfs
Mavid/Amap	Bray & Pachter, 2004 *http://baboon.math.berkeley.edu/mavid/*	Mavid is a progressive-alignment approach incorporating maximum-likelihood inference of ancestral sequences, automatic guide-tree construction, protein-based anchoring of ab-initio gene predictions, and constraints derived from a global homology map of the sequences
Msa	Lipman, Altschul, & Kececioglu, 1989 *http://www.ncbi.nlm.nih.gov/CBBresearch/Schaffer/msa.html*	Msa uses the *branch-and-bound* technique to eliminate many unnecessary calculations making it possible to compute the WSP function for a limited number of sequences
Multalin	Corpet, 1988 download: *ftp://ftp.toulouse.inra.fr/pub/multalin/* *http://bioinfo.genopole-toulouse.prd.fr/multalin/*	Multalin is a conventional dynamic-programming method of pairwise alignment and hierarchical clustering of the sequences using the matrix of the pairwise alignment scores
Multi-Lagan	Brudno *et al.*, 2003 *http://lagan.stanford.edu/lagan_web/index.shtml*	Multi-Lagan has been developed to generate multiple alignments of long genomic sequences at any evolutionary distance
Mummals	Pei & Grishin, 2006 *http://prodata.swmed.edu/mummals/mummals.php*	Mummals uses probabilistic consistency, like ProbCons, but incorporates complex pairwise alignment HMMs and better estimation of HMM parameters
Murlet	Kiryu *et al.*, 2007 *http://www.ncrna.org/papers/Murlet/*	Murlet is an alignment tool for structural RNA sequences using a variant of the Sankoff algorithm and an efficient scoring system that reduces time and space requirements
Musca	Parida, Floratos, & Rigoutsos, 1998 *http://cbcsrv.watson.ibm.com/Tmsa.html*	An algorithm for constrained alignment of multiple sequences that identifies two relatively simpler sub-problems whose solutions are used to obtain the alignment of the sequences

(*cont.*)

Table 3.1 (*cont.*)

Software	Reference and URL*	Description
Muscle	Edgar, 2004a; Edgar, 2004b *http://www.drive5.com/muscle/* *http://phylogenomics.berkeley.edu/cgi-bin/muscle/input_muscle.py*	Muscle is multiple alignment software that includes fast distance estimation using Kmer counting, progressive alignment using the log-expectation score, and refinement using tree-dependent restricted partitioning
Pcma	Pei, Sadreyev, & Grishin, 2003 *ftp://iole.swmed.edu/pub/PCMA/*	Pcma performs fast progressive alignment of highly similar sequences into relatively divergent groups, which are then aligned based on profile-profile comparison and consistency
PileUp	Based on Feng & Doolittle, 1987 implemented in Gcg: *http://www.gcg.com*	PileUp creates a multiple sequence alignment from a group of related sequences using a simplification of the progressive alignment method of Feng & Doolittle (1987)
Pima	Smith & Smith, 1992 Download: *http://genamics.com/software/downloads/* *http://searchlauncher.bcm.tmc.edu/multi-align/Options/pima.html*	Pima performs multiple alignment using an extension of the covering pattern construction algorithm. It performs all pairwise comparisons and clusters the resulting scores. Each cluster is multiply aligned using a pattern-based alignment algorithm
Poa	Lee, Grasso, & Sharlow, 2002 *http://bioinfo.mbi.ucla.edu/poa2/*	Poa uses a graph representation of a multiple sequence alignment (MSA), called a POA-MSA, that can itself be aligned directly by pairwise dynamic programming, eliminating the need to reduce the MSA to a profile
Praline	Simossis & Heringa, 2005 *http://zeus.cs.vu.nl/programs/pralinewww/*	Praline is a multiple sequence alignment program that implements global or local preprocessing, predicted secondary structure information and iteration capabilities
Prime	Yamada, Gotoh, & Yamana, 2006 *http://prime.cbrc.jp/*	Prime is a group-to-group sequence alignment algorithm with piecewise linear gap cost, instead of traditional affine gap cost
Probalign	Roshan & Livesay, 2006 *http://www.cs.njit.edu/usman/probalign*	Probalign combines amino acid posterior probability estimation using partition function methods and computation of maximal expected accuracy alignment
ProbCons	Do *et al.*, 2005 *http://probcons.stanford.edu/*	ProbCons uses a combination of probabilistic modeling and consistency-based alignment techniques and, according to the original authors, performs very well in terms of accuracy

ProDA	Phuong *et al.*, 2006 *http://proda.stanford.edu/*	ProDA has been developed to identify and align all homologous regions appearing, not necessarily colinear, in a set of sequences
Promals	Pei & Grishin, 2007 *http://prodata.swmed.edu/promals/promals.php*	Promals uses probabilistic consistency-based scoring applied to progressive alignment in combination with profile information from database searches and secondary structure prediction
Prrn/Prrp	Based on Gotoh, 1996 *http://prrn.hgc.jp/* *http://prrn.ims.u-tokyo.ac.jp/*	Prrn/Prrp programs are prototypical iterative refinement tools using a doubly nested randomized iterative (DNR) method to make alignment, phylogenetic tree and pair weights mutually consistent
PSAlign	Sze, Lu, & Yang, 2006 *http://faculty.cs.tamu.edu/shsze/psalign/*	PSAlign uses consistency-based pairwise alignment, like the first stage of the programs TCoffee or ProbCons, but replaces the second heuristic progressive step by an exact preserving alignment step
Qoma	Sze, Lu, & Yang, 2006 *http://www.cise.ufl.edu/~tamer/other_files/msa.html*	Quasi-Optimal Multiple Alignment uses a graph-based method to represent an initial alignment and performs local improvements in the SP score using a greedy algorithm
Refiner	Chakrabarti *et al.*, 2006 *ftp://ftp.ncbi.nih.gov/pub/REFINER*	A method to refine multiple sequence alignments by iterative realignment of its individual sequences with a predetermined conserved core model of a protein family
Saga	Notredame & Higgins, 1996 *http://www.tcoffee.org/Projects_home_page/saga_home_page.html*	Saga is based on the WSP objective function but uses a genetic algorithm to find the best alignment. This stochastic optimisation technique grows a population of alignments and evolves it over time using a process of selection and crossing
Sam	Hughey & Krogh, 1996; Krogh *et al.*, 1994 *http://www.cse.ucsc.edu/research/compbio/sam.html*	The Sequence Alignment and Modeling system (Sam) is a collection of software tools for creating, refining, and using linear hidden Markov models for biological sequence analysis

(*cont.*)

Table 3.1 (*cont.*)

Software	Reference and URL*	Description
SATCHMO	Edgar & Sjolander, 2003 *http://www.drive5.com/lobster/index.htm* *http://phylogenomics.berkeley.edu/cgi-bin/satchmo/input_satchmo.py*	SATCHMO simultaneously constructs a tree and multiple sequence alignments for each internal node of the tree. These alignments of subsets predict which positions are alignable and which are not
SPEM	Zhou & Zhou, 2005 *http://sparks.informatics.iupui.edu/* *http://sparks.informatics.iupui.edu/Softwares-Services_files/spem.htm*	SPEM builds profiles and uses secondary structure information for pairwise alignment. These pairwise alignments are refined using consistency-based scoring and form the basis of a progressive multiple alignment
T-COFFEE	Notredame, Higgins, & Heringa, 2000 *http://www.tcoffee.org/Projects_home_page/t_coffee_home_page.html* *http://www.tcoffee.org/*	Tree-based Consistency objective function for alignment evaluation is the prototypical consistency-based alignment method capable of combining a collection of multiple/pairwise, global/local alignments into a single one
TRACKER/Anchored DIALIGN	Morgenstern *et al.*, 2006 *http://dialign.gobics.de/anchor/index.php* *http://dialign.gobics.de/anchor/submission.php*	Tracker is a semi-automatic version of the alignment program DIALIGN that can take predefined constraints into account and uses these as 'anchor points' in the alignment

When two URLs are provided, the first one represents the link for download or the link to the home page, whereas the second one is a link to a webserver.

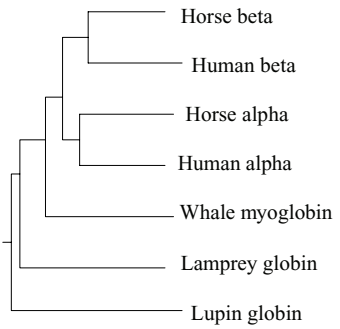


Fig. 3.7 A rooted tree showing the possible phylogenetic relationships between the seven globin sequences in Fig. 3.1. Branch lengths are drawn to scale.

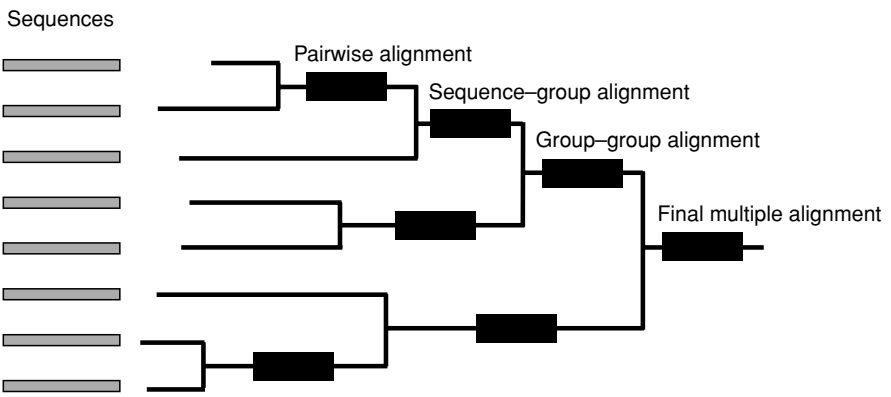


Fig. 3.8 Schematic representation of progressive alignment. This procedure begins with the aligning the two most closely related sequences (pairwise alignment) and subsequently adds the next closest sequence or sequence group to this initial pair (sequence–group or group–group alignment). This process continues in an iterative fashion along the guide tree, adjusting the positioning of indels in all grouped sequences.

(proportion of residues that differ between the two sequences) in each case. Such distances are used to make the tree with one of the widely available distance methods such as the ***neighbor-joining*** **(NJ)** method of Saitou and Nei (1987) (see Chapter 5). NJ trees, for example, can be calculated quickly for as many as a few hundred sequences. Next, the alignment is gradually built up by following the branching order in the tree (Fig. 3.8). The two closest sequences are aligned first using dynamic programming with GPs and a weight matrix (Fig. 3.8). For further alignment, the two sequences are treated as one, such that any gaps created between the two cannot be moved. Again, the two closest remaining sequences or pre-aligned groups of sequences are aligned to each other (Fig. 3.8). Two unaligned sequences or two sub-alignments can be aligned or a sequence can added to a sub-alignment, depending

on which is more similar. The process is repeated until all the sequences are aligned. Once the initial tree is generated, the multiple alignment can be carried out with only *N*-1 separate alignments for *N* sequences. This process is fast enough to align many hundreds of sequences.

CLUSTAL

The most commonly used software for progressive alignment is **CLUSTALW** (Thompson *et al.*, 1994) and **CLUSTALX** (Thompson *et al.*, 1997). These programs are freely available as source code and/or executables for most computer platforms (Table 3.1). They can also be run using servers on the Internet at a number of locations. These programs are identical to each other in terms of alignment method but offer either a simple text-based interface (**CLUSTALW**) suitable for high-throughput tasks or a graphical interface (**CLUSTALX**). In the discussion that follows, we will refer only to **CLUSTALW** but all of it applies equally to **CLUSTALX**. **CLUSTALW** will take a set of input sequences and carry out the entire progressive alignment procedure automatically. The sequences are aligned in pairs in order to generate a distance matrix that can be used to make a simple initial tree of the sequences. This guide tree is stored in a file and is generated using the Neighbor-Joining method of Saitou and Nei (1987). This produces an unrooted tree (see Chapter 1), which is used to guide the multiple alignment. Finally, the multiple alignment is carried out using the progressive approach, as described above.

CLUSTALW has specific features which help it make more accurate alignments. First, sequences are down-weighted according to how closely related they are to other sequences (as judged by the guide tree). This is useful because it prevents large groups of similar sequences from dominating an alignment. Secondly, the weight matrix used for protein alignments varies, depending on how closely related the next two sequences or sets of sequences are. One weight matrix can be used for closely related sequences that gives high scores to identities and low scores otherwise. For very distantly related sequences, the reverse is true; it is necessary to give high scores to conservative amino acid matches and lower scores to identities. **CLUSTALW** uses a series of four matrices chosen from either the BLOSUM, or PAM series (Henikoff & Henikoff, 1992 and Dayhoff *et al.*, 1978, respectively). During alignment, the program attempts to vary GPs in a sequence- and position-specific manner, which helps to align sequences of different lengths and different similarities. Position-specific GPs are used in an attempt to concentrate gaps in the loops between the secondary structure elements, which can be set either manually using a mask or automatically. In the latter case, GPs are lowered in runs of hydrophilic residues (likely loops) or at positions where there are already many gaps. They are also lowered near some residues, such as Glycine, which are known to be common near gaps from empirical analysis (Pascarella & Argos, 1992). GPs are raised adjacent

to existing gaps and near certain residues. These and other parameters can be set by the user before each alignment. Although **CLUSTALW** is still the most popular alignment tool, several more recent methods have now been shown to perform better in terms of accuracy and/or speed.

3.6.2 Consistency-based scoring

Progressive alignment is fast and simple, but it does have one obvious drawback: a local-minimum problem. Any alignment errors (i.e. misaligned residues or whole domains) that occur during the early alignment steps cannot be corrected later as more data are added. This may be due to an incorrect tree topology in the guide tree, but it is more likely due to simple errors in the early alignments. The latter occurs when the alignment with the best score is not the best one biologically or not the best if one considers all of the sequences in a data set. An effective way to overcome this problem is to use "consistency-based" scoring (Kececioglu, 1992), which is based on the `WSP` function and has been successfully used as an improvement over progressive alignment (Notredame *et al.*, 1998). Consistency-based alignment techniques use intermediate sequence information to improve the quality of pairwise comparisons and search for the multiple sequence alignment that maximizes the agreement with a set of pairwise alignments computed for the input sequences (such a procedure acknowledges that the pairwise alignments A–B and B–C may imply an A–C alignment that is different from the alignment directly computed for A and C) (Do *et al.*, 2004; Edgar & Batzoglou, 2006).

T-COFFEE

Tree-based consistency objective function for alignment evaluation (**T-COFFEE**) is the prototypical consistency-based alignment method (Notredame *et al.*, 2000). Although **T-COFFEE** is relatively slow, it generally results in more accurate alignments than **CLUSTALW, PRRP** (Gotoh, 1996), and **DIALIGN** (Morgenstern *et al.*, 1996; Morgenstern, 2004) when tested on **BALIBASE** (see below). Indeed, this increase in accuracy is most pronounced for difficult test cases and is found across all of the **BALIBASE** test sets. The method is based on finding the multiple alignment that is most consistent with a set of pairwise alignments between the sequences. These pairwise alignments can be derived from a mixture of sources such as different alignment programs or from different types of data such as structure superpositions and sequence alignments. These are processed to find those aligned pairs of residues in the initial data set which are most consistent across different alignments. This information is used to compile data on which residues are most likely to align in which sequences. Finally, this information is used to build up the multiple alignment using progressive alignment, a fast and simple procedure

that requires no other parameters (GPs or weight matrices). The main disadvantage of T-Coffee over ClustalW is that the former is much more computationally demanding and cannot handle large alignment problems as can the latter.

3.6.3 Iterative refinement methods

The pioneering work on alignment algorithms by Gotoh eventually resulted in the Prrn and Prrp programs (now unified to "ordinary" or "serial" Prrn) (Gotoh 1995, 1996 and 1999). Prrn uses an iterative scheme to gradually work toward the optimal alignment. At each iteration cycle, the sequences are randomly split into two groups. Within each group, the sequences are kept in fixed alignment, and the two groups are then aligned to each other using dynamic programming. This cycle is repeated until the score converges. The alignments produced by Prrn are excellent, as judged by performance using BaliBase and other test cases (Notredame *et al.*, 1998; Thompson *et al.*, 1999). The program is, however, relatively slow with more than 20 sequences and not widely adopted.

Recently, more efficient implementations of iterative refinement methods have been developed under the name of Mafft and Muscle (Katoh *et al.*, 2002, 2005; Edgar, 2004a,b). These have the same basic strategy of building a progressive alignment, to which horizontal refinement subsequently is applied. Mafft applies the "fast Fourier Transform" to rapidly detect homologous regions and also uses an improved scoring system. Muscle allows fast distance estimation using "*k*-mer counting," progressive alignment using a novel profile function, and refinement using "tree-dependent restricted partitioning." Both programs are fine-tuned towards high-throughput applications, offering significant improvements in scalability while still performing relatively accurately. It should be noted that recent versions of Mafft (v.5 or higher) implement variants of the algorithm with an objective function combining the WSP score and Coffee-like score, which could be classified under consistency-based alignment.

3.6.4 Genetic algorithms

The Saga program (Notredame & Higgins, 1996) is based on the WSP objective function but uses a ***genetic algorithm*** instead of dynamic programming to find the best alignment. This stochastic optimization technique grows a population of alignments and evolves it over time using a process of selection and crossing to find the best alignment. Comparisons with the performance of Msa, which implements the ***branch-and-bound*** technique, suggest that Saga can find the optimal alignment in terms of the WSP function. The advantage of Saga, however, is its ability to deliver good alignments for more than eight sequences; the disadvantage is that it is still relatively slow, perhaps taking many hours to compute to a good alignment for 20 or 30 sequences. The program must also be run several times because the

stochastic nature of the algorithm does not guarantee the same results for different runs. Despite these reservations, SAGA has proven useful because it allows the testing of alternative WSP scoring functions.

3.6.5 Hidden Markov models

An interesting approach to alignment is the so-called hidden Markov models or HMMs, which are based on probabilities of residue substitution and gap insertion and deletion (Krogh *et al.*, 1994). HMMs have been shown to be extremely useful in a wide variety of situations in computational molecular biology, such as locating introns and exons or predicting promoters in DNA sequences. HMMs are also very useful for summarizing the diversity of information in an existing alignment of sequences and predicting whether new sequences belong to the family (e.g. Eddy, 1998; Bateman *et al.*, 2000). Packages are available that simultaneously generate such an HMM and find the alignment from unaligned sequences. These methods used to be inaccurate (e.g. see results in Notredame *et al.*, 1998); however, some progress has been made and the SAM method (Hughey & Krogh, 1996) is now roughly comparable to CLUSTALW in accuracy, although not as easy or as fast to use.

3.6.6 Other algorithms

The DIVIDE AND CONQUER, DCA (Stoye *et al.*, 1997), program also computes alignments according to the WSP scoring function. The algorithm finds sections of alignment, which when joined together head to tail, will give the best alignment. Each section is found using the MSA program and so, ultimately, it is limited in the number of sequences it can handle, even if more than MSA.

Consistency-based progressive alignment methodology has been combined with probabilistic HMMs to account for suboptimal alignments (Do *et al.*, 2004). The PROBCONS program implements this approach called posterior probability-based scoring (Durbin *et al.*, 1998), with additional features such as unsupervised expectation-maximization parameter training. This makes the approach highly accurate, but unfortunately, also computationally expensive, limiting practical application to data sets of less than 100 sequences. More complex pairwise alignment HMMs that incorporate local structural information and better estimation of HMM parameters have recently been implemented in MUMMALS (Pei & Grishin, 2006).

All methods described above try to globally align sequences, which "forces" the alignment to span the entire length of all query sequences. Recently developed methods, like ALIGN-M (Van Walle *et al.*, 2004), DIALIGN (Morgenstern *et al.*, 1998; Morgenstern, 1999; Subramanian *et al.*, 2005), POA (Lee *et al.*, 2002; Grasso & Lee, 2004) and SATCHMO (Edgar & Sjölander, 2003), have relaxed the constraint

of global alignability to make possible the alignment of sequences highly variable in length or sequences with different domain structure. The DIALIGN method of Morgenstern (1999) is based on finding sections of local multiple alignment in a set of sequences; that is, sections similar across all sequences, but relatively short. DIALIGN will therefore be useful if the similarity is localized to isolated domains. In practice, the algorithm performed well in two of the BALIBASE test sets: those with long insertions and deletions; otherwise, CLUSTALW easily outperforms DIALIGN. Although DIALIGN allows unalignable regions, the alignable domains must still appear in the same order in the sequences being compared. To overcome this problem, PRODA has recently been developed to identify and align all homologous regions appearing, not necessarily colinear, in a set of sequences (Phuong *et al.*, 2006).

3.7 Testing multiple alignment methods

The most common way to test an alignment method is by assessing its performance on test cases. Exaggerated claims have been made about how effective some methods are or why others should be avoided. The user-friendliness of the software and its availability are important secondary considerations, but ultimately, it is the quality of the alignments that matters most. In the past, many new alignment programs have been shown to perform better than older programs, using one or two carefully chosen test cases. The superiority of the new method is then often verbally described by authors who selected particular parameter values (i.e. GPs and weight matrix) that favor the performance of their program. Of course, authors do not choose test cases for which their programs perform badly and, to be fair, some underappreciated programs perform well on some test cases and are still potentially useful.

One solution is to use benchmark data sets, generated by experts, with extensive reference to secondary and tertiary structural information. For structural RNA sequences, the huge alignments of ribosomal RNA sequences can be used as tests; these comparisons, however, are difficult and not necessarily generalizable. For proteins, an excellent collection of test cases is called BALIBASE (Thompson *et al.*, 1999), to which we have referred to in the description of some programs. BALIBASE is a collection of 141 alignments representing five different types of alignment situations: (1) equidistant (small sets of phylogenetically equidistant sequences); (2) orphan (as for Type 1 but with one distant member of the family); (3) two families (two sets of related sequences, distantly related to each other); (4) long insertions (one or more sequences have a long insertion); (5) long deletions. These test cases are made from sets of sequences for which there is extensive tertiary-structure information. The sequences and structures were carefully compared and multiple alignments were produced by manual editing.

The use of these test cases involves several assumptions. Although contradictory scenarios can be envisaged, the first assumption is that an alignment maximizing structural *similarity* is somehow "correct." A more serious assumption is that the reference alignment has been made correctly. Although parts of alignments will be difficult to judge, even with tertiary information, this can be largely circumvented by selecting and focusing only on those regions of core alignment from each test case that are clearly and unambiguously aligned. More recently, several new benchmarks have been proposed, including OXBENCH (Raghava *et al.*, 2003), PREFAB (Edgar, 2004a), SABmark (Van Walle *et al.*, 2005), IRMBASE (Subramanian *et al.*, 2005), and an extended version of **BaliBase** (BaliBase 3, available at *http://www-bio3d-igbmc.u-strasbg.fr/balibase/*). These new test cases have generally been constructed by automatic means implying that the overall quality and accuracy of an individual alignment cannot be guaranteed. However, averaging accuracy scores over a large set of such alignments can still result in a meaningful ranking of multiple alignment tools.

3.8 Which program to choose?

Validation of multiple alignment programs using benchmark data sets provides useful information about their biological accuracy. Although this generally is of major concern, it is not the only criterion for choosing a particular software tool. Execution time and memory usage, in particular, can be limiting factors in practice. This is the main drawback of programs that achieve high accuracy, like **T-Coffee** and **ProbCons**. It is interesting to note that **ClustalW** is still relatively memory efficient compared with modern programs.

As a guideline to choosing alignment programs, we follow here the recommendations made by Edgar and Batzoglou (2006), which are summarized in Table 3.2. Because multiple alignment is an ongoing and active research field, these recommendations can evolve relatively rapidly. If accuracy is the only concern, which is generally the case for limited size data sets, it is recommended to apply different alignment methods and compare the results using the **AltAVisT** web server (Morgenstern, 2003).

Incorporation of structural information in protein sequence alignment can provide significant improvement in alignment accuracy. This has inspired the development of programs that use such information in an automated fashion, like **3DCoffee** (O'Sullivan *et al.*, 2004). The **3DCoffee** algorithm has been implemented in a webserver called **Expresso** that only requires input sequences (Armougom *et al.*, 2006). **Expresso** automatically runs **Blast** to identify close homologs of the sequences within the PDB database (see Chapter 2). These PDB structures are used as templates to guide the alignment of the original sequences using structure-based sequence alignment methods. Homology

Table 3.2 Typical alignment tasks and recommended procedures

Input data	Recommendations
2–100 sequences of typical protein length (maximum around 10 000 residues) that are approximately globally alignable	Use PROBCONS, T-COFFEE, and MAFFT or MUSCLE, compare the results using ALTAVIST. Regions of agreement are more likely to be correct. For sequences with low percent identity, PROBCONS is generally the most accurate, but incorporating structure information (where available) via 3DCOFFEE (a variant of T-COFFEE) can be extremely helpful
100–500 sequences that are approximately globally alignable	Use MUSCLE or one of the MAFFT scripts with default options. Comparison using ALTAVIST is possible, but the results are hard to interpret with larger numbers of sequences unless they are highly similar
>500 sequences that are approximately globally alignable	Use MUSCLE with a faster option (we recommend maxiters-2) or one of the faster MAFFT scripts
Large numbers of alignments, high-throughput pipeline	Use MUSCLE with faster options (e.g. maxiters-1 or maxiters-2) or one of the faster MAFFT scripts
2–100 sequences with conserved core regions surrounded by variable regions that are not alignable	Use DIALIGN
2–100 sequences with one or more common domains that may be shuffled, repeated or absent	Use PRODA
A small number of unusually long sequences (say, >20 000 residues)	Use CLUSTALW. Other programs may run out of memory, causing an abort (e.g. a segmentation fault)

This table is published in *Current Opinion in Structural Biology*, **16**, Edgar R.C. & Batzoglou S., Multiple sequence alignment, 368–373, Copyright Elsevier (2006).

information is also exploited by PRALINE (Simossis & Heringa, 2005), which uses PSI-BLAST (see Chapter 2) to retrieve homologs and build profiles for these sequences. SPEM also builds profiles, and additionally, uses secondary structure information (Zhou & Zhou, 2005). COBALT uses pairwise constraints derived from database searches, in particular from the `Conserved Domain Database` and PROSITE protein motif database, or from user input, and incorporates these into a progressive multiple alignment (Papadopoulos & Agarwala, 2007). Finally, PROMALS uses probabilistic consistency-based scoring applied to progressive alignment in combination with profile information from database searches and secondary structure prediction (Pei & Grishin, 2007). This approach has been shown to be a particular improvement for highly divergent homologs.

All these programs try to exploit information that does not merely come from the input sequences, a research direction that deserves more attention

in the future. In addition, parameter selection needs to be further explored in alignment tools, as benchmarking results may be sensitive to parameter choices. Finally, the rapid development of new alignment algorithms remains unmatched by independent large-scale comparisons of the software implementations, making it difficult to make justified recommendations. A recent comparison of frequently used programs, including CLUSTALW, DIALIGN, T-COFFEE, POA, MUSCLE, MAFFT, PROBCONS, DIALIGN-T and KALIGN, indicated that the iterative approach available in MAFFT, and PROBCONS were consistently the most accurate, with MAFFT being the faster of the two (Nuin *et al.*, 2006).

3.9 Nucleotide sequences vs. amino acid sequences

Nucleotide sequences may be coding or non-coding. In the former case, they may code for structural or catalytic RNA species but more commonly for proteins. In the case of protein-coding genes, the alignment can be accomplished based on the nucleotide or amino acid sequences. This choice may be biased by the type of analysis to be carried out after alignment (Chapter 9); for example, silent changes in closely related sequences may be counted. In this case, an amino acid alignment will not be of much use for later analysis. By contrast, if the sequences are only distantly related, an analysis of amino acid or of nucleotide differences can be performed. Regardless of the end analysis desired, amino acid alignments are easier to carry out and less ambiguous than nucleotide alignments, which is also true for sequence database searching (Chapter 2). Another disadvantage of nucleotide alignment is that most programs do not recognize a codon as a unit of sequence and can break up the reading frame during alignment. Particular two-sequence alignment and database search programs can be exceptions (e.g. SEARCHWISE and PAIRWISE by Birney *et al.*, 1996). A typical approach is to carry out the alignment at the amino acid level and to then use this to generate a corresponding nucleotide sequence alignment. Different computer programs are available to perfom such an analysis; for example, PROTAL2DNA by Catherine Letondal: *http://bioweb.pasteur.fr/seqanal/interfaces/protal2dna.html*, REVTRANS (Wernersen & Pedersen, 2003), TRANSALIGN (Bininda-Emonds, 2005) or DAMBE (Xia & Xie 2001; see Chapter 20).

If the sequences are not protein coding, then the only choice is to carry out a nucleotide alignment. If the sequences code for structural RNA (e.g. small sub-unit ribosomal RNA [SSU rRNA]), these will be constrained by function to conserve primary and secondary structure over at least some of their lengths. Typically, there are regions of clear nucleotide identity interspersed by regions that are free to change rapidly. The performance of automatic software depends on the circumstances, but specific algorithms are being developed (e.g. MURLET;

Kiryu *et al.*, 2007). With rRNA, the most commonly used programs manage to align the large conserved core sections, which are conserved across very broad phylogenetic distances. These core alignment blocks, however, will be interspersed by the highly variable expansion segments; most programs have difficulties with them. Consideration of the secondary structures, perhaps using a dedicated RNA editor, can help but it may still be difficult to find an unambiguous alignment. Excluding these regions from further analysis should be seriously considered; if the alignment is arbitrary, further analysis would be meaningless anyway. Fortunately, there are usually enough clearly conserved blocks of alignment to make a phylogenetic analysis possible.

If the nucleotide sequences are non-coding (e.g. SINES or introns), then alignment may be difficult once the sequences diverge beyond a certain level. Sequences that are highly unconstrained can accumulate indels and substitutions in all positions; these rapidly become unalignable. There is no algorithmic solution and the situation will be especially difficult if the sequences have small repeats (see Section 3.2). Even if a "somewhat optimal" alignment is obtained using a particular alignment score or parsimony criterion, there may be no reason to believe it is biologically reasonable. Such scores are based on arbitrary assumptions about the details of the evolutionary processes that have shaped the sequences. Even if the details of the assumptions are justifiable, the alignment may be so hidden that it has become unrecoverable. Caution should be taken if one alignment cannot be justified over an alternative one.

3.10 Visualizing alignments and manual editing

Although there is a continuous effort to improve biological accuracy of multiple alignment tools, manually refined alignments remain superior to purely automated methods. This will be particularly the case when manual editing takes advantage of additional knowledge (other than sequence data), which is difficult to take into account automatically (although see 3DCoffee, Praline and SPem in Section 3.8). Manual editing is necessary to correct obvious alignment errors and to remove sections of dubious quality. Analogous to automatic alignment, manual editing should be performed on the amino acid level for protein coding sequences. In this way, information about protein domain structure, secondary structure, or amino acid physicochemical properties can be taken into consideration. In Chapter 9, some general recommendations for manual adjustment of a protein alignment are presented.

After refinement, regions that are still aligned unambiguously need to be deleted and a decision needs to be made on how to handle gaps. Not necessarily all positions with gaps need to be discarded (often referred to as "gap stripping") because they

Table 3.3 Multiple alignment editors

Editor	Reference/author and URL	Operation System	Software features
BASE-BY-BASE	Brodie *et al.*, 2004 *http://athena.bioc.uvic.ca/* (look for BASE BY BASE under Workbench)	Java Web Start Application: W, L, M	BBB is a user-friendly alignment viewer with extensive annotation capabilities, including primer display. Although visualization is its main feature, BBB also includes editing and alignment facilities for nucleotide and protein sequences, as well as some additional sequence analysis tools
BIOEDIT	Hall T., 1999 *http://www.mbio.ncsu.edu/BioEdit/bioedit.html*	W	BIOEDIT is a feature-rich manual alignment editor that includes many standard analysis tools and the possibility to configure several external tools to be run through the BIOEDIT interface (e.g. TREEVIEW, BLAST and CLUSTALW). The program allows easy toggling between nucleotide and amino acid translations during editing
GDE (and MACGDE)	De Oliveira *et al.*, 2003; Smith *et al.*, 1994 *http://www.bioafrica.net/GDElinux/index.html*	L, M	GDE is as a multiple sequence alignment editor that can call external command-line programs in a highly customizable fashion (e.g. READSEQ, BLAST, CLUSTALW, PHYLIP programs, etc.). Therefore, it is ideally suited for flexibly pipelining high-throughput analyses
GENEDOC	Nicholas, Nicholas, & Deerfield, 1997 *http://www.nrbsc.org/gfx/genedoc/index.html*	W	GENEDOC is full-featured software for visualizing, editing and analyzing multiple sequence alignments. The program has extensive shading features (incl. structural and biochemical), allows scoring while arranging, and can "re-gap" a DNA alignment when changes were made to the protein alignment
GENEIOUS	Biomatters *http://www.geneious.com/*	W, L, M	GENEIOUS is well-maintained, feature-rich commercial software for visualization and analysis of sequences, including an interface that integrates sequences, alignments, 3-D structures and tree data. The program can perform nucleotide alignment via amino acid translation and back and has user-friendly visualization and editing facilities

(*cont.*)

Table 3.3 (*cont.*)

Editor	Reference/author and URL	Operation System	Software features
JalView	Clamp *et al.*, 2004 *http://www.jalview.org/*	Java Web Start Application: W, L, M	In addition to visualizing and editing protein and nucleotide sequences, JalView can realign selected sequences using Muscle, Mafft or ClustalW, and infer distance-based trees
MacClade	Maddison & Maddison, 1989 *http://macclade.org/macclade.html*	M	MacClade is commercial software for exploring phylogenetic trees and character data, editing data with its specialized spreadsheet editor, and interpreting character evolution using parsimony. Sequence alignment can be done both manually, with extensive selecting and editing functions, and automatically. Amino acid translations can be shown alongside the nucleotide data
Se-Al	Rambaut A. *http://tree.bio.ed.ac.uk/software/seal/*	M	Se-Al is a user-friendly multiple alignment editor particularly useful for manipulating protein coding DNA/RNA sequences since it has the ability to translate in real time DNA to amino acids whilst editing the alignment, meaning that changes made in the amino acid translation are in fact made to the underlying DNA sequences
SeaView	Galtier *et al.*, 1996 *http://pbil.univ-lyon1.fr/software/seaview.html*	W, L, M	SeaView is a graphical multiple sequence alignment editor capable of reading and writing various alignment formats (NEXUS, MSF, CLUSTAL, FASTA, PHYLIP, MASE). It allows the user to manually edit the alignment, and also to run Dot-Plot or ClustalW/Muscle programs to locally improve the alignment

W, Windows; M, MacOs; L, Linux.

can still contain useful information. If the gaps have been inserted unambiguously and the alignment columns are not overly gapped, preferably less than 50%, they can be kept in the alignment. By unambiguous, we mean that it is obvious that a gap needs to be inserted in a particular column rather than the preceding or the following one to recover the true positional homology. The way the information in gapped columns is used in phylogenetic analysis depends on the inference method. Distance-based methods (Chapter 5) either ignore all sites that include gaps or missing data (complete deletion) or compute a distance for each pair of sequences ignoring only gaps in the two sequences being compared (pairwise deletion). In likelihood analysis (Chapters 6 and 7), gaps are generally treated as unknown characters. So, if a sequence has a gap at a particular site, then that site simply has no information about the phylogenetic relationships of that sequence, but it can still contribute information about the phylogenetic relationships of other sequences.

There are several software programs available for manual editing, visualizing and presenting alignments. A detailed software review is available online at the webpage of the Pasteur Institute (*http://bioweb.pasteur.fr/cgi-bin/seqanal/review-edital.pl*). In Table 3.3, we present a selection of some useful, user-friendly alignment editors with a brief description of their specific features, their availability, and the operating system on which they can be run. Although editors are useful tools in sequence analysis, funding and/or publication is not always obvious. As a consequence, researchers may sometimes need to resort to well-maintained and supported commercial software packages.

PRACTICE

Des Higgins and Philippe Lemey

In this practical exercise, we will align TRIM5α gene sequences from different primate species. TRIM5α is a retroviral restriction factor that protects most Old World monkey cells against HIV infection. This data set was originally analyzed by Sawyer *et al.* (2005), and is also used in the practical exercise on molecular clock analysis (Chapter 11). We will employ progressive alignment (**CLUSTALX**), consistency-based scoring (**T-COFFEE**) and iterative refinement (**MUSCLE**) to create different protein alignments and compare the results using the **ALTAVIST** web server. The exercise is designed to cover a program with a graphical user interface, a webserver and a command-line program. We will align the sequences at the amino acid level, compare different alignment algorithms, generate a corresponding nucleotide alignment and manually refine the result. Both the amino acid and nucleotide sequences ("primatesAA.fasta" and "primatesNuc.fasta," respectively) are available for download at *www.thephylogenetichandbook.org*.

3.11 CLUSTAL alignment

3.11.1 File formats and availability

As discussed in the previous chapter, the most common file formats are Genbank, EMBL, SWISS-PROT, FASTA, PHYLIP, NEXUS, and Clustal. The database formats Genbank, EMBL, and SWISS-PROT are typically used for a single sequence and most of each entry is devoted to information about the sequence. These are generally not used for multiple alignment. Nonetheless, **CLUSTAL** can read these formats and write out alignments (including gaps, "-") in different formats, like PHYLIP and Clustal, used exclusively for multiple alignments. In fact, **CLUSTAL** programs can be used as alignment converters, being able to read sequence files in the following formats: NBRF/PIR, EMBL/SWISS-PROT, FASTA, GDE, Clustal, GCG/MSF and NBRF/PIR; and write alignment files in all of the following formats: NBRF/PIR, GDE, Clustal, GCG/MSF and PHYLIP.

CLUSTALW and **CLUSTALX** are both freely available and can be downloaded from the EMBL/EBI file server (*ftp://ftp.ebi.ac.uk/pub/software/*) or from ICGEB in Strasbourg, France (*ftp://ftp-igbmc.u-strasbg.fr/pub/*`ClustalW/` and *ftp://ftp-igbmc.u-strasbg.fr/pub/*`ClustalX/`). These sites are also accessible through the phylogeny programs website maintained by Joe Felsenstein (*http://evolution.genetics.washington.edu/phylip/software.html*). The programs are available for PCs running MSDOS or Windows, Macintosh computers, VAX VMS and for Unix/Linux. In each case, **CLUSTALX** (X stands for X windows) provides

a graphical user interface with colorful display of alignments. CLUSTALW has an older, text-based interface and is less attractive for occasional use. Nonetheless, it does have extensive command line facilities making it extremely useful for embedding in scripts for high-throughput use. The CLUSTAL algorithm is also implemented in a number of commercial packages.

CLUSTAL is also directly accessible from several servers across the Internet, which is especially attractive for occasional users, but it is not without drawbacks. First, users generally do not have access to the full set of features that CLUSTAL provides. Second, it can be complicated to send and retrieve large numbers of sequences or alignments. Third, it can take a long time to carry out large alignments with little progress indication; there may even be a limit on the number of sequences that can be aligned. Nonetheless, excellent CLUSTAL servers can be recommended at EBI (*http://www.ebi.ac.uk/clustalw/*) and BCM search launcher at *http://searchlauncher.bcm.tmc.edu/*.

3.11.2 Aligning the primate Trim5α amino acid sequences

Download the primatesAA.fasta file from the website accompanying the phylogenetic handbook (*http://www.thephylogenetichandbook.org*), which contains 22 primate Trim5α amino acid sequences in fasta format. Open CLUSTALX and open the sequence file using `File → Load Sequences`. The graphical display allows the user to slide over the unaligned protein sequences. Select `Do complete Alignment` from the `Alignment` menu. CLUSTALX performs the progressive alignment (progress can be followed up in the lower left corner), and creates an output guide tree file and an output alignment file in the default `Clustal` format. It is, however, possible to choose a different format in the `Output Format Options` from the `Alignment` menu, such as, for example, the PHYLIP format which, in contrast to CLUSTAL, can be read by many of the phylogeny packages. The file with ".dnd" extension contains the guide tree generated by CLUSTALX in order to carry out the multiple alignment (see Section 3.6.1). Note that the guide tree is a very rough phylogenetic tree and it should not be used to draw conclusions about the evolutionary relationships of the taxa under investigation! CLUSTALX also allows the user to change the alignment parameters (from `Alignment Parameters` in the `Alignment` menu). Unfortunately, there are no general rules for choosing the best set of parameters, like the gap-open penalty or the gap-extension penalty. If an alignment shows, for example, too many large gaps, the user can try to increase the gap-opening penalty and redo the alignment. CLUSTALX indicates the degree of conservation at the bottom of the aligned sequences, which can be used to evaluate a given alignment. By selecting `Calculate Low-Scoring Segment` and `Show Low-Scoring Segments` from the `Quality` menu, it is also possible to visualize particularly unreliable parts of the

alignment, which may be better to exclude from the subsequent analyses. As noted in Section 3.9, the user should keep in mind that alignment scores are based on arbitrary assumptions about the details of the evolutionary process and are not always biologically plausible. Save the amino acid alignment using `File` → `Save Sequences as` and selecting FASTA output format, which automatically saves the file with fasta extension to the directory that contains the input file (and may thus overwrite the input file).

3.12 T-COFFEE alignment

Although standalone **T-COFFEE** software is also available for different operating systems (Windows, Unix/linux, and MacosX), we will perform the consistency-based alignment using the **T-COFFEE** webserver in this exercise (available at *http://www.tcoffee.org/*). We select the regular submission form that only requires us to upload the "primatesAA.fasta" file or paste the sequences in the submission window. An email notification with a link to the results page will be sent if the email address is filled in, but this is not required. As noted above, **T-COFFEE** computations are more time-consuming than **CLUSTAL** progressive alignment. A job submitted on the webserver should take less than 2 minutes to complete. When the alignment procedure is completed, a new page will appear with links to the output files. Save the alignment in fasta format to your desktop (e.g. as "primatesAA_tcoffee.fas"). The score_pdf file contains a nicely quality-colored alignment in pdf format; the same output is presented in an html file.

3.13 MUSCLE alignment

To compute an iteratively refined alignment, we will use the standalone **MUSCLE** software, available for Windows, Unix/linux, and MacosX at *http://www.drive5.com/*. **MUSCLE** is a command-line program and thus requires the use of a terminal (Unix/Linux/MacosX) or a DOS-Window on a Windows operating system. Copy the input file ("primatesAA.fasta") to the **MUSCLE** folder, open a terminal/DOS-Window and go to the **MUSCLE** folder (using "`cd`"). To execute the program type `muscle −in primatesAA.fasta −out primatesAA_muscle.fasta`. In a DOS-Window the executable with extension needs to be specified: `muscle.exe −in primatesAA.fasta −out primatesAA_muscle.fasta`. The program completes the alignment procedure in a few seconds and writes out the outfile in fasta format ("primatesAA_muscle.fasta").

3.14 Comparing alignments using the AltAVisT web tool

To evaluate results obtained using the different alignment algorithms, we use a simple and user-friendly WWW-based software program called **AltAVisT** (*http://bibiserv.techfak.uni-bielefeld.de/altavist/*). **AltAVisT** compares two alternative multiple alignments and uses different color-codes to indicate local agreement and conflict. The regions where both alignments coincide are generally considered to be more reliable than regions where they disagree. This is similar to the reasoning that clusters present in phylogenetic trees, reconstructed by various algorithms, are more reliable than clusters that are not consistently present. A color printout of the result pages obtained for the following alignment comparisons will be useful for further manual editing in Section 3.16.

Go to the **AltAVisT** webserver and click on `OPTION 2: Enter two different pre-calculated alignments of a multiple sequence set`. Upload or paste the alignments generated by **ClustalX** and **T-Coffee**, enter the title "ClustalX" and "T-Coffee," respectively and click `submit`. In the results page, the two alignments are shown with colored residues (several parts of this alignment are shown in Fig. 3.9). When all residues in a single column are shown in the same color, which is the case for the first 46 columns, these residues are lined up in the same way in both alignments. If a residue has a different color, e.g. the arginine "R" for "Howler" in column 52, it is aligned differently in the second alignment (in column 47 instead of 52). The same is true for the glutamic acid residues "E" in the second block (column 89). With respect to column 89 and 90, there is no obvious reason to prefer the first or second alignment since the total alignment score for these columns is the same and phylogenetic inference would not be influenced.

Different colors are used to distinguish groups of residues where the alignment coincides *within* groups but not *between* different groups. An example of this can be observed in the sixth alignment block (column 342–343; Fig. 3.9): the "LT" residues in AGM, Tant_cDNA and AGM_cDNA are aligned by **ClustalX** as well as `T-Coffee`, but not with the same residues from the other taxa. Also the "PS" residues in Rhes_cDNA and baboon are in common columns in the two alignments, but not aligned with the same residues (e.g. not with the "LT" residues from AGM, Tant_cDNA, and AGM_cDNA in the second alignment), explaining why they have different color-code. In the second alignment, all residues have the same color as in the first alignment allowing straightforward comparison of the two alignments. The color-coded comparison reveals that large blocks of alignment discrepancies are concentrated close to gapped regions. So, if gapped regions are to be stripped from this alignment, it is recommended also to delete the neighboring ambiguously aligned columns. The same comparison for the **ClustalX** and **Muscle** alignments

Fig. 3.9 Parts of the TRIM5α CLUSTALX alignment coloured by ALTAVIST. The colours are based on the comparison between CLUSTALX and T-COFFEE. The different parts of the alignment shown are interspersed with two dots; the position of the start and end of the column of each part are indicated at the bottom.

reveals very similar discrepancies. Not surprisingly, there are fewer differences between the **T-COFFEE** and **MUSCLE** alignments. We will use this information when manually editing the sequence alignment (Section 3.16).

3.15 From protein to nucleotide alignment

All three programs discussed above frequently break up the coding reading frame when aligning the TRIM5α nucleotide sequences. This would invalidate further codon-based analysis, e.g. inference of positively selected sites like originally performed on this data set (Sawyer *et al.*, 2005; see also Chapter 14), and extensive manual editing would be required to restore the reading frame. To avoid this, the protein alignment can be used to generate a corresponding nucleotide sequence alignment. This procedure also takes advantage of the fact that protein alignments are less ambiguous and faster to compute.

We will create a TRIM5α nucleotide alignment using the **PROTAL2DNA** web-application available at *http://bioweb.pasteur.fr/seqanal/interfaces/protal2dna.html*. Go to WWW-submission form and enter your email address. Upload the **CLUSTALX** protein alignment and the unaligned nucleotide sequences ("primatesNuc.fasta," available at *www.phylogenetichandbook.org*). Because the order in which sequences appear in both files may be different, select the option `identify corresponding DNA sequences by same ID or name (-i)`. Select the `Pearson/Fasta Output Alignment format` and run the application.

Save the output file "protal2DNA.out" to your computer and change the name to "primatesNuc_pro2DNA.fasta."

The complete procedure of aligning protein sequences and generating the corresponding nucleotide alignment can also be performed in an automated fashion using programs like RevTrans, transAlign and Dambe. We will briefly discuss how to do this using Dambe, a feature-rich PC program for data analysis in molecular biology (*http://dambe.bio.uottawa.ca/software.asp*), assuming a fully working version of the program is installed on your PC.

(i) Copy the "primatesNuc.fasta" file to the Dambe folder, run DAMBE.exe, and open the file by choosing `Open standard sequence file` from the `File` menu. The window "Open" appears: set the `Files of type` option to `Pearson/Fasta`, browse to the Dambe folder and open "primatesNuc.phy."
(ii) In the "Sequence Info" window, select `Protein-coding Nuc. Seqo` and keep the standard genetic code. Click the `Go!` button.
(iii) Select `Work on Amino Acid Sequence` from the `Sequences menu` and confirm that all sequences have a complete start codon.
(v) Select `Align Sequences Using ClustalW` from the `Alignment` menu, keep default settings and Click the `Go!` button.
(vi) When the protein alignment is completed select `Align Nuc. Seq. to aligned AA seq. in buffer` from the `Alignment` menu. Have a look at the requirements for the back-translation and proceed by opening the original "primatesNuc.fasta" file. The corresponding nucleotide alignment can be saved in different output formats by selecting `Save or Convert Sequence Format` from the `File` menu. Note that Dambe has several automated features to delete gapped columns or other parts of the alignment using the `Get Rid of Segment` option in the `Sequences` menu.

3.16 Editing and viewing multiple alignments

As recommended in Section 3.10, we will edit the alignment at the amino acid level. However, if we wish to perform further analysis on the nucleotide sequences, it would be preferable to edit the nucleotide alignment by making changes in the appropriate amino acid translation mode. Two editors allowing the user to flexibly toggle between nucleotides and amino acids while making changes to the alignment are BioEdit for Windows and Se-Al for MacOS. We will now briefly discuss how to delete ambiguously aligned regions in the coding gene alignment based on the AltAVisT comparisons and provide some basic instructions for using either BioEdit or Se-Al.

Open the coding gene alignment "primatesNuc_pro2DNA.fasta" obtained by the PROTAL2DNA web-application (`BioEdit` & `Se-AL`: `File` → `Open`; Note that shortcut keys are available for both programs). Changing the background colour

can make the view more clear (BioEdit: View → View Mode → Inverse Background Colored; Se-Al: Alignment → Use Block Colours). Switch to amino acid translation (BioEdit: Ctrl+A to select all sequences, Sequence → Toggle Translation or Ctrl+G; Se-Al: Alignment → Alignment Type → Amino Acid or command+T).

The first alignment ambiguity indicated by the AltAVisT comparisons is situated in columns 47 to 52, where different algorithms did not agree how to align the arginine "R" for "Howler" (Fig. 3.9). The most conservative option here would be to delete these six columns all together. In both programs, these columns can be selected by click-dragging the mouse over the relevant alignment region. In Se-Al, the columns can simply be deleted using Edit → Cut or command-X (Fig. 3.10). In BioEdit, we need to switch to the Edit mode in the alignment window menu (Fig. 3.10), and revert (Ctrl+G) to the nucleotide sequences. No direct editing on the amino acid translation is allowed, but our selection remains in the nucleotide view and can be deleted using Edit → Cut or Ctrl-X. Note that Se-Al can only "undo" the last action, whereas BioEdit does not have this limitation. The next alignment differences were indicated in columns 89 and 90. However, as discussed above, the differences are, in fact, equivalent for phylogenetic inference and no editing is required for this purpose. Based on the ambiguities indicated by AltAVisT (both ClustalX vs. T-Coffee and ClustalX vs. Muscle) and the presence of highly gapped columns, the alignment regions that should preferably be deleted are 47–52, 326–362, 403–408 and 425–511 (numbering according to the protein translation of "primatesNuc_pro2DNA.fasta"). The first three of these regions are shown in Fig. 3.9. When these regions are being deleted from the beginning of the aligment the numbering will change to 47–52, 320–356, 360–365 and 376–462. The edited alignments can now be saved to your computer in different formats (BioEdit: File → Save As ... or File → Export → Sequence Alignment; Se-Al: File → Export or command+E). A file with the modified alignment has been made available to check whether the correct editing was performed.

In this exercise, editing was restricted to deleting unreliable regions. This will be the case in many real-world alignment situations. However, considering only a single alignment, it can be necessary to correct obvious alignment errors. Both programs have extensive facilities to perform these refinements using select and slide, grab and drag, copy, cut, and paste functions.

3.17 Databases of alignments

In the future, it will become more clear how many different protein and RNA sequence families actually exist, and large alignments of these will be available

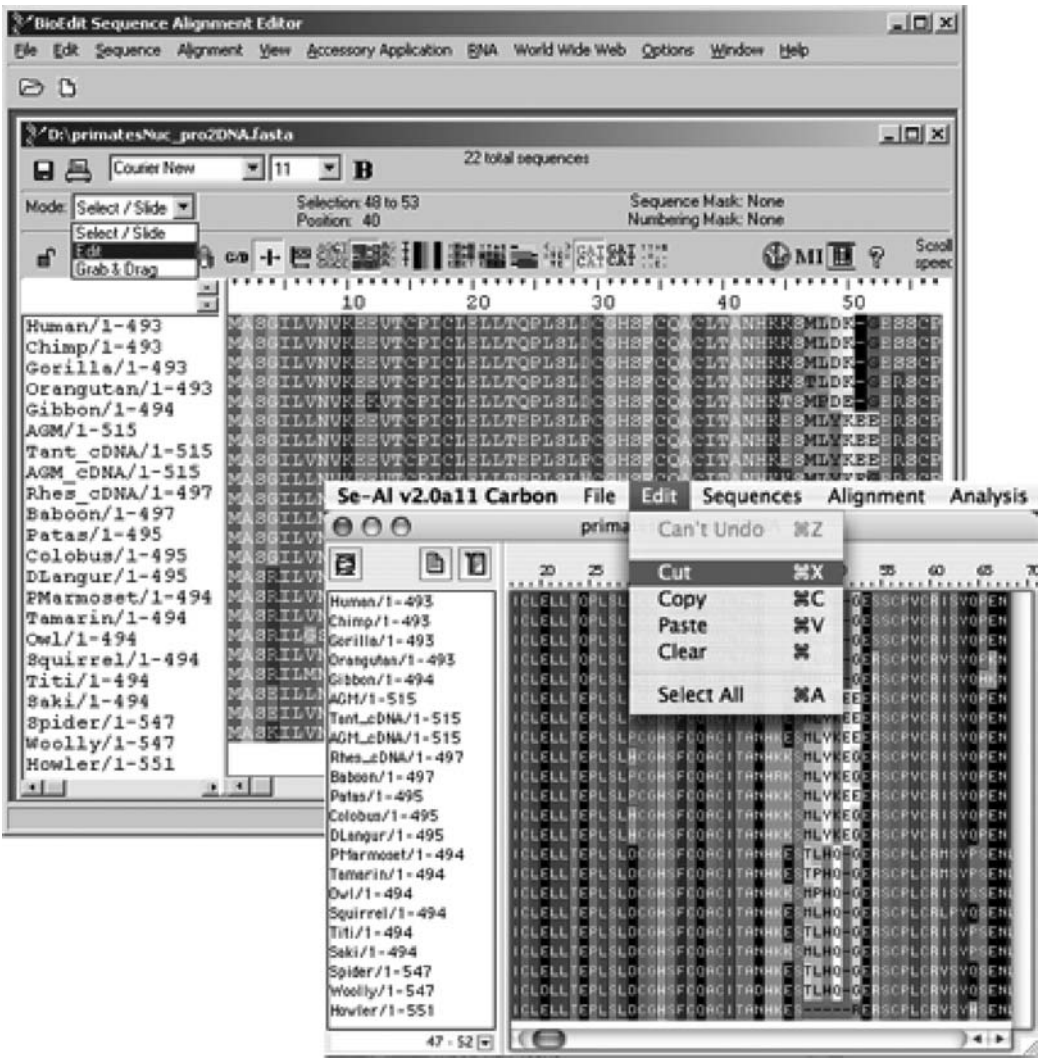


Fig. 3.10 Screen shots of BioEdit and Se-Al. In both programs, the protein coding alignment is loaded, translated to amino acids, and the columns 47 to 52 are selected. In BioEdit, the `Mode` is being switched to Edit; in Se-Al, the columns are being deleted.

through computer networks. In the meantime, huge alignments for the biggest and most complicated families (rRNA) already exist, which are maintained and updated regularly (*http://www.psb.ugent.be/rRNA/*). Also databases for specific organisms with alignments are being maintained (e.g. HIV: *http://www.hiv.lanl.gov/*). In this context, it is interesting to note that programs like Clustal have utilities to align sequences or alignments against pre-built alignments. There are numerous databases of protein alignments, each with different advantages and

disadvantages, different file formats, etc. The situation is settling down with the INTERPRO project (*http://www.ebi.ac.uk/interpro/index.html*), which aims to link some of the biggest and most important protein family databases. INTERPRO is an invaluable resource for obtaining detailed information on a family of interest and for downloading sample alignments. These alignments can be used as the basis for further phylogenetic work or simply for educational or informational purposes.