

Error in the alignment of your sequences can have a major impact on the reconstructed phylogeny

2.1 The Basic Idea

Before you build a phylogenetic tree from sequence data, the usual way to begin is to first make a *multiple sequence alignment (msa)*. This is an ordering of the homologous sequences in which equivalent spatial positions are lined up against each other [Figure 2.1](#).

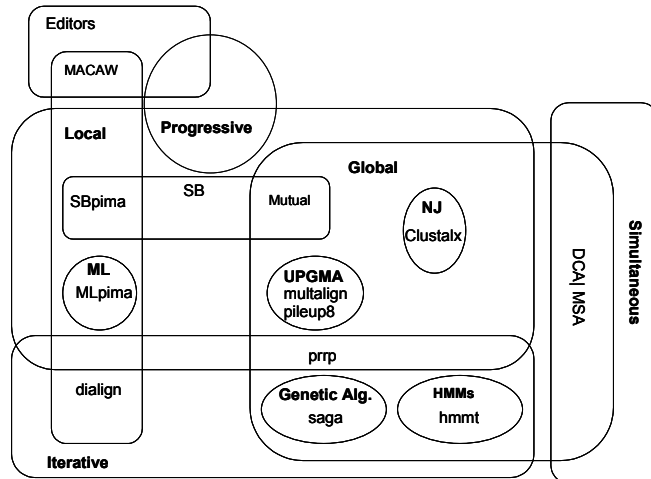
```
seqA A-UUUAA-GCGT-TG
seqB AC--UAAGCGCGCTG
seqC ACUAUAAGCGTGCCG
seqD ACCUAAA-GTGC-TG
```

2.1

The task is simple when the sequences to be aligned are very similar. However as sequences become increasingly diverged from each other during evolution, the task becomes less simple. *Insertion and deletion* of residues (*indels*) often occur in one or more of the sequences. You must decide whether it is reasonable to align the sequences end to end or to only align sub-regions that are similar in all the sequences. You also need to decide, in which sequences, and where to put gaps so as to maintain equivalent spatial positions between the sequences [Figure 2.1](#).

An important point to make at the outset is that no current alignment algorithm is capable of producing good, reliable alignments with all sequence data sets. Consequently, as with tree building, recognising certain properties of your sequences is important in helping to decide the most appropriate alignment method(s) to choose. Important questions for you are: how different are your sequences from each other? Are some of the sequences more diverged than others? Do any of the sequences have large or many indels? Answers to these questions will help guide your approach and help you to face an almost overwhelming number of methods with overlapping and defining properties. A schematic overview of some of these methods and their sources is given in [Figure 2.2](#). In this chapter it is not possible to cover all alignment methods. Instead we focus on important

principles and practical concerns when building alignments. We point to recent observations on the performance of different methods.



2.2

URLs

MACAW: <ftp://ftp.ncbi.nlm.nih.gov/pub/macaw/>

Se-AI2: <http://evolve.zoo.ox.ac.uk/projects/BBSRC-Nutilus/SoftwarePages/SeAI.html>

BioEdit: <http://www.mbio.ncsu.edu/BioEdit/bioedit.html>

PIMA: <http://bioweb.pasteur.fr/seqanal/interfaces/pima-simple.html>

DIALIGN: <http://bibiserv.techfak.uni-bielefeld.de/dialign/>

SAGA: http://igs-server.cnrs-mrs.fr/~cnotred/Projects_home_page/saga_home_page.html

HMMT: <http://www.psc.edu/general/software/packages/hmmer/manual/node71.html>

MULTALIGN: www.toulouse.inra.fr/multalin.html

PILEUP: <http://www.genomics.purdue.edu/gcg/gcg-docs/pileup.html>

CLUSTALX: <http://www-igbmc.u-strasbg.fr/BioInfo/ClustalX/Top.html>

DCA: <http://bibiserv.techfak.uni-bielefeld.de/dca>

MSA: www.psc.edu/general/software/packages/msa/manual/manual.html

Pairwise Alignment

We begin our treatment of alignment by first introducing the concept of a *dot plot* and the alignment of two sequences. We cover the principles of *dynamic programming (DP)*, and although this might sound terrifying, please keep in mind that DP is just a simple method of rigorously deciding where to place the gaps between sequences that you are aligning. Our first example with dynamic programming uses a *Needleman-Wunsch* objective function. The set of mathematical rules – *algorithm* – for this function produce an *optimal alignment* of sequences end to end – i.e. an *optimal global alignment*. Next we show how the *Needleman-Wunsch* objective function can be easily modified to give the

Smith-Waterman objective function. The algorithm for this function will produce an *optimal local alignment*. To illustrate both methods, we use examples with a simple scoring system to help us decide whether or not one alignment is better than another. In scoring alignments we count only the number of matches, mismatches and number of residue gaps. We develop alignment principles further by introducing *log odds scoring matrices*. These are tables of scores that indicate how frequently you expect to find a particular residue in one sequence lined up against a particular residue in another homologous sequence. Empirically, it has been found that use of *log odds scores* result in more biologically realistic alignments. We outline the problem of scoring penalties for introducing gaps into alignments and discuss *complex* and *dynamic gap penalties*.

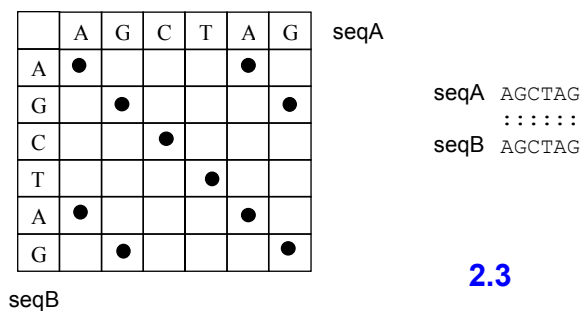
Multiple Sequence Alignment

At this point you have all the basics needed to move on to the alignment of more than two sequences using dynamic programming. We do this and discuss the computational problem caused by this natural extension. Heuristic approaches implemented to overcome this problem are introduced. In this context we discuss principles of the software programs MSA, DCA and CLUSTALX. We also cover the iterative approach implemented in the program Prrp which has been found to improve the quality of global multiple sequence alignment in some cases. We come back to the issue raised at the outset of the chapter concerning whether or not it is reasonable to align sequences across their entire length. We describe the segment based method of multiple sequence alignment used in DIALIGN. This is a relatively new approach, which performs well on sequence data sets with large N/C terminal extensions and internal insertions. After a discussion on the relative performance of different methods we end with some words of advice from researchers.

2.2 From Dot Plots to Dynamic Programming

Dot Plots

The *Dot plot* is a helpful way to introduce basic ideas of pair-wise alignment – i.e. the alignment of two sequences. [Figure 2.3](#) shows an example of a dot plot when a sequence is lined up against itself. The dot plot helps us to visualize the similarity between two sequences that are compared. The dot plot is constructed simply by lining one of the sequences across the top of a table, and then by lining the other sequence down the side of the table. You will notice that a dot is placed where the intersection of a column and row match the same residue (A:A, G:G etc) and for the case – such as ours – where the sequences are identical to each other you will obtain a series of consecutive dots which are all on the same diagonal. In general, the more dots that appear on the same diagonal, the stronger the exact match is between the two sequences compared.

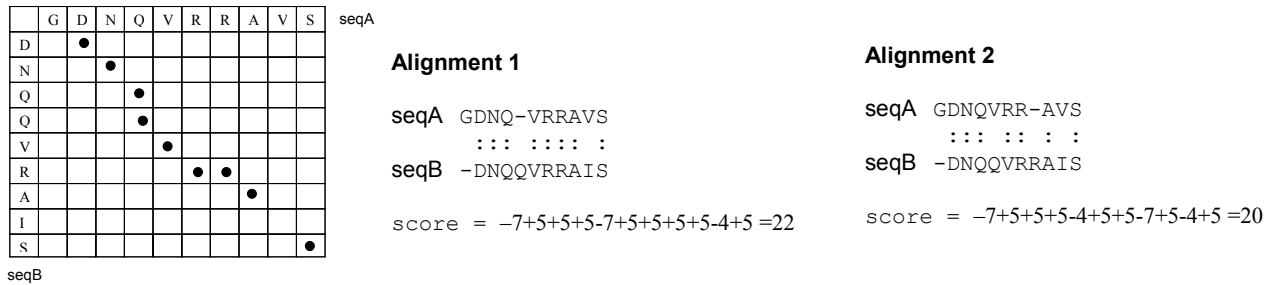


[Figure 2.4](#) shows an example where substitution differences occur between the two sequences compared. Again you see consecutive exact matches indicated by dots on diagonals but notice in our example that there are now three diagonals of dots offset with respect to each other. This is the observation expected if indels have occurred in one or other of the sequences. To accommodate these indels, gaps need to be added into the alignment so as to maintain positional homology between the two sequences.

Using a simple scoring system we can calculate the scores for possible alignments made between our two sequences. For the example used in [Figure 2.4](#) let us score: matching bases between the two sequences = 5, mismatching bases = -4, introducing a gap in one of the sequences = -7, introducing two gaps = -14, three gaps = -21 and so forth. Using

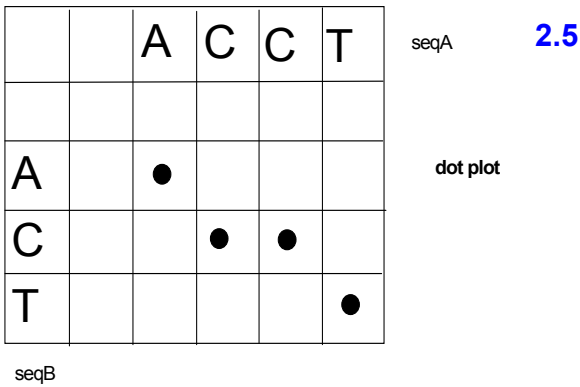
this criteria we can see that the placement of gaps as shown in alignment 1 gives us a better score (22) than that of the gap placement choices in alignment 2 (score = 20). We would say that the *alignment quality* of 1 is better than that of 2.

2.4

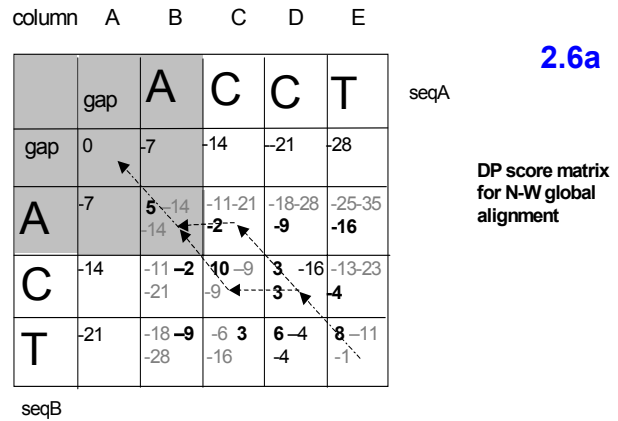


In general a rigorous and systematic approach is needed for finding the best place to put the gaps. This can be done using *dynamic programming (DP)*, an approach that will evaluate the scores for all gap placements between any two (or more) sequences. In principle, a scoring table is produced that looks somewhat like a modified dot plot. The values calculated for the table allow a path or paths to be drawn through the table. These paths identify the optimal alignment between the sequences.

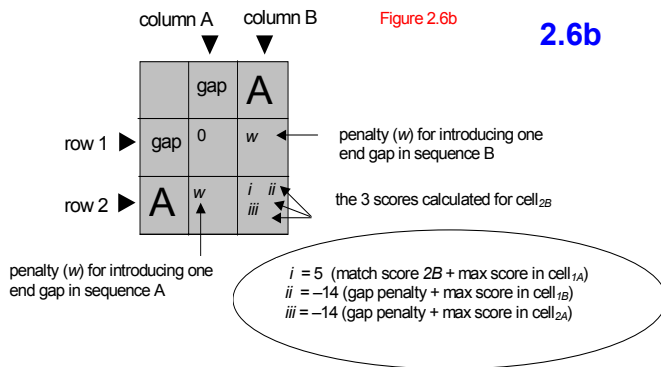
Let us consider an example to see how this works. [Figure 2.5](#) shows the dot plot for two sequences and [Figure 2.6](#) shows a dynamic programming (DP) score matrix for a global alignment of the same two sequences. Instead of having dots in the cells of a DP score matrix the cells contain scores calculated using simple formulae. Note that most descriptions of dynamic programming introduce two matrices (an S_{ij} matrix and a trace back path matrix). However, you can make do with just one matrix or table. For each cell in the matrix you need to calculate and record three values. Moving from one corner of the matrix to the other, these three entries are filled out in every cell. Once this is done, you can trace a path back through the matrix to identify the optimal alignment between the two sequences. The formulae for filling out a DP score matrix differs depending on whether you want to make a local optimal alignment (i.e. identify the highest scoring aligned sub region between your two sequences) or a global optimal alignment (i.e. identify the highest scoring alignment which goes end to end between your two sequences). We first consider a global optimal alignment.



Close up of part of DP score matrix



Alignment 1 **Alignment 2** Both score = 8
 ACCT ACCT (3 matches + 1 gap)
 :: : : :: ((3x5) - 7) = 8
 AC-T A-CT



Global Optimal Alignment

The values in [Figure 2.6](#) have been calculated using the algorithm described by Needleman and Wunsch (N-W) - which will identify an optimal global alignment(s) between two sequences. To illustrate their method, we use a scoring system in which we penalise the placement of gaps at the beginning of the alignment since this helps force the sequences to be aligned end to end. To penalise endgaps, we fill out the first row and column of the DP matrix with values that correspond to the penalties incurred for introducing 1 gap, 2 gaps, 3 gaps etc at the beginning of either sequence. For our matrix we will use the same scoring criteria as used in the example of [Figure 2.4](#) (match = 5, mismatch = -4, gap = -7). Thus the values -7, -14, -21, -28 etc in [Figure 2.6a](#) correspond to the penalties that would be incurred if either sequence in the alignment was to begin with one, two, three etc gaps.

The three values (*i*, *ii* and *iii*) that are calculated for all other cells in the matrix are made considering the three possibilities:

- (i) that the sequences can be lined up to that point without introducing any gaps
- (ii) that a gap should be introduced at that point in sequence A
- (iii) that a gap should be introduced at that point in sequence B.

To calculate *i* (the running score for aligning the facing pair of residues) for cell_{2B} look at the residues in the column and row that face each other across cell_{2B} (for a close up view of nine cells from this example see [Figure 2.6b](#)). Add the score for the match/mismatch of these facing residues to the highest positive value from cell_{1A}. Write this value in the top left corner of cell_{2B}. Corresponding to cell_{2B} you will see that an “A” (row 2) matches against an “A” (column A) thus the score for *i* in cell_{2B} = 5+0=5.

To calculate *ii* (the running score for putting a gap at that point in sequence A) for cell_{2B} add the gap penalty to the highest positive score in the cell immediately above cell_{2B}. In our example the cost of introducing a gap = -7 so the score for *ii* in cell_{2B} = -7+0=-7. Write the value for *ii* into cell_{2B} as shown in the figure.

To calculate *iii* (the running score for putting a gap at that point in sequence B) for cell_{2B} add the penalty score for introducing a gap to the highest positive score in the cell immediately to the left of cell_{2B} you are in. In our example the cost of introducing a gap = -7 so the score for *iii* in cell_{2B} = 0-7=-7. Write the value for *iii* into cell_{2B} as shown in the figure.

Repeat the above process for each cell in the matrix until you have filled out all the cells with three values. The filled out matrix is shown in [Figure 2.6a](#). The highest positive score in each cell has been indicated in bold font. Once the table is completed, it shows a running score for aligning the two sequences – scored position by position and evaluating gap placements at all possible positions.

The Needleman and Wunsch global alignment is found by starting at the bottom right corner of the table and tracing back a path that goes from cell to cell until you reach the opposite corner of the matrix. The cells that lie on the optimal path are identified by the scores in each cell. Portions of the path on a diagonal mean the two sequences are aligned without gaps (as in [Figure 2.3](#)), while vertical portions of the path indicate introducing a gap or gaps in the sequence along the top of the matrix. Horizontal portions of the path indicate introducing a gap or gaps in the sequence down the side of the matrix.

A cell is on the optimal trace back path if the best score in that cell has given rise to the best score in the cell you are currently visiting. For example as shown in [Figure 2.6a](#) cell_{3D} is on the optimal path leading to cell_{4E} because 3 (the best score in cell_{3D}) + 5 (the score for aligning T:T – the facing residues in cell_{4E}) = 8 (the best score in cell_{4E}). Similarly, cell_{2C} is on the path to cell_{3D} because the best score in cell_{2C} + the score for aligning C:C – the facing residues in cell_{3D} = the best score in cell_{3D}. Note that here may be more than one optimal trace back path and hence optimal alignment. This is the case in our example where there are actually two equally good alignments as judged by our scoring criterion. You will see this if you examine [Figure 2.6a](#) – cell_{3C} is also on the path to cell_{3D} because the best score in cell_{3C} + gap penalty = the best score in cell_{3D}. The optimal trace back paths across the whole matrix and the alignments that they correspond to are shown in [Figure 2.6a](#).

The overall alignment score – i.e. *measure of alignment quality* is the sum of the scores obtained for each site in the alignment. You might like to think of this as a sum of pairs scores where for each site in the alignment there is just one pair (residue matched against residue or residue matched against a gap). Later when we discuss multiple sequence alignment we also consider a sum of pairs criterion to measure alignment quality but in that case there are more pairs at each site to consider.

Local Optimal Alignment

Whilst global alignments are the most common starting point when tree building from sequences, a multiple sequence alignment can also be built using a local alignment

criterion. This is sometimes done when sub-regions of the sequences are highly conserved whilst other regions of the molecule (which often are excluded at the tree building stage) are either poorly conserved, contain difficult to align indels or perhaps are not even homologous to regions in the other sequences. For sequence data of this type and other complex data, global and local methods may give sometimes produce different alignments. Consider the problem case illustrated in [Figure 2.7](#) .

| | gap | D | E | D | E | V |
|-----|-----|-----|-----|-----|-----|-----|
| gap | 0 | -7 | -14 | -21 | -28 | -35 |
| D | -7 | 5 | -14 | -11 | -21 | -25 |
| E | -14 | -11 | -2 | -9 | -16 | -23 |
| V | -21 | -18 | -9 | -6 | -4 | -11 |
| D | -28 | -16 | -16 | -13 | -4 | -8 |

DP score matrix for N-W global alignment

2.7a

| | gap | D | E | D | E | V |
|-----|-----|---|----|---|----|----|
| gap | 0 | 0 | 0 | 0 | 0 | 0 |
| D | 0 | 5 | 0 | 0 | 5 | 0 |
| E | 0 | 0 | 10 | 0 | 10 | 0 |
| V | 0 | 0 | 0 | 3 | 0 | 15 |
| D | 0 | 0 | 0 | 0 | 8 | 0 |

DP score matrix for S-W local alignment

2.7b

To obtain a *Smith Waterman* (S-W) local alignment you fill out a DP score matrix as explained above for a N-W global alignment. However, you modify the Needleman - Wunsch global algorithm by setting to zero any negative values that you obtain as you fill out the table. To get the values in the matrix of [Figure 2.7a](#) use the same scoring system as used before in [Figure 2.6](#) (i.e. 5 for a match, -4 for a mismatch, -7 for a gap). In contrast if the same scoring system is used with the Needleman –Wunsch algorithm you get the values shown in the matrix of [Figure 2.7b](#).

With the Smith - Waterman algorithm the optimal local alignment is identified by searching for the most positive value in the matrix (it is 15 in our example) and then by using the same back tracking rules as before for finding the global alignment. The overall score for the optimal Smith - Waterman alignment is the score for each aligned position summed over all positions. Because of the way the table is filled out this will equal the most positive number in the matrix. In our example, the score for the optimal local alignment = 15.

Note that in our example, the local alignment ([Figure 2.7b](#)) is not contained within the global alignment ([Figure 2.7a](#)), and if there is only one correct biological alignment, then either alignment [2.7a](#) or [2.7b](#) must be wrong! Researchers have reported cases where global methods but not local methods have been able to align expected structural motifs in sequences. Conversely examples have also been found where local methods outperform global methods. Unfortunately it is not possible to simply assume one approach is better than another, thus you may need to resort to other available biological information in deciding which alignment is better. This might be structural information or it could be reference to a database of substitution types that are more likely than others. In the case of the latter, alignment procedures are known to be improved – made more sensitive by considering the substitution types that have occurred between the sequences.

2.3 Log odds scoring matrices

Empirical observations from studies comparing aligned homologous sequences have shown that certain residue substitutions occur more frequently than others. For example, a commonly made inference is that DNA “transitions” (e.g. adenine substituting the guanine; cytosine to thymidine etc) occur more frequently than DNA “transversions” (e.g. adenine substituting to thymidine, guanine to cytosine etc). Similarly, certain amino acids substitute more frequently than others (e.g. leucine, valine and isoleucine exchange more frequently with each other in certain proteins than say cysteine or tryptophane do). Information on relative rates of exchangeability can be used to make sequence alignment more sensitive. Later, when we more fully describe *sequence substitution models* ([Chapter 3](#)), we will describe different approaches that have been used to empirically gather the observations on residue exchanges in homologous sequences. For now though, we introduce the idea that empirical studies on sequence substitution patterns have led to development of scores for different types of substitutions, and that using these scores makes alignment procedures more sensitive. The empirical studies are most advanced in the study of protein sequence evolution. Hence much of our discussion at this point concerns alignment of amino acid sequences rather than DNA sequences.

Scores for matching one residue against another in an alignment are typically contained in *log odds matrices*. An example of a log odds matrix is shown in [Figure 2.8](#).

Pam 250 log odds score matrix

| | C | S | T | P | A | G | N | D | E | Q | H | R | K | M | I | L | V | F | Y | W | | |
|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|----|----|--|---|
| C | 12 | | | | | | | | | | | | | | | | | | | | | C |
| S | 0 | 2 | | | | | | | | | | | | | | | | | | | | S |
| T | -2 | 1 | 3 | | | | | | | | | | | | | | | | | | | T |
| P | -3 | 1 | 0 | 6 | | | | | | | | | | | | | | | | | | P |
| A | -2 | 1 | 1 | 1 | 2 | | | | | | | | | | | | | | | | | A |
| G | -3 | 1 | 0 | -1 | 1 | 5 | | | | | | | | | | | | | | | | G |
| N | -4 | 1 | 0 | -1 | 0 | 0 | 2 | | | | | | | | | | | | | | | N |
| D | -5 | 0 | 0 | -1 | 0 | 1 | 2 | 4 | | | | | | | | | | | | | | D |
| E | -5 | 0 | 0 | -1 | 0 | 0 | 1 | 3 | 4 | | | | | | | | | | | | | E |
| Q | -5 | -1 | -1 | 0 | 0 | -1 | 1 | 2 | 2 | 4 | | | | | | | | | | | | Q |
| H | -3 | -1 | -1 | 0 | -1 | -2 | 2 | 1 | 1 | 3 | 6 | | | | | | | | | | | H |
| R | -4 | 0 | -1 | 0 | -2 | -3 | 0 | -1 | -1 | 1 | 2 | 6 | | | | | | | | | | R |
| K | -5 | 0 | 0 | -1 | -1 | -2 | 1 | 0 | 0 | 1 | 0 | 3 | 5 | | | | | | | | | K |
| M | -5 | -2 | -1 | -2 | -1 | -3 | -2 | -3 | -2 | -1 | -2 | 0 | 0 | 6 | | | | | | | | M |
| I | -2 | -1 | 0 | -2 | -1 | -3 | -2 | -2 | -2 | -2 | -2 | -2 | -2 | 2 | 5 | | | | | | | I |
| L | -6 | -3 | -2 | -3 | -2 | -4 | -3 | -4 | -3 | -2 | -2 | -3 | -3 | 4 | 2 | 6 | | | | | | L |
| V | -2 | -1 | 0 | -1 | 0 | -1 | -2 | -2 | -2 | -2 | -2 | -2 | -2 | 2 | 4 | 2 | 4 | | | | | V |
| F | -4 | -3 | -3 | -5 | -4 | -5 | -4 | -6 | -5 | -5 | -3 | -4 | -5 | 0 | 1 | 2 | -1 | 9 | | | | F |
| Y | 0 | -3 | -3 | -5 | -3 | -5 | -2 | -4 | -4 | -4 | 0 | -4 | -4 | -2 | -1 | -1 | -2 | 7 | 10 | | | Y |
| W | -8 | -2 | -5 | -6 | -6 | -7 | -4 | -7 | -7 | -5 | -3 | -2 | -3 | -4 | -5 | -2 | -6 | 0 | 0 | 17 | | W |
| C | | S | T | P | A | G | N | D | E | Q | H | R | K | M | I | L | V | F | Y | W | | |

2.8

The numbers in this matrix correspond to the scores specified for every possible matching of amino acid pairs (C:S, C:T and so on). Each value in a log odds matrix is a measure of the evolutionary significance of finding a particular residue matched against another particular residue at the same site in a pairwise alignment of sequences. The higher the log odds score the more evolutionary significant is the match of residues. Each value in the *log odds matrix* is actually the logarithm (usually to base 2) of the ratio of two probabilities. The reason for taking the logarithm is because, to get the overall score for the alignment, we simply add the log odds scores for each sequence position in the alignment (otherwise we would need to be multiplying probabilities across sequence positions and that could get complicated). In terms of alignment quality, the higher the summed log odds score, the better the alignment. The construction of *Log odds matrices* takes into account two probabilities:

- (1) the probability of matching a given pair of residues in an alignment of two homologous sequences
- (2) the probability of matching the same pair of residues in two aligned sequences if they were not homologous.

The first probability can be obtained from observations on the rates of exchange for particular residues in alignments of homologous sequences. These observations have been made for numerous sequence data sets: i.e. for exchanges between closely related homologous protein sequence data sets (e.g. as used in the construction of *PAM matrices*) as well as for conserved regions or blocks contained in multiple sequence alignments for homologous protein sequences showing different degrees of divergence (e.g. as in *BLOSUM matrices*).

The second probability takes into account how common different residues are in the data base of sequences used to calculate the first probability. Calculating the second probability is important, because for example, if you had a big collection of non homologous sequences, but nevertheless certain residues (e.g. leucine and valine) which were very common, then you might expect that just by chance these amino acids would face each other in an alignment even if the two sequences were not homologous. The ratio of probability (1) and (2) – which are measures of evolutionary significance of finding two particular residues aligned in any two sequences - are called *odds relatedness* values. Taking the logarithm of these values gives *log odds relatedness* values, and it is these values that are contained in the log odds scoring matrices.

2.4 Complex gap penalties

For our dynamic programming example in [Figure 2.6](#) and [2.7](#) we used a simple gap penalty. It was of the form: gap penalty (w) = -7. However, some alignment programs use a more complicated gap penalty. For example, the *opening* of gaps can be more heavily penalized than the *extension* of existing gaps. Such a penalty is often written in the form $w = g - r(x-1)$ where g = opening penalty, r = extension penalty and x = the number of residue positions that the gap covers. If used to help fill out a DP matrix, doing this becomes a bit more complicated. The reason for this is that when you move in either a horizontal or vertical direction in the matrix the penalty value you add depends on

whether you are opening a new gap at that position or whether you are extending a gap that is already open at that position.

Assigning gap penalties for dynamic programming purpose can become even more complicated in an attempt to better model biology. Empirical studies suggest that different gene regions can be more or less susceptible to the occurrence of indels. Additionally, the rate of indel formation seems to depend on the extent of divergence of homologues. Some alignment programs such as CLUSTALX can take into account the local residue composition and divergence of sequences when determining an appropriate gap penalty. With CLUSTALX as sequences are progressively aligned, different penalties are used.

2.5 The computational problem of aligning more than two sequences

The principle of using dynamic programming to align two sequences can be extended to more than two sequences. For very small data sets this can be done rigorously, by simultaneously evaluating the best places to put gaps in all of the sequences - an approach called *simultaneous multiple sequence alignment*. However, recall that with pairwise alignment the dynamic programming problem was equivalent to finding the shortest path (i.e. the back tracking path) through a two dimensional space. When you align three sequences simultaneously, the problem is equivalent to finding the shortest path through a 3 dimensional space. For n sequences the problem is equivalent to finding the shortest path through an n dimensional space. (don't think about this too much it will give you a headache). Fortunately, you do not need to think in terms of multidimensional space to understand how a simultaneous multiple sequence alignment might be constructed. Another way to think about the problem is to recognise that the optimal alignment for different pairs of sequences from your data set may require different gap placements. For example consider the alignment in [Figure 2.9](#).

| | | |
|-----|------------|------|
| 2.9 | sequence A | DEDR |
| | sequence B | D-DR |
| | sequence C | DDEK |

When aligning sequence A and sequence B, and without reference to sequence C, you might place a gap in the position as shown in sequence B - so as to maintain the alignment of the D residues in these two sequences. However, if you were to align sequence B and sequence C without reference to sequence A you might place the gap after both D residues and before the R residue in sequence B. In simultaneous multiple sequence alignment the decision of where to place gaps is made with reference to all of the sequences, not just pairs of sequences.

The objective function used to determine the gap placements is typically based on a *sum of pairs* criterion. The best alignment being the one that has gap placements that result in the best sum of pairs score. This objective function is very similar to that used in making pair-wise alignments. In a pairwise alignment the score for a particular site is the score for matching either (a) a pair of residues, (b) a gap against a residue or (c) a gap against a gap. The score for the overall alignment is obtained by the individual site scores across all sites. In a multiple sequence alignment, the score for a particular site is the sum of pairs score for that site. The overall alignment score is the sum of pairs scores summed across all sites in the alignment. An example calculation is shown in [Figure 2.10](#).

2.10

| | | | |
|------------|------------|------------|--|
| | (a) | (b) | |
| sequence A | DEDR | DEDR | |
| sequence B | D-DR | DD-R | |
| sequence C | DDEK | DDEK | |

for alignment (a) calculation of sum of pairs score =

column 1: (D_{seqA}:D_{seqB} + D_{seqA}:D_{seqC} + D_{seqB}:D_{seqC}) +
column 2: (E_{seqA}:gap_{seqB} + E_{seqA}:D_{seqC} + gap_{seqB}:D_{seqC}) +
column 3: (D_{seqA}:D_{seqB} + D_{seqA}:E_{seqC} + D_{seqB}:E_{seqC}) +
column 4: (R_{seqA}:R_{seqB} + R_{seqA}:K_{seqC} + R_{seqB}:K_{seqC}).

= (4+4+4) + (-10+3-10) + (4+3+3) + (5+3+3)= 17.

for alignment (b) calculation of sum of pairs score = 17.

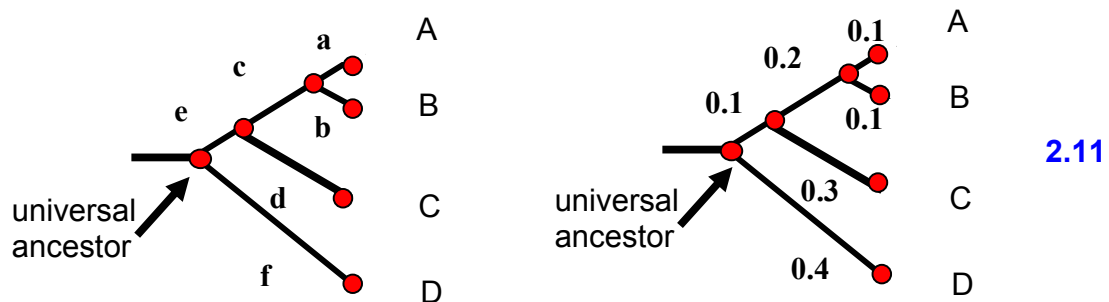
(assuming a PAM250 log odds matrix and gap penalty of -10)

The above example is somewhat of a simplification because in practice when building a multiple sequence alignment - a *weighted sum of pairs* score is more often than not used. The idea behind weighting is to attempt to down play the significance of the closely related sequences in the data set. This modification of the sum of pairs criterion is helpful when there are many closely related sequences in a data set compared to the number of

more distantly related sequences. In this situation, closely related sequences will dominate the sum of pairs score and have a greater influence on the positioning of gaps. Consequently, distinctive (but poorly represented) sequences tend not to be well aligned.

To obtain the “weights” that are used to modify the sum of pairs criterion a *weighted rooted bifurcating tree* is needed. The details of how such a tree is obtained is left until **Chapter 4** when we describe tree building methods. In principle though, pairwise alignments are first constructed for all sequences in your data set. The scores for each of these is then regarded as a measure of genetic dissimilarity between the sequences in each pair – in one sense a quantitative measure of the evolutionary events needed to transform one sequence into another. You will learn in **Chapter 4** that given a table or matrix of differences between all sequence pairs – such values can be used to build a bifurcating weighted tree. Some assumptions are then usually added to root this tree. Given such a tree it is then possible to calculate weights for each of the sequences by assigning weights to the sequences so that the weights reflect the evolutionary distinctiveness of each of the sequences in the tree. Essentially, the weighting takes into account (a) how diverged a particular sequence is from other sequences, and (b) how diverged each is from the root of the tree - i.e. the universal ancestral sequence. The more diverged a sequence is from the root of the tree and the more diverged it is from other homologous sequences, the higher its weight and the more important is its contribution in deciding gap placements.

[Figure 2.11](#) illustrates how weights can be calculated.



The weight for a particular sequence is calculated by considering the path from the tip of the tree (where that particular sequence is) back to the universal ancestor. The length of the branches (identified by small case letters as well as corresponding values in [Figure 2.11](#)) on this path as well as the number of sequences that derive from each branch are considered in calculating the weight. The weight for sequence A = $1(a) + \frac{1}{2}(c) + \frac{1}{3}(e) = 0.1 + \frac{1}{2}(0.2) + \frac{1}{3}(0.1) = 0.233$. The weight for sequence B = $1(a) + \frac{1}{2}(c) + \frac{1}{3}(e) = 0.1 + \frac{1}{2}(0.2) + \frac{1}{3}(0.1) = 0.233$. The weight for sequence C = $1(d) + \frac{1}{3}(e) = 0.3 + \frac{1}{3}(0.1) = 0.33$. The weight for sequence D = $1(f) = 0.4$. You will notice that the two most closely related sequences (A and B) have the smallest weights. The example in [Figure 2.12](#) incorporates these weights to give a weighted sum of pairs score for a particular site pattern:

| | | |
|------------|-------|-------------|
| sequence A | ..R.. | |
| sequence B | ..R.. | |
| sequence C | ..K.. | 2.12 |

The sum of pairs score = $R_{seqA}:R_{seqB} + R_{seqA}:K_{seqC} + R_{seqB}:K_{seqC} = (5+3+3) = 11$
The weighted sum of pairs score = $R_{seqA}:R_{seqB} w_A w_B + R_{seqA}:K_{seqC} w_A w_C + R_{seqB}:K_{seqC} w_B w_C$
= $(5 \times 0.23 \times 0.23) + (3 \times 0.23 \times 0.33) + (3 \times 0.23 \times 0.33) = 0.72$
where w_A = weight associated with sequence A,
 w_B = weight associated with sequence B
 w_C = weight associated with sequence C.

2.6 Different multiple sequence alignment strategies

Simultaneous multiple sequence alignment is too computationally intensive for this method to be generally used as an approach for constructing alignments. The computational problem becomes worse with every sequence and residue position added to the data set since increasing the product of sequence length and number of sequences results in an exponential increase in the number of possible places to consider putting gaps in the alignment. To get round this problem, but still use dynamic programming, various strategies have been implemented which remove from consideration the need to evaluate every possible position for placing gaps. The basic idea is that, if scores need only be calculated for the most likely alignments, then this will make the problem of finding a good multiple sequence alignment tractable. One strategy to reduce computation that has been used in both the development of sequence alignment and tree building methodologies is the mathematical technique of *branch and bound*. For slightly

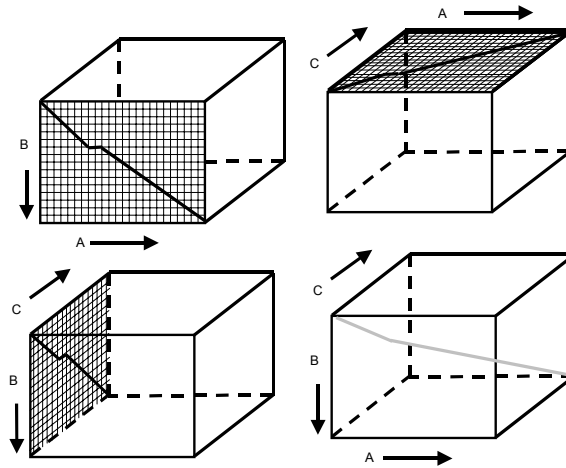
bigger data sets than can be handled by exhaustive methods, branch and bound can guarantee that you find the best mathematical solution (e.g. multiple sequence alignment or tree) for a specified objective function. Unfortunately, for bigger data sets – often of the size you will work with, methods that best guess the solution space need to be used. These latter - *heuristic* - methods cannot guarantee that you will find the optimal mathematical solution. However, various bench mark tests have shown that in practise they can perform well under certain conditions.

Heuristic methods are widely used both for building alignments and phylogenetic trees. Their implementation sacrifices mathematical rigour for speed of execution, and algorithms are often laced with a large dollop of biological cunning. Whilst heuristic methods cannot guarantee to find the optimal solution, it is worth keeping in mind that a good solution found in a reasonable time is some times better than a rigorous solution never found. Also, another thing to keep in mind is that the process of optimising some mathematical objective function does not necessarily guarantee that you will find the most meaningful biological solution. The reason for this is that the evolutionary properties of sequence data are sometimes more complex than can be accommodated for using simple mathematical algorithms. We describe some commonly used methods.

2.7 MSA

MSA is a simultaneous global multiple sequence alignment program, that is based on a ***branch-and-bound*** principle developed by Carrillo and Lipman and on a heuristic implementation of Gupta and colleagues. Carrillo and Lipman recognised that the solution space in which the optimal msa could be found must be bounded by the projections of the pairwise alignments into this space. [Figure 2.12](#) illustrates this concept for the alignment of three sequences A,B,C. In principle, this projection identifies and restricts the search or solution space in which the optimal alignment is guaranteed to lie. Once this space is identified, the set of possibilities in it are evaluated to find the optimal alignment.

2.13



How it works

Gupta and colleagues approximated this bounded space by implementing a heuristic multiple sequence alignment procedure. That is, they suggested using the score for a multiple sequence alignment made using a heuristic method as an upper bound for restricting the search space for the optimal msa alignment (i.e. assume that the score for the optimal alignment will be as good or better than that of the heuristic alignment) – so you do not need to consider alignments and possible gap placements that will give scores worse than the heuristic alignment). In MSA the scoring of alignments is done using a weighted sum of pairs criterion. On desktop computers MSA may simultaneously align up to 6-10 sequences of length 1000 residues.

2.8 DCA

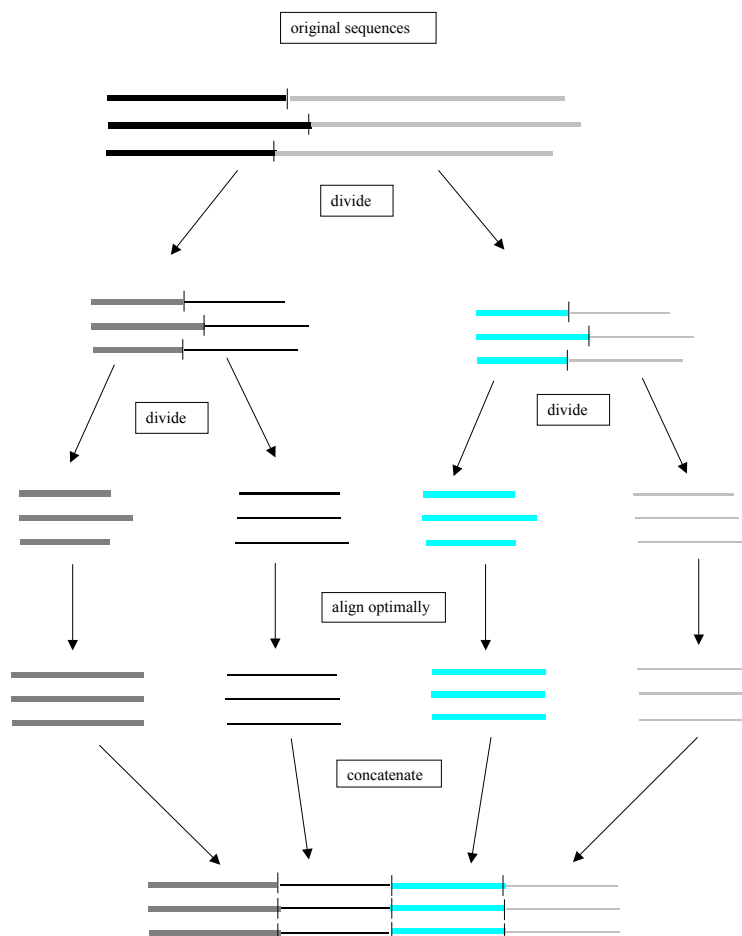
DCA is a method that implements MSA to produce a simultaneous global multiple. It is able to simultaneously align more sequences than MSA through the introduction of an additional heuristic step. This involves aligning only short sections of the sequences simultaneously rather than attempting to align all residues across the complete length of the sequences. The short aligned sections are eventually concatenated to produce a global msa.

How it works

DCA begins by examining an alignment of each pair of sequences and cutting it into two pieces - the part of the alignment 5' to the cut is called the prefix region and the part of the alignment 3' to the cut is called the suffix region. For every aligned pair of sequences there is a cut point such that [the alignment score of the prefix region] + [the alignment

score of the suffix region] – [the alignment score of the complete pairwise alignment] = 0. This cut point will be somewhere near the midpoints of the two sequences. However, the best position for the cut point can differ for different pairs of sequences (in the same way that the placement of gaps in one sequence aligned to a second sequence may not be in the best place when aligned to a third sequence and so on). DCA uses a type of sum of pairs score criterion (cut point score) to find the best cut point considering all pairs of sequences. Once this cut point is determined, if the prefix and suffix regions are sufficiently short enough in length then MSA will simultaneously align all the suffix sequences to each other and align all the prefix sequences to each other. If the computational complexity of this alignment task is still too great, then the sequences are cut again and the computational complexity of aligning prefix and suffix sequences is re-evaluated. The cutting procedure continues until lengths of sequence are short enough to be aligned by MSA in a reasonable time. The aligned subregions are then concatenated to produce a multiple sequence alignment ([Figure 2.14](#)).

2.14



2.9 CLUSTALX

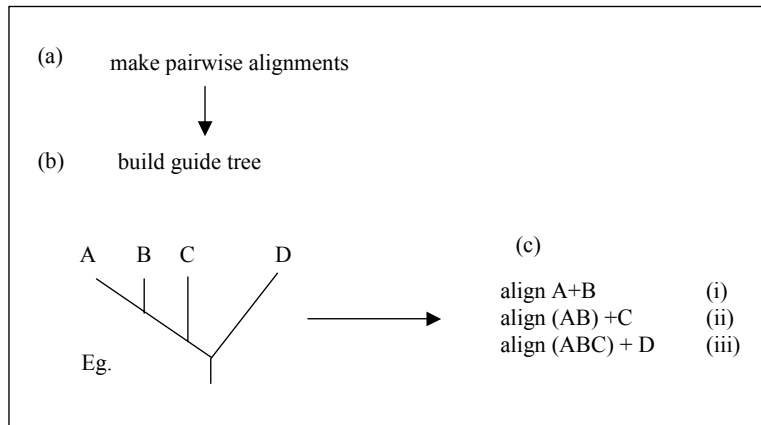
A widely used alternative to simultaneous multiple sequence alignment is to progressively align sequences and groups of sequences, building up a multiple sequence alignment by a progressive clustering procedure. One of the earliest alignment programs of this type aligned sequences one at a time to a growing alignment. Latter approaches specified a phylogenetic guide tree which then determined the order sequences and groups of sequences were aligned. One of the most sophisticated and commonly used programs that implements a strategy of progressive multiple sequence alignment is CLUSTAL. It produces a multiple sequence alignment in three steps:

- (i) For all sequences pairwise alignments and their scores are calculated
- (ii) These scores are used to build a guide tree using *Neighbor Joining*
- (iii) This tree guides the construction of a multiple sequence alignment using a number of heuristic protocols and a weighted sum of pairs criterion.

How it works

Either a fast approximate method or a slower more sensitive (dynamic programming) method can be selected to align all pairs of sequences. Alignment scores for pairs are then calculated and interpreted as measures of evolutionary distance. These distance values are stored in a table which we will refer to as a d_{ij} matrix – where d_{ij} simply means a measure of difference between sequence i and sequence j . Pairwise alignments with low similarity scores, arising because of many mismatches and/or the presence of gaps – have high d_{ij} values. Pairwise alignments with high similarity scores have low d_{ij} values. A weighted phylogenetic tree is calculated from the d_{ij} matrix using the clustering algorithm of *Neighbor Joining* (**Chapter 4**). This tree is rooted at the midpoint of the longest path through the tree. This midpoint is assumed to indicate the position of the universal ancestral sequence and thus the direction that evolution has gone in the tree. The tree provides the basis for calculating both the sequence weights (as done in [Figure 2.11](#) and [2.12](#)) as well as to the order for making progressive pairwise alignments of sequences and groups of sequences. The latter is determined by the relative lengths of branches that

separate taxa in the guide tree. The most phylogenetically similar sequences are aligned first, and the most dissimilar aligned last. In summary, using CLUSTAL a multiple sequence alignment is constructed through the progressive pairwise alignment of sequences, of individual sequences to groups of sequences and groups of sequences to other groups of sequences. The method is illustrated in [Figure 2.15](#).



2.15

CLUSTALX unlike current implementations of MSA and DCA allows greater flexibility of some parameters used in multiple sequence alignment. These features of the method are appealing because they recognise more of the biological complexity of sequence data. CLUSTALX algorithms uses different substitution scoring matrices for sequences of different degrees of divergence (e.g. the pairwise alignment between very divergent sequences or groups will use a BLOSUM30 or PAM350 log odds matrix whilst between very closely related sequences it will use a BLOSUM80 or PAM20 matrix). They also use complex and dynamic gap penalties since their use has been found to help identify the correct alignment of structural motifs. Gap opening penalties are scaled based on the relative sequence divergence of sequences - making gaps more likely in more distantly related sequences. If these penalties are too high then the alignments may contain too few gaps. Conversely if they are too low, there will be too many gaps. Gap opening and extension penalties are modified for the absolute and relative length of the sequences, since empirical studies have shown that the longer a sequence the more likely it is to have gaps. Additionally, these penalties are also modified at each pairwise alignment step so

that gaps become more or less likely at different sequence positions. The placement of gaps is encouraged at positions at which there are already gaps and discouraged in regions of close proximity (within 8 residues) to such gaps. Gaps are favoured in regions here there are runs of hydrophilic residues (e.g. 5 or more residues of the type D, E, G, K, N, Q, P, R, S) as these usually indicate loop regions in protein structures which often favour the presence of indels. Where no run of hydrophilic residues occur, the gap opening penalty is scaled depending on the residue(s) present at every sequence position. The scaling factor used is empirically derived from structure based analyses and is based on the frequency of each of the 20 amino acid residues most often associated with gap regions.

2.10 RRP (the DNR iterative strategy)

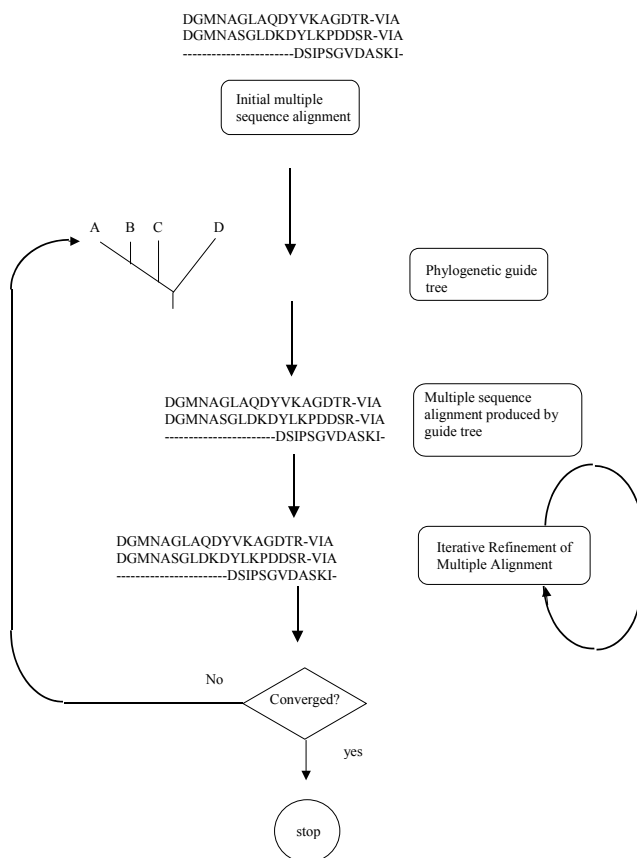
A weakness of progressive alignment is that when gaps are placed at each pairwise alignment step, they are not revised – they are fixed in the growing alignment. This feature - *once a gap always a gap* - is a defining characteristic of progressive alignment. It is a property that significantly reduces the computation time needed for implementing a weighted sum of pairs criterion. It is also a weakness of the progressive alignment strategy and one which has prompted the development of *non-stochastic iterative alignment* methods such as used in PRRP. The general concept of iteration as applied to multiple sequence alignment involves re-aligning sequences or groups of sequences contained in the initial multiple sequence alignment – the idea being to revisit gap placement choices and alter aligned residues that may have been wrongly placed. PRRP uses a double nested iterative strategy.

How it works

A multiple sequence alignment is first generated, perhaps by a progressive alignment procedure such as CLUSTAL. Alignment scores for each pair of sequences contained in the multiple sequence alignment are calculated and stored in a d_{ij} matrix. The matrix is used to construct a phylogenetic guide tree (possibly by Neighbor Joining). The branch lengths on this tree are used to provide weights for each sequence in the initial multiple

sequence alignment. The sequences in the initial multiple sequence alignment are then re-aligned using a random order pairwise alignment procedure. To do this, a subgroup of sequences is extracted from the initial alignment and then, using the sequence weights calculated for each sequence, this subgroup of sequences is realigned to the remaining group of sequences in a pairwise manner. This step of random group selection and realignment can be repeated numerous times. Once this is done, a d_{ij} matrix is recalculated for the msa and used to build a phylogenetic guide tree. This tree is used to recalculate sequence weights, and to guide pairwise realignment of sequences, a sequence to a group of sequences as well as groups of sequences ([Figure 2.16](#))

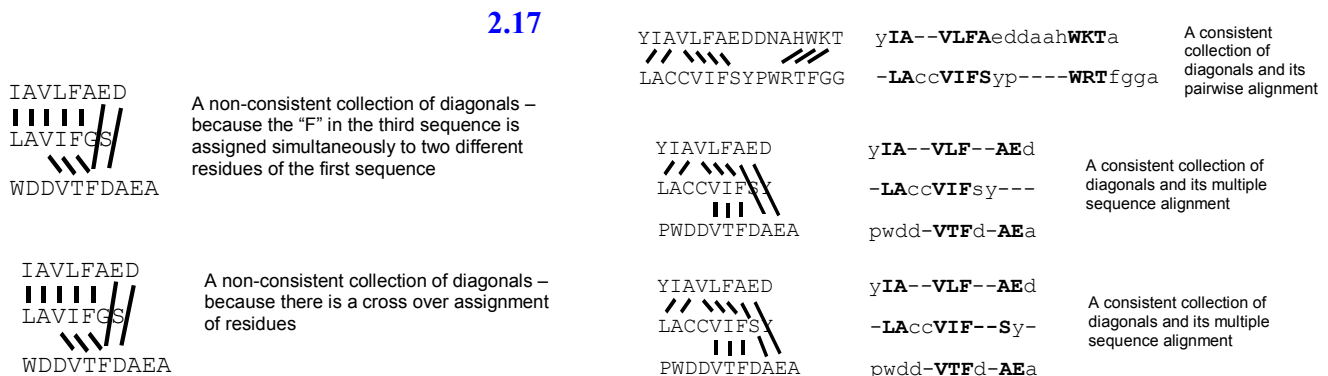
During alignment and re-alignment the weighted sum of pairs score for the multiple sequence alignments are tracked. Once the alignment score no longer improve, the process is terminated and the multiple sequence alignment with the best score is accepted.



2.16

2.11 DIALIGN

The multiple sequence alignment methods discussed above – which are based on dynamic programming and global alignment - are able to produce biologically meaningful alignments when the sequences are globally related, contain few insertions and deletions and are not separated by large numbers of substitutions. However, often distantly related homologues share only isolated regions of similarity. In these cases, it may be less meaningful to align homologues end to end. Motivated by this problem, DIALIGN implements a different strategy using a criterion called *consistency* ([Figure 2.17](#)) and sequence similarity to identify evolutionary conserved regions of sequences. Regions of a sequence are called consistent if the ends of aligned sections are non overlapping.

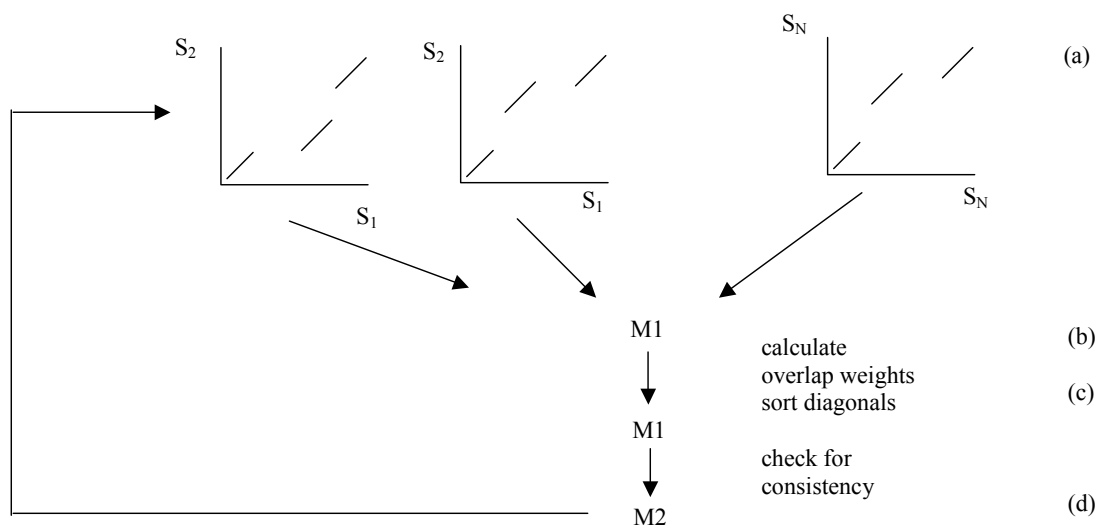


Since there could potentially be many consistent multiple sequence alignments for the same set of sequences, an objective function is used which will identify the set of non overlapping conserved blocks that have the highest similarity score.

How it works

An overview of the method is given in [Figure 2.18](#). All possible diagonals (ungapped pairwise aligned regions above a threshold summed log odds score) are identified between all pairs of sequences to be aligned. A weighted score is then assigned to each of these diagonals based on the evolutionary significance of each (calculated by considering

the probability of observing such a diagonal or pairwise alignment score by chance) and the extent that the diagonals overlap with other diagonals. Diagonals are given higher weights if they preserve **motifs (exact matches)** of residues in more than two sequences. All the diagonals are then ranked based on their relative scores. Starting with the diagonals of highest score, the pairwise alignments that these correspond to are incorporated into a growing alignment. Only diagonals that are consistent with diagonals already in the growing alignment are added. This process continues until all possible diagonals have been added. At this point, additional diagonals are next sought amongst the fragments of sequences not yet aligned. These diagonals must be consistent with the regions already aligned. They are ranked in size and added in as done previously, the process is continued until no additional diagonals above a threshold score can be found. Finally, the program adds in gaps so as to arrange the connected diagonals so that homologous residues are lined up with each other.



2.18

2.12 The relative performance of different methods

Information on conserved protein structures provides a means for comparing methods to determine whether or not, and under what conditions, alignment methods produce results that are *biologically meaningful*. In this respect, McClure and colleagues began a trend by asking the question whether or not particular methods were able to detect an ordered set of expected structural motifs. Databases that contain structural alignments such as

BALiBASE (www-igbmc.u-strasbg.fr/BioInfo/BALiBASE/) provide an important resource for this purpose.

Results from recent comparative studies have shown that the best choice of alignment program depends greatly on the sequences to be aligned and that no single alignment strategy works well with every dataset – the degree of divergence, size and distribution of indels amongst the sequences is important to consider. For example, in one recent and comprehensive study by Thompson and colleagues, the global methods implemented in PRRP and CLUSTALX were found to outperform other global and also local methods when sequences were equidistant or contained only one or more highly divergent sequences. In contrast, when the sequence data sets contained large C or N terminal extensions local construction methods outperformed the global methods. However, when sequences had large internal indels although the local method DIALIGN performed best out of all methods trialled, other local methods performed poorly. No methods seem to cope well with repeats, and when there are strings of low complexity residues.

1.13 Some words of advice from different researchers

Cedric Notredame emphasizes that it is impossible to generalize about the performance of a given method, and that consideration of the sequences to be aligned is very important. He cautions against “blindly aligning all homologues available” – which will result in alignments that are “slow to compute and hard to analyse”. He points to the problem of using objective functions such as the weighted sum of pairs criterion as they may not necessarily correctly align expected structural motifs. Unfortunately, very few packages incorporate 3D structural information, and a proper tool is still lacking for simultaneous alignment of sequences and structures. Although, when using the weighted sum of pairs function, weighting minimizes the effect of similar or highly correlated sequences, empirical results suggest that weighting is not entirely satisfactory, and overrepresented subgroups can dominate the alignment – the consequence being that less well represented sequences may be poorly aligned. Consideration for the choice of homologues to be aligned is important. It may well be worth aligning sequences with and

without particular homologues to investigate the effect of the presence of any potentially problematic homologues. The order in which sequences are aligned is also important. Essentially progressive alignment attempts to align the least divergent sequences first and to then sequentially add in the more diverged sequences. However, numerous authors have reported examples where the order that sequences are aligned has had a significant effect on phylogenetic reconstruction. Thus it may not always be prudent to simply accept the guide tree suggested by a progressive alignment program and to investigate for yourself the effect that different alignment orders have on the alignment of residues in your data set.

Other useful advice has been provided by Hickson and colleagues. These authors investigated a number of methods using a 12S rRNA data set – they found that all programs tested aligned the expected motifs for at least 1 set of parameters. However, the parameter values that worked well with one program were not optimal for another. Additionally they found that program default settings did not necessarily give the best results – the message being that optimal parameter values may need to be trialled by the researcher.

Others, such as Morrison and Ellison, have also stressed the importance of investigating parameter values when building alignments. On a protozoan 18S sequence data set they found that changing gap penalty parameters in CLUSTAL had a larger affect than choice of alignment program. This result differed from those of Hickson and colleagues in respect of the relative importance of alignment method and parameter optimisation. However, the differences in the density of taxon sampling, size and number of indels in their respective data sets studied may well account for their different findings.

Less clear at the moment is the relative importance of different substitution scoring matrices in multiple sequence alignment. Gotoh recently reported results from comparative analyses of global multiple sequence alignment methods benchmarked against structural alignments. His results suggest that if iterative methods such as the

DNR method are employed, alignment is less sensitive to choice of substitution matrix and gap penalties particularly when aligning highly diverged sequences.

There is an important issue concerning alignment uncertainty, and what to do about it. Most alignment packages do not indicate the uncertainty but they will often give a measure of alignment quality – such as the weighted sum of pairs score. Studies by Arnt von Haseler and colleagues on pairwise alignment suggest the use of Monte Carlo methods (**Chapter 4**) could be very useful for investigating alignment uncertainty. However, as yet such approaches are not implemented in the context of multiple sequence alignment. There is a problem to know what to do about ambiguously aligned regions. Morrison and Ellison suggest building alignments using different parameters to identify ambiguous regions and to down weight these regions. However, some programs can misalign even well conserved motifs, particularly when they are adjacent to indels. Knowledge of secondary and tertiary structures may be helpful to delimit choice of parameter values that are to be investigated for evaluating alignment uncertainty. In principle conserved structures can be used to anchor and provide a framework for alignment of other regions. The editor MACAW uses this principle. Other editors are also helpful for building and studying alignments (e.g. freely available ones include: Se-AL2 and BioEdit). The editor of Castresana (Gblocks) is particularly helpful for obtaining conserved blocks of residues for subsequent phylogenetic analysis.

1.14 Further reading

Castresana J. (2000) Selection of conserved blocks from multiple alignments for their use in phylogenetic analysis. *Mol. Biol. Evol.* 17, 540-552

Gotoh O. (1996) Significant improvement in accuracy of multiple protein sequence alignments by iterative refinement as assessed by reference to structural alignments. *J. Mol. Biol.* 264, 823-838

Hickson R.E., Simon C. and Perry S.W. (2000) The performance of several multiple sequence alignment programs in relation to secondary-structure features for an rRNA sequence. *Mol. Biol. Evol.* 17, 530-539

- Hickson R.E., Simon C., Cooper A., Spicer G.S., Sullivan J. and Penny D. (1996) Conserved sequence motifs, alignment, and secondary structure for the third domain of animal 12 rRNA. *Mol. Biol. Evol.* 13, 150-169
- Kjer K. (1995) Use of rRNA secondary structure in phylogenetic studies to identify homologous positions: an example of alignment and data presentation from frogs. *Mol. Phylogenet. Evol.* 4, 314-330
- Lassman T. and Sonnhammer L.L. (2002) Quality assessment of multiple alignment programs FEBS lett. 529, 126-130
- Lake J. A. (1991). The Order of Sequence Alignment Can Bias the Selection of Tree Topology. *Mol. Biol. Evol.* 8(3), 378-385
- McClure M.A., Vasi T.K. and Fitch W.M. (1994) Comparative analysis of multiple protein-sequence alignment methods. *Mol. Biol. Evol.* 11, 571-592
- Metzler D., Fleißner R., Wakolbinger A. and von Haeseler A. (2001) Assessing Variability by Joint Sampling of Alignments and Mutation Rates *J. Mol. Evol.* 53, 660-669
- Morgenstern B. (1999) DIALIGN2: improvement of the segment-to-segment approach to multiple sequence alignment. *Bioinformatics* 15, 211-218
- Morrison D.A. and Ellis J.T. (1997) Effects of Nucleotide sequence alignment on phylogeny estimation: A case study of 18S rDNA of Apicomplexa. *Mol Biol Evol* 14, 428-441
- Notredame C. (2001) Recent progress in multiple sequence alignment: a survey. *Pharmacogenomics* 3, 1-14
- Thompson J.D., Higgins D.G. and Gibson T.J. (1994) CLUSTALW: improving the sensitivity of progressive alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acid Res.* 22, 4673-4680
- Thompson J.D., Plewniak F. and Poch O. (1999) A comprehensive comparison of multiple sequence alignment programs. *Nucleic Acids Res.* 27, 2682-2690
-