

## Chapter 2

# SEQUENCE ALIGNMENT

### 1. INTRODUCTION

Sequence alignment is not only the essential first step in molecular phylogenetics, quantification of substitution patterns, and dating of speciation and gene duplication events, but also a powerful tool for identify mutations leading to genetic diseases. For example, aligning the  $\beta$ -hemoglobin gene sequence from one type of  $\beta$ -thalassemia against the normal  $\beta$ -hemoglobin gene immediately reveals an insertion of T at site 79 (Figure 2-1).

```

          10      20      30      40      50      60
-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
Normal  ATGGTGCACCTGACTCCTGAGGAGAAGTCTGCCGTTACTGCCCTGTGGGGCAAGGTGAACGT
Thalas. ATGGTGCACCTGACTCCTGAGGAGAAGTCTGCCGTTACTGCCCTGTGGGGCAAGGTGAACGT
*****
          70      80      90      100     110     120
--|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
Normal  GGATGAAGTTGGTGGT-GAGGCCCTGGGCAGGTTGGTATCAAGGTTACAAGACAGG.....
Thalas. GGATGAAGTTGGTGGTTGAGGCCCTGGGCAGGTTGGTATCAAGGTTACAAGACAGG.....
*****
```

Figure 2-1. Alignment between the normal and mutant  $\beta$ -hemoglobin gene sequences.

The insertion creates an inframe stop codon and results in a truncated  $\beta$ -hemoglobin protein. When the  $\beta$ -hemoglobin locus is heterozygous with one

mutant and one normal gene, the carrier is said to have  $\beta$ -thalassemia minor and can be easily detected by gel electrophoresis because the mutant  $\beta$ -hemoglobin, being much shorter than the normal, would migrate much faster on the gel under an electric field.

This chapter covers (1) pairwise global and local alignment by dynamic programming with different scoring schemes, from the simplest scoring scheme with match/mismatch scores and gap penalties all specified by constants, to more useful scoring scheme with match/mismatch scores specified by a similarity matrix and gap penalties specified by the affine function, (2) profile alignment between one sequence and a set of aligned sequences which is essential for practical implementation of multiple sequence alignment, and (3) multiple alignment that is reduced to pair-wise alignment and profile alignment by using a guide tree. Most textbooks on bioinformatics omit the affine function, and no textbook I know of includes any detailed explanation of profile alignment.

Dynamic programming algorithms constitute a general class of algorithms not only used in sequence alignment, but also in many other applications. For example, the Viterbi algorithm and the forward algorithm used in hidden Markov models (HMM) are also dynamic programming algorithms. We will cover HMM in great detail and illustrate with numerical examples latter. Learning the dynamic programming algorithms used in sequence alignment paves the way for more advanced applications in latter chapters.

Sequence alignment methods, especially those for obtaining multiple alignments, are central to molecular biology, evolution and phylogenetics. One of the global sequence alignment programs, ClustalW (Higgins and Sharp, 1988; Thompson *et al.*, 1994) is probably the second most used bioinformatics tool next to the BLAST suite of programs (Altschul *et al.*, 1990; Altschul *et al.*, 1997).

## 2. PAIRWISE ALIGNMENT

Given two strings  $S (=s_1s_2\dots s_n)$  and  $T (=t_1t_2\dots t_m)$ , a pairwise alignment of  $S$  and  $T$  is defined as an ordered set of pairings of  $(s_i, t_j)$  and of gaps  $(s_i, -)$  and  $(-, t_j)$ , with the constraint that the alignment is reduced to the two original strings when all gaps in the alignment are deleted. A prefix of  $S$ , specified here as  $S_i$ , is a substring of  $S$  equal to  $s_1s_2\dots s_i$ , where  $i \leq n$ .

An optimal alignment is operationally defined as the pairwise alignment with the highest alignment score for a given scoring scheme. For this reason, an optimal alignment is meaningless without the specification of the scoring scheme.

Alignment by dynamic programming guarantees that the resulting alignment is the optimal alignment or one of the equally optimal alignments. We will first illustrate the global pairwise alignment (Needleman and Wunsch, 1970) followed by local pairwise alignment (Smith and Waterman, 1981b). Local sequence alignment is for searching local similarities between sequences, e.g., homeobox genes which are not similar globally but all share a very similar homeodomain motif. The best known algorithm for local alignment is Smith and Waterman (1981b).

Here we will first learn a simple dynamic programming algorithm for pairwise alignment using a simple scoring scheme with constant gap penalty. This is then extended in two ways, first by introducing a similarity matrix to replace match and mismatch scores, and second by introducing the affine function to better approximate the origin of the insertion and deletion during sequence evolution.

## 2.1 Pairwise alignment with constant gap penalty

### 2.1.1 Global alignment

Suppose we want to align two sequences S and T with S = ACGT and T = ACGGCT. A simple scoring scheme is used with a constant gap penalty (G) of -2, a match score (M) of 2 and a mismatch score (MM = -1). Global alignment with the dynamic programming approach is illustrated numerically in Figure 2-2. One sequence of the two sequences occupies the top row and will be referred to hereafter as the row sequence (sequence S in our example). The other sequence occupies the first column and will be referred to hereafter as the column sequence (sequence T in our example). Based on these two sequences, two matrices are computed. The first is the scoring matrix to obtain the alignment score, with the dimensions (n+1, m+1). The second is the backtrack matrix needed to obtain the actual alignment, with the dimensions (n,m). In Figure 2-2, the two matrices are superimposed, with the scoring matrix being the numbers and the backtrack matrix being made of arrows. The backtrack matrix is sometimes called the traceback matrix. However, the word traceback is marked by an annoying red wavy line in Microsoft WORD. So my choice of the two is obvious.

A value in row  $i$  and column  $j$  in the scoring matrix is the alignment score between a prefix of S and a prefix of T, i.e.,  $S_j$  and  $T_i$ . This will become clear later.

The first row and the first column of the scoring matrix is filled with  $i \times G$  (where  $i = 0, 1, \dots, n$ ) and  $j \times G$  (where  $j = 0, 1, \dots, m$ ), respectively. They represent consecutive insertion of gaps. For example, the number -8 in the

last cell of the first row of the scoring matrix implies the following alignment with four consecutive gaps in the column sequence and an alignment score of -8:

```
ACGT
----
```

		A	C	G	T
	0	-2	-4	-6	-8
A	-2	↖ 2	← 0	← -2	← -4
C	-4	↑ 0	↖ 4	← 2	← 0
G	-6	↑ -2	↑ 2	↖ 6	← 4
G	-8	↑ -4	↑ 0	↖↖ 4	← 5
C	-10	↑ -6	↑ -2	↑ 2	↖ 3
T	-12	↑ -8	↑ -4	↑ 0	↖ 4

Figure 2-2. . Computation involved in obtaining the scoring and the backtrack matrices (superimposed) with the match score equal to 2, mismatch -1 and the gap penalty equal to -2. There are two equally optimal alignments each with an alignment score of 4.

Similarly, the number of -12 in the last cell of the first column of the scoring matrix implies the following alignment with six consecutive gaps on the row sequence and an alignment score of -12:

```
-----
ACGGCT
```

The first cell where we need to compute the score is the one corresponding to the first character of S and T, i.e., the cell with a value of 2. To compute the value for the cell, we need values in three other cells, one to its left, one above it and one to its upleft, with their cell values designated as L, U and UL, respectively. Note that the cell has an upleft (UL) value of 0, a top value (U) equal to -2, and a left value (L) equal to -2. The following three values are calculated:

$$\begin{aligned} \text{DIAG} &= \text{UL} + \text{IF}(\text{Corresponding characters match, M, MM}) = 0 + 2 = 2 \\ \text{LEFT} &= \text{L} + \text{G} = -2 + (-2) = -4 \\ \text{UP} &= \text{U} + \text{G} = -2 + (-2) = -4 \end{aligned}$$

The IF function above takes the value of M if the two corresponding nucleotides match, or MM if they do not. The maximum of these three values is DIAG, i.e., 2, which was entered as the first computed element in the scoring matrix. The cell is also filled with an upleft arrow because DIAG is the maximum of the three values. If LEFT (or UP) happened to be the maximum of the three, we would have put a left-pointing (or up-pointing) arrow in the corresponding cell in the backtrack matrix.

The computation is from left to right and from top to bottom. For the second cell, the maximum of the three values is LEFT (= 0), and the corresponding cell in the backtrack matrix is filled with a left-pointing arrow. We continue the computation to the bottom right cell, with the final value in the bottom-right value equal to 4. This is the alignment score. You may note that the cell corresponding to the nucleotide G in the row sequence and the second G in the column sequence is special with two arrows. You will find that the DIAG and UP values are both equal to 4 in this cell. Hence both the upleft and the up-pointing arrows in this cell. Such a cell implies the existence of equally optimal alignments.

The aligned sequences are obtained directly from the backtrack matrix. We start from the bottom-right cell and follow the direction of the arrow in the cell. The upleft arrow in the bottom-right cell means that we should stack the two corresponding nucleotides (T and T) in the row and column sequences (Figure 2-3). Note that you would be stacking the two corresponding nucleotides regardless of whether they are the same or different as long as an upleft arrow is in the cell. A left-pointing or up-pointing arrow in the cell means a gap in the column sequence or row sequence, respectively.

(a)	(b)
654321	654321
ACG--T	AC-G-T
ACGGCT	ACGGCT

Figure 2-3. The protocol of obtaining the sequence alignment by following the backtrack matrix. The numbers in the first row show the order of obtaining the alignment site by site from the last to the first (i.e., backtracking).

The upleft arrow in the bottom-right cell leads us to the cell containing an up-pointing arrow, meaning a gap in the row sequence, i.e., we stack a gap

character “-” over the corresponding nucleotide (C) in the column sequence (Figure 2-3). This up-pointing arrow brings us to the special cell with two arrows, one pointing upleft and the other up (Figure 2-2). This leads to alternative construction of the sequence alignment. If we choose the up-pointing arrow, we will stack a gap character over the corresponding nucleotide (G) in the column sequence and proceed to the cell with a value of 6 and an upleft arrow. This ultimately leads to the sequence alignment in Figure 2-3a. Alternatively, we may choose to follow the upleft arrow and stack the two nucleotides (G in both sequences) as shown in Figure 2-3b. This ultimately leads to the alternative sequence alignment in Figure 2-3b.

Both alignments in Figure 2-3 have four matches, two gaps, and zero mismatch. So the alignment score is  $4 \times M + 2 \times G + 0 \times (MM) = 4$ , which we already know after completing the scoring matrix whose bottom-right cell contains the alignment score.

Recall that each cell with a score and an arrow specifies an optimal alignment between a prefix of S and a prefix of T, i.e.,  $S_j$  and  $T_i$ . For example, the first cell with a calculated value of 2 and an upleft arrow specifies the optimal alignment of  $S_1$  (= ‘A’) and  $T_1$  (= ‘A’) with an alignment score of 2 (which is the match score). The next cell to the right, with a score of 0 and a left arrow, specifies the optimal alignment of  $S_2$  and  $T_1$  (The 0 score results from a match and a gap penalty):

```
AC
A-
```

The cell with a score of 4 and two arrows specifies two equally optimal alignments both with an alignment score of 4:

```
Alignment 1:
ACG-
ACGG
```

```
Alignment 2:
AC-G
ACGG
```

When sequences are long, there might be many equally optimal alignments and few computer programs would try to find and output all of them. Instead, only one path is followed, leading to the output of only one of the potentially many equally optimal alignments.

Dynamic programming guarantees that the resulting alignment is optimal given the scoring scheme. In other words, there is no alignment that can have

an alignment score greater than 4 given the two sequences and the scoring scheme of  $M = 2$ ,  $MM = -1$  and  $G = -2$ . However, an optimal alignment may change when the scoring scheme is changed. This is illustrated in Figure 2-4, where the use of scoring scheme 1 would result in Alignment 1 (with alignment score = 14) better than Alignment 2 (with alignment score = 12) but the use of scoring scheme 2 would lead to the opposite, with Alignment 2 (alignment score = 20) much better than Alignment 1 (alignment score = 9).

Alignment 1: ACCCAGGGCTTA	ACCCGGGCTTAG				
Alignment 2: ACCCAGGGCTTA-	ACCC-GGGCTTAG				
Scoring scheme 1: $M = 2$ , $MM = 0$ , $G = -5$					
Scoring scheme 2: $M = 2$ , $MM = -1$ , $G = -1$					
Alignment	Match	Mismatch	Gap	Score1	Score2
1	7	5	0	14	9
2	11	0	2	12	20

Figure 2-4. Illustration of the dependence of optimal alignment on scoring scheme. Score1 and Score2 in the bottom table refer to Scoring scheme 1 and Scoring scheme 2, respectively.

Because there is no objective way of choosing the right scoring scheme, it is therefore important to keep in mind that sequence alignment is a method of data exploration instead of an analytical method that will lead to a single best solution. For this reason, nearly all computer programs for sequence alignment allow the user to try various scoring schemes and post-alignment manual editing.

### 2.1.2 Local alignment

Local sequence alignment (Smith and Waterman, 1981b) is similar to global alignment presented above, with only three major differences. First, the first row and the first column of the scoring matrix is filled with zero instead of  $i \times G$ . Second, whenever the cell value becomes negative (i.e., the maximum of the three values is smaller than 0), the cell value is set to 0. Thus, when two sequences have a short but perfect local match, and little similarity elsewhere, the alignment score of the short but perfect match is not affected by the low similarity elsewhere. Third, because a local alignment can end anywhere in the matrix, we will not trace back from the cell in the bottom-right corner of the score matrix. Instead, we find the maximal score

in the matrix and trace back from that point until we reach a cell with a value of 0, which indicates the start of the local alignment.

In Chapter 1 we have already covered two widely used heuristic methods for local alignment, i.e., BLAST and FASTA. In Chapter 7 we will learn Gibbs sampler which is another method for searching local similarities and local alignment.

### 2.1.3 The simple scoring scheme needs extension

The simple scoring scheme that we have used has three major problems. First, transitions (i.e., substitutions between nucleotides A and G and between C and T) generally occur more frequently than transversions (When A or G is replaced by C or T). This suggests that we should not treat transitional differences and transversional differences with the same mismatch score. Instead, transitions should be penalized less than transversions. Second, there are often ambiguous bases in input the sequences, e.g., R for A or G and Y for C or T. An A-R pair is neither a strict match nor a strict mismatch, but has a probability of 0.5 being a match and a probability of 0.5 being a transition. The simple scoring scheme we have used cannot handle these problems, which necessitates the use of a similarity matrix.

The simple scoring scheme also has another, perhaps even more serious problem, caused by constant gap penalty. A biologist will complain loudly that an alignment method is wrong if it considers the two alternative alignments in Figure 2-3 as equally good, and would have chosen the alignment in Figure 2-3a as a better alignment. The simplest solution to this problem is to use what is called an affine function for gaps. In the following sections, we will learn these two extensions, first with a similarity matrix and second with an affine function.

## 2.2 Pairwise alignment with a similarity matrix

### 2.2.1 DNA matrices

One example of a similarity matrix is the “transition bias matrix” (Table 2-1) used in DAMBE (Xia, 2001; Xia and Xie, 2001b) for multiple alignment with a star tree. A star tree contains only one internal node with all leaves connected to this same internal node. The meaning of the top 4×4 matrix (bolded values in Table 2-1) is easy to understand. The first four diagonal values of 30 are equivalent to the match score, a mismatch score involving a transversion or a transition is -30 or 0, respectively, because



transitions in general occur much more frequently than transversions and consequently penalized less (Table 2-1). The rest of the matrix (Table 2-1) involves ambiguous codes specified in Table 2-2, according to the Nomenclature Committee of the International Union of Biochemistry (1985).

Table 2-1. A similarity matrix accommodating the transition bias frequently observed in nucleotide substitutions.

A	C	G	U	R	Y	M	W	S	K	D	H	V	B	N
<b>30</b>														
<b>-30</b>	<b>30</b>													
<b>0</b>	<b>-30</b>	<b>30</b>												
<b>-30</b>	<b>0</b>	<b>-30</b>	<b>30</b>											
15	-30	15	-30	15										
-30	15	-30	15	-30	15									
0	0	-15	-15	-8	-8	0								
0	-15	-15	0	-8	-8	-8	0							
-15	0	0	-15	-8	-8	-8	-15	0						
-15	-15	0	0	-8	-8	-15	-8	-8	0					
0	-20	0	-10	0	-15	-10	-5	-10	-5	-3				
-10	0	-20	0	-15	0	-5	-5	-10	-10	-10	-3			
0	-10	0	-20	0	-15	-5	-10	-5	-10	-7	-10	-3		
-20	0	-10	0	-15	0	-10	-10	-5	-5	-10	-7	-10	-3	
-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8

Table 2-2. IUB codes of nucleotides.

Code	Meaning	Complement
A	A	T
C	C	G
G	G	C
T/U	T	A
M	A or C	K
R	A or G	Y
W	A or T	W
S	C or G	S
Y	C or T	R
K	G or T	M
V	A or C or G	B
H	A or C or T	D
D	A or G or T	H
B	C or G or T	V
X/N	G or A or T or C	X
-	Gap (not G or A or T or C)	-

The coding scheme is often referred to as the IUB code or IUB notation. For example, R represents either A or G, so an A-R pair has a probability of 0.5 being an A-A match and a probability of 0.5 being an A-G transition. The corresponding score (=15 in Table 2-1) is consequently somewhere between

a perfect match and a transition. In contrast, Y stands for either C or T/U and an A-Y pair is always a transversion, with a score of -30 (Table 2-1).

### 2.2.2 Protein matrix

Amino acids differ from each other in volume, charge, polarity and many other properties (Figure 2-5), and amino acid residues in a protein confer to the protein different properties. Proteins with long half-life (>1 day) typically have glycine, valine or methionine at their N-terminus, whereas those with short half-life (a few minutes) typically have positively charged residues (arginine, lysine) at their N-terminus. A small amino acid residue such as glycine and alanine at the penultimate site (the second amino acid site in the nascent peptide) allows the initiator methionine to be efficiently cleaved (Moerschell *et al.*, 1990). Amino acid replacements involving very different amino acids are generally selected against (Xia and Li, 1998). For this reason, a scoring scheme with only match and mismatch is rarely used for protein sequence alignment.

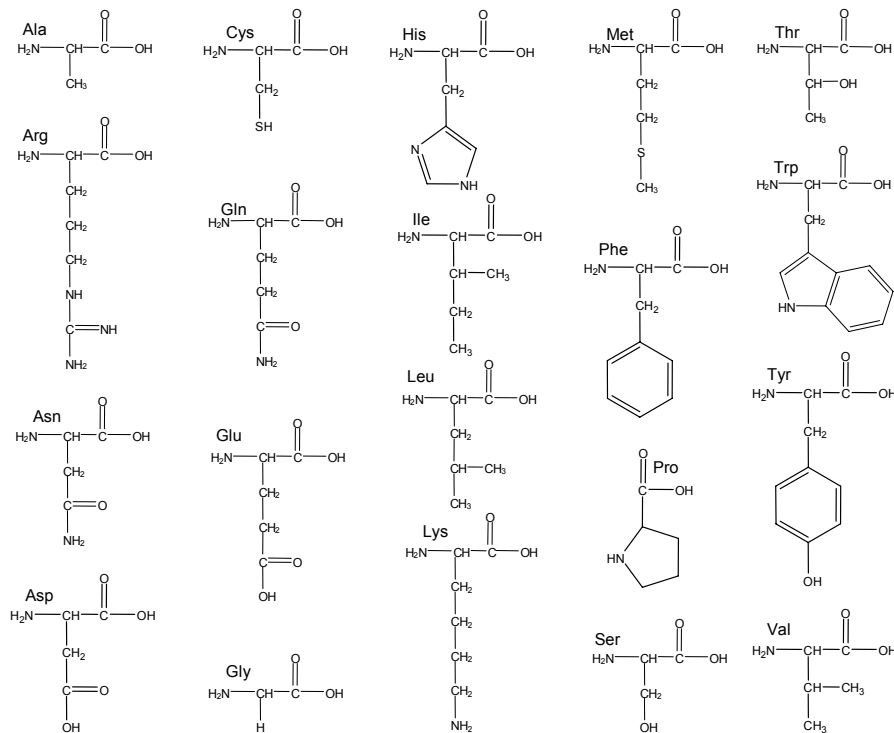


Figure 2-5. Structural formula of 20 amino acids.

A frequently used example to illustrate the effect of an amino acid being replaced by a different amino acid is the sickle-cell anemia. Sickle-cell anemia is caused by a single amino acid replacement in the  $\beta$ -chain of the human hemoglobin at the six position, with a glutamate residue replaced by a valine residue (Figure 2-6). Glutamate is negatively charged and hydrophilic, and tends to stay on the surface of the protein in the aqueous environment in the blood. In contrast, valine is a non-polar and hydrophobic residue and tends to shrink into the middle of the protein. The deformed protein molecules then form bundles and distort the red blood cell that carries them, resulting in the characteristic shape of a sickle (Figure. 2-6). It is generally true that amino acids of different polarity rarely replace each other (Xia and Li, 1998), whereas amino acids with similar polarity can replace each other quite frequently (Xia and Kumar, 2006).



Figure 2-6. Sickle-cell anemia is caused by a single amino acid replacement of a glutamate residue at the sixth position (Hb-A allele) by a valine residue (Hb-S allele). The mutant deformed hemoglobin molecules distort the red blood cell which progresses from the normal disk-like shape to the sickle-like shape.

Frequently used substitution matrices for protein sequences are of two types, the PAM matrix (Dayhoff *et al.*, 1978) and the BLOSUM matrix (Henikoff and Henikoff, 1992). The letter codes for amino acids proposed by the Nomenclature Committee of the International Union of Biochemistry (1985) are shown in Table 2-3. These codes are now universally adopted by the scientific community. I have omitted an introduction of these matrices because (1) the limit of page size of the book precludes the presentation of  $20 \times 20$  matrices and (2) an excellent introduction of these matrices appropriate for readers of this book is already available (Higgs and Attwood, 2004). In short, both PAM and BLOSUM matrices are derived from sequence alignment related proteins, with the former based on global alignment and the latter based on local alignment. The PAM1 matrix is based on comparisons of sequences with no more than 1% divergence and all other PAM matrices are derived from this PAM1 matrix. The requirement of proteins with no more than 1% divergence is necessary for reliable global alignment. The most frequently used BLOSUM matrix is BLOSUM 62

which is calculated from comparisons of sequences with no less than 62% divergence. BLOSUM xx matrix is based on sequence blocks with no less than xx% divergence, i.e., all BLOSUM matrices are based on observed alignments in contrast to the PAM matrices all derived from the PAM1 matrix. BLOSUM 62 is the default matrix in BLAST 2.0.

Table 2-3. IUB letter codes of amino acids.

1-letter	3-letter	Meaning	Codon <sup>(1)</sup>
A	Ala	Alanine	GCT,GCC,GCA,GCG
B		Asp or Asn	GAT,GAC,AAT,AAC
C	Cys	Cysteine	TGT,TGC
D	Asp	Aspartic	GAT,GAC
E	Glu	Glutamic	GAA,GAG
F	Phe	Phenylalanine	TTT,TTC
G	Gly	Glycine	GGT,GGC,GGA,GGG
H	His	Histidine	CAT,CAC
I	Ile	Isoleucine	ATT,ATC,ATA
K	Lys	Lysine	AAA,AAG
L	Leu	Leucine	TTG,TTA,CTT,CTC,CTA,CTG
M	Met	Methionine	ATG
N	Asn	Asparagine	AAT,AAC
P	Pro	Proline	CCT,CCC,CCA,CCG
Q	Gln	Glutamine	CAA,CAG
R	Arg	Arginine	CGT,CGC,CGA,CGG,AGA,AGG
S	Ser	Serine	TCT,TCC,TCA,TCG,AGT,AGC
T	Thr	Threonine	ACT,ACC,ACA,ACG
V	Val	Valine	GTT,GTC,GTA,GTG
W	Trp	Tryptophan	TGG
X	Xxx	Unknown	
Y	Tyr	Tyrosine	TAT,TAC
Z		Glu or Gln	GAA,GAG,CAA,CAG
*	End	Terminator	TAA,TAG,TGA

(1) assuming the standard genetic code.

### 2.3 Pairwise alignment with gap penalty specified by the affine function

The second extension of the simple scoring scheme is to replace the constant gap penalty with what is called an affine function. The problem with the constant gap penalty is exemplified in the two optimal alignments in Figure 2-3. From a biological point of view, the alignment with two independent gaps (Figure 2-3b) is less likely than the one with only one gap of length 2 (Figure 2-3a). So we should find a gap penalty scheme that favors the alignment in Figure 2-3a against the one in Figure 2-3b. The

affine function, which is used in BLAST (Altschul *et al.*, 1990; Altschul *et al.*, 1997), is the simplest of the gap penalty schemes that will do the job. One particular advantage of the affine function is that it allows the alignment to be completed in time proportional to  $MN$ , where  $M$  and  $N$  are the length of the two sequences to be aligned.

The affine function for gap penalty is specified as

$$G(x) = -(a + bx) \quad (2.1)$$

where  $x$  is the length of the gap, and  $a$  and  $b$  are the gap open and gap extension penalties, respectively. The gap penalty increases linearly with the length of the gap. BLAST has its defaults with  $a = 5$  and  $b = 2$ , together with the match score ( $M$ ) = 1 and mismatch score ( $MM$ ) = -3.

Figure 2-7 illustrates the computation involved in aligning two sequences with  $M = 1$ ,  $MM = -3$  and the gap penalty specified with  $a = 5$  and  $b = 2$ , i.e., the default BLAST scoring scheme. Note that, while the first value in the matrix is still 0 (Figure 2-7) as before, the next value on the first row or first column is -7 which results in  $-(a + 1b) = -(5 + 1 \times 2) = -7$  (Note that a shift leftward or downward means inserting a gap either in the row or in the column sequence, respectively, and the first gap is associated with both the gap open and gap extension penalties. The values after -7 on the first row or on the first column are decreased by gap extension only, i.e., if a gap is already open, additional gaps will only suffer from gap extension penalties.

We again need to calculate three values in each remaining cells. In general, we calculate the DIAG, LEFT and UP values as specified below and fill the cell with the maximum of the three,

$$\begin{aligned} \text{DIAG} &= \text{UL} + \text{if}(\text{match}, M, MM) \\ \text{LEFT} &= L - \text{If}(\text{GapOpened already}, 0, a) - b \\ \text{UP} &= U - \text{if}(\text{GapOpened already}, 0, a) - b \end{aligned}$$

For the first cell,  $\text{DIAG} = 1$  because of the match of the two corresponding nucleotides, i.e., the A-A pair. The UP and LEFT values are both -9. So we have 1 in the cell with an upleft arrow (Figure 2-7). For the next cell, we have

$$\begin{aligned} \text{DIAG} &= -7 - 3 = -10 \\ \text{UP} &= -9 - 2 = -11 \\ \text{LEFT} &= 1 - 5 - 2 = -6 \end{aligned}$$

		A	C	G	T
	0	-7	-9	-11	-13
A	-7	1	-6	-8	-10
C	-9	-6	2	-5	-7
G	-11	-8	-5	3	-4
G	-13	-10	-7	-4	0
C	-15	-12	-9	-6	-7
T	-17	-14	-11	-8	-5

Figure 2-7. Pairwise alignment with  $M = 1$ ,  $MM = -3$ , and gap penalty defined by an affine function with  $a = 5$  and  $b = 2$ .

Note that LEFT value for this cell is penalized with both the gap open and gap extension penalty because the preceding cell (with value = 1) has an upleft arrow, i.e., no gap (Figure 2-7). If the preceding cell had a left-pointing arrow, which means that the gap has already been opened, only the gap extension penalty would be applied. The largest value of the three is LEFT (= -6), and the cell is therefore filled with -6 with a left-pointing arrow. This process continues until we get to the last cell, with a value of -5. This is the alignment score based on the scoring scheme with gap penalties specified with the affine function.

There are a few cells that need some explanation. The first is the cell with a value of -4 corresponding to the nucleotide G in the row sequence and the second nucleotide G in the column sequence. The cell has two arrows, one pointing up and one pointing upleft (Figure 2-7). This is because both the DIAG and UP values are equal to -4:

$$\text{DIAG} = -5 + 1 = -4$$

$$\text{UP} = 3 - (5 + 2) = -4$$

Had this cell been the last cell, i.e., if we were aligning the partial row sequence of “ACG” against the partial column sequence “ACGG”, we would get two alternative optimal alignments both with alignment score of -4:

Alignment 1: ACG-  
ACGG

Alignment 2: AC-G  
ACGG

The other cell with two arrows is in the last column, second row from the bottom. Had this cell been the last cell, i.e., if we were aligning the row sequence of “ACGT” against the partial column sequence “ACGGC”, we would get two alternative optimal alignments both with alignment score of -7 (i.e., three matches, one mismatch and one gap open and one gap extension penalty):

Alignment 1: ACGT-  
                  ACGGC

Alignment 2: ACG-T  
                  ACGGC

The trickiest cell is at the second last column and second last row, i.e., the one with an UP value of -6 and an up-pointing arrow. The DIAG and LEFT values for this cell are simple:

$$\text{DIAG} = -7 - 3 = -10$$

$$\text{LEFT} = -9 - 2 = -11$$

However, the UP value depends on which of the arrows in the cell above (i.e., the cell with a value of -4 and two arrows) we should take. If we take the up-pointing arrow (i.e., a gap has already been opened), then  $\text{UP} = -4 - 2 = -6$ . However, if we take the upleft arrow (i.e., no gap opened yet), then  $\text{UP} = -4 - (5 + 2) = -11$ . We should choose the maximum value, i.e., -6, and this constrains the previous cell (i.e., the cell with a value of -4 and two arrows) to have an up-pointing arrow. In other words, after we have put a -6 value into the cell, the cell above will no longer have two arrows but will only have an up-pointing arrow. It is for this reason that I have set the upleft arrow in that cell to grey but left the up-pointing arrow black.

Now following the backtrack arrows, we obtain the alignment in Figure 2-3a. The alignment has four matches and one gap of length 2. So the alignment score is  $4 \times 1 - (5 + 2 \times 2) = -5$ , which confirms that our scoring matrix (Figure 2-7) has been obtained correctly (note that the lower right value in the scoring matrix is the alignment score).

### 3. MULTIPLE SEQUENCE ALIGNMENT

#### 3.1 Profile alignment

Profile alignment aligns one sequence (designated T) against a set of already aligned sequences in the form of a profile (designated S), or align two profiles  $S_1$  and  $S_2$ . It is an essential technique for multiple sequence alignment. There are various approaches to profile alignment. The simplest is to get a consensus sequence from S (designated  $C_S$ ) and align T and  $C_S$  by using the pairwise alignment method we learned in previous sections. Whenever we insert a gap in  $C_S$ , we insert a corresponding gap in all sequences in S. However, we will learn a mathematically more acceptable approach in this section.

Suppose we want to align sequence T = “ACG” against S containing the following three aligned sequences:

```
AC-GT
AC-GT
GCCAT
```

The first step in profile alignment is to represent S with a site-specific frequency profile. The set of three aligned sequences have five symbols (A, C, G, T and the gap symbol “-”) and can be represented by the profile shown in the first five rows in Figure 2-8. The first column is a list of the five symbols, followed by five data columns corresponding to five aligned sites. The first data column represents the frequencies of symbols in the first aligned site, with the frequencies of A and G being  $2/3$  and  $1/3$ , respectively. The second data column represents the frequencies of the second aligned site with the frequency C being 1 and the frequencies of other symbols being 0, and so on. Thus, S can always be represented by a site-specific profile in the form of a  $N \times L$  matrix where N is the number of symbols and L is the sequence length. It is important to note that any phylogenetic information among sequences in S is lost in converting the set of aligned sequences to a profile.

The profile representation in the Clustal family of programs (Higgins and Sharp, 1988; Thompson *et al.*, 1994) uses all ambiguous codes (Tables 2-2 and 2-3). In addition, two synonymous pairs of ambiguous codes are used in Clustal, the X/N pair and the T/U pair. It would be computationally more efficient to pre-processing the sequences to use either X or N (or either T or U) but not both, especially when one uses a programming language that does not support pointers, e.g., Visual Basic or Java.



	A	2/3	0	0	1/3	0
	C	0	1	1/3	0	0
	G	1/3	0	0	2/3	0
	T	0	0	0	0	1
	-	0	0	2/3	0	0
	0	-3	-6	-9	-12	-15
A	-3	5/3	-4/3	-5/3	-14/3	-23/3
C	-6	-4/3	11/3	10/3	1/3	-8/3
G	-9	-13/3	2/3	3	5	2

Figure 2-8. Application of dynamic programming to profile alignment. T (“ACG”) is the column sequence, and the “row sequence” is a profile.

The second step is to perform a special version of the dynamic programming to generate the score matrix and backtrack matrix. With the length of T being 3 and the length of S being 5, there are only 15 cells to fill in. Because one needs to compute three values (DIAG, UP and LEFT) for each cell, the total number of values to compute is 45. Yet even for such a small problem, manual computation is quite difficult and error-prone.

The score matrix and the backtrack matrix (Figure 2-8) were obtained with a special scoring scheme. There are two kinds of matches, a match involving two identical nucleotides, or a match involving two gap symbols. The match score for the former and latter are designated  $M_{\text{Nuc}}$  and  $M_{\text{Gap}}$ , respectively, with corresponding values set to 2 and 1, respectively, in the example. There are also two kinds of mismatches, one involving a transitional difference and the other a transversional difference. They are designated as  $MM_s$  and  $MM_v$ , respectively, with corresponding values set to 1 and -1, respectively, in this example. In order not to make things too complicated, we use constant gap penalty with  $G = -3$ .

We now illustrate how the score matrix and backtrack matrix (Figure 2-8) are computed. For the first cell, the UP and LEFT values are simple:

$$\begin{aligned} \text{UP} &= -3 + G = -3 - 3 = -6 \\ \text{LEFT} &= -3 + G = -3 - 3 = -6 \end{aligned}$$

For the DIAG value, we should keep in mind that the nucleotide A in the column sequence has a probability of 2/3 of an A-A match and a probability of 1/3 of an A-G transition. This leads to

$$\text{DIAG} = 0 + 2/3 \times M_{\text{Nuc}} + 1/3 \times MM_s = 5/3$$

Because DIAG is the maximum of the three, it is used to fill the first cell, together with the associated upleft arrow (Figure 2-8). The second cell (to the right the first cell) is simple because the profile at the second site contains C only. So the computation is the same as in regular pairwise alignment:

$$\begin{aligned}\text{DIAG} &= -3 - 1 = -4 \\ \text{UP} &= -6 - 3 = -9 \\ \text{LEFT} &= 5/3 - 3 = -4/3\end{aligned}$$

Because LEFT is the largest of the three, the cell is filled with  $-4/3$  together with a left-pointing arrow.

The cell likely to cause some confusion is the third, i.e., the one with a value of  $-5/3$  and a left-pointing arrow. Note that an upleft arrow in that cell implies that A will pair with C with a probability of  $1/3$ , penalized by  $MM_v = -1$ , and pair with “-” with a probability of  $2/3$ , penalized by  $G = -3$ . Therefore,

$$\text{DIAG} = -6 - 2/3 \times 3 - 1/3 \times 1 = -25/3$$

The calculation of UP is simply  $\text{UP} = -9 - 3 = -12$ . A left-pointing arrow, however, implies a gap in the column sequence, so we have a gap with a probability of  $2/3$  of facing a gap in the row profile, with  $M_{\text{gap}} = 1$ , and a probability of  $1/3$  of facing a C, with  $G = -3$ . Therefore,

$$\text{LEFT} = -4/3 + 2/3 \times 1 - 1/3 \times 3 = -5/3.$$

Because LEFT is the largest of the three, the cell is filled with  $-5/3$  and a left-pointing arrow. The rest of the cells are relatively straightforward. The alignment can again be obtained by tracing the backtrack matrix (Figure 2-8):

```
AC-GT
AC-GT
GCCAT
AC-G-
```

The profile alignment outlined above represents an extension of the pairwise alignment, with the row sequence replaced by a profile. One can also replace the column sequence by a profile to align two profiles instead of

two sequences. This approach is used in Clustal for multiple sequence alignment.

One might argue that the profile alignment has a serious problem as follows. T may be phylogenetically more closely related to some sequences than others in S. However, the profile alignment approach does not take this into consideration. This critique is justified. Unfortunately, alternative approaches by combining both phylogenetic reconstruction and multiple sequence alignment (Hein, 1990, 1994; Sankoff *et al.*, 1973) are generally too computationally intensive to be practical. However, recent advances in Gibbs sampler has paved alternative ways for pairwise sequence alignment (Zhu *et al.*, 1998) and multiple sequence alignment conditional on a phylogenetic tree (Holmes and Bruno, 2001; Jensen and Hein, 2005).

### 3.2 Multiple alignment with a guide tree

The main difficulty in aligning multiple sequences by dynamic programming is the rapidly increased need for memory and computational power. While aligning three sequences by dynamic programming has been implemented (Huang, 1993), it is not practical to align more than three sequences. For this reason, only heuristic approaches that reduce the multiple alignment problem to pairwise and profile alignment problems have been widely used for multiple sequence alignment. The most well known representative of this approach is the Clustal family of programs (Higgins and Sharp, 1988; Thompson *et al.*, 1994).

Multiple alignment in Clustal consists of three steps. The first is to perform all pairwise alignments by dynamic programming. With N sequences, there are  $N(N-1)/2$  pairwise alignments, leading to a triangular matrix of alignment scores. The second step is to construct a guide tree by using the alignment score matrix as a sequence similarity matrix in conjunction with a clustering algorithm. Alternatively, one can convert the similarity matrix into a distance matrix and then use either UPGMA or neighbor-joining method (Saitou and Nei, 1987) to build guide tree such as the one shown in Figure 2-9. Clustal uses this latter approach. The third and final step is to traverse the node to align sequences by pairwise alignment and profile alignment (the pairwise alignments in the first step, typically an approximate one, are not reused here).

The multiple alignment starts from the most similar sequences. So we move to internal node 11 and align Seq2 and Seq6. We then move to internal node 10 and align Seq5 against a sequence profile representing aligned Seq2 and Seq6 using the method outlined in Figure 2-8. A profile is then created to represent the three aligned sequences (Seq2, Seq6 and Seq5). Moving to internal node 9, we found one child node (internal node 9) has two child

nodes with two unaligned sequences (Seq3 and Seq4) which are then first aligned by using the dynamic programming method. A profile is then created to represent the aligned Seq3 and Seq4. This profile is then aligned against the profile presenting the aligned Seq2, Seq6 and Seq5. The process continues until all sequences are aligned.

It is easy to see why we should start with the most similar sequences because any alignment error will be propagated in subsequent alignment. Obviously, a wrong guide tree will bias the subsequent alignment which in turn will bias subsequent phylogenetic reconstruction based on the alignment. Unfortunately, a guide tree built from alignment scores is typically a very poor tree. For this reason, it is better to input a well established tree, whenever available, as a guide tree for multiple sequence alignment.

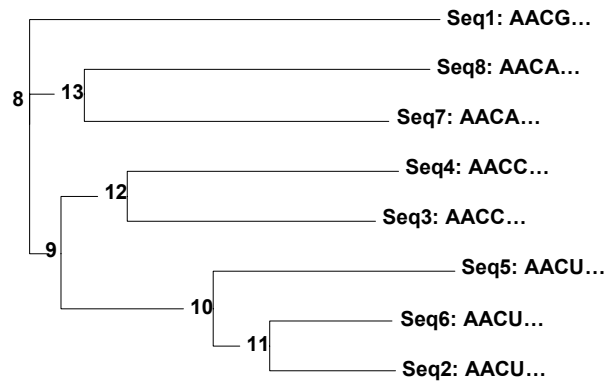


Figure 2-9. An example of a guide tree for multiple sequence alignment of eight sequences, called leaves. The internal nodes are numbered from 8 to 13 (with terminal nodes, or leaves, numbered from 0 to 7).

An alternative method of multiple sequence alignment is to use a star tree instead of fully resolved bifurcating tree as a guide tree. One starts with pairwise alignment to obtain a matrix of alignment scores in the same way as in ClustalW. This matrix is then converted to a distance matrix. We first align two sequences with the smallest distance to build the first sequence profile. The next sequence with the average distance closest to the sequence profile is the aligned to the sequence profile using the algorithm illustrated in Figure 2-8. A new sequence profile is obtained from these three sequences and the next sequence with the smallest average distance to those in the profile is aligned against the profile using the algorithm illustrated in Figure 2-8. This process continues until all sequences have been aligned.

This method is simpler than the method in ClustalW because one does not need to align two sequence profiles, i.e., in every step, the alignment is

done between a sequence and a profile. The “Quick multiple alignment” function in my program DAMBE (Xia, 2001; Xia and Xie, 2001b) represents a crude implementation of this method of multiple alignment.

#### 4. SEQUENCE ALIGNMENT WITH SECONDARY STRUCTURE

Figure 2-10 shows two sequences, S and T, being a fragment of a fictitious rRNA gene from two related species. The fragment forms a stem-loop structure. For simplicity, suppose that T was derived from S through the intermediate T' in two steps. First, the C at position 11 was deleted. Second, a substitution from A to G at position 5 leads to a correlated substitution from T(U) to C to maintain the stem of length 5. The resulting sequence is T.

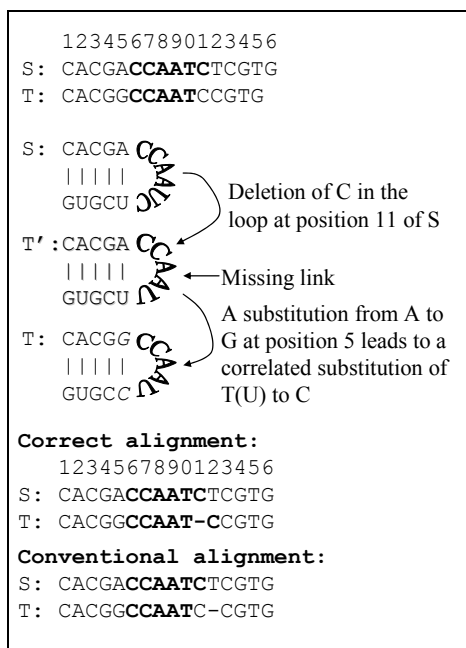


Figure 2-10. Illustration that the correct alignment may differ from the optimal alignment. Note that T becomes U in the secondary structure.

If we constrain the alignment with the secondary structure information, i.e., the first five and the last five nucleotides of S are respectively homologous to the first five and the last five nucleotides of T, then the resulting alignment (designated as the correct alignment in Figure 2-10) correctly identifies the gap at position 11. However, any currently used

alignment program based on linear sequence information, with any sensible scoring scheme, would recover the ‘conventional alignment’ (Figure 2-10) that identified the gap at position 12. The two C’s at position 11 in the ‘conventional alignment’ are nicely aligned but are not homologous given the evolutionary steps given above in generating T. It is easy to see that the conventional alignment will have a greater alignment score than the correct alignment and consequently is more “optimal”. Thus, optimal alignment (the alignment with the largest alignment score) may not necessarily be the correct alignment.

Aligning rRNA genes with the constraint of secondary structure has now been frequently used in practical research in molecular evolution and phylogenetics (Hickson *et al.*, 2000; Kjer, 1995; Notredame *et al.*, 1997; Xia, 2000; Xia *et al.*, 2003a). However, one cannot always assume that rRNA secondary structure is stable over time. It is now well established that variation in rRNA secondary structure is strongly affected by the optimal growth temperature in prokaryotes (Galtier and Lobry, 1997; Hurst and Merchant, 2001; Nakashima *et al.*, 2003; Wang and Hickey, 2002; Wang *et al.*, 2006).

## 5. ALIGN NUCLEOTIDE SEQUENCES AGAINST AMINO ACID SEQUENCES

During the evolution of protein-coding genes, an entire codon or multiple codons may be deleted or inserted, but it is much rarer to see an insertion or deletion (often abbreviated as indel) of one or two nucleotides because such indel events lead to frameshifting mutations that almost always disrupt the original protein function and are strongly selected against. However, alignment of protein-coding nucleotide sequences often produce indels of one or two bases as alignment artifacts. The correctly aligned sequences should have complete codons, not one or two nucleotides, inserted or deleted.

One way to avoid the above alignment problem is to align the protein-coding nucleotide sequences against amino acid sequences, which was implemented in software DAMBE (Xia, 2001; Xia and Xie, 2001b). The approach obviously requires amino acid sequences which can be obtained in two ways. First, if you have nucleotide sequences of good quality, then you can translate the sequences into amino acid sequences, which can be done automatically in DAMBE which implements all known genetic codes for translating protein-coding sequences from diverse organisms. Second, if you are working on nucleotide sequences deposited in GenBank, then typically you will find the corresponding translated amino acid sequences.

The alignment of protein-coding nucleotide sequences is typically done in three steps. First, the nucleotide sequences are translated into amino acid sequences. These amino acid sequences are then aligned, and the nucleotide sequences are then aligned against the aligned amino acid sequences.

Here is a simple illustration. Suppose we are to align the following two protein-coding sequences designated S1 and S2, respectively:

```
S1 ATG CCG GGA TAA
S2 ATG CCC GGG ATT TAA
```

Step 1: Translate the sequences into amino acid sequences (one-letter notation) to get:

```
S1 MPG*
S2 MPGI*
```

Step 2: Align the amino acid sequences:

```
S1 MPG-*
S2 MPGI*
```

This alignment implies the deletion of an amino acid (and its associated codon) just before the termination codon.

Step 3. Align the protein-coding nucleotide sequences against aligned amino acid sequences. This is done by essentially mapping the codon sequences to the aligned amino acid sequences. Keep in mind that a gap in the aligned amino acid sequences correspond to a triplet gap in nucleotide sequences:

```
S1 ATG CCG GGA --- TAA
S2 ATG CCC GGG ATT TAA
   *** **  *** *   ***
```

This alignment, designated as Alignment 1, has 10 matches, 2 mismatches, and 1 gap of length 3. Recall that the main objective of sequence alignment is to identify homologous sites and it is important to note that different alignments may lead to different interpretations of sequence homology. With the alignment above, sites 6 of the two sequences (G in S1 and C in S2) are interpreted as a homologous site, so is site 9 (A in S1 and G in S2). These interpretations are not established facts. They are only inferences of what might have happened.

Depending on the scoring scheme, a nucleotide-based sequence alignment, i.e., without using aligned amino acid sequences as a mapping reference, may well generate the following alignment designated Alignment 2, with 12 matches, 0 mismatch and two gaps of lengths 1 and 2, respectively:

```
S1 ATG CC- GGG A-- TAA
S2 ATG CCC GGG ATT TAA
   *** **  *** *   ***
```

Note three different interpretations of the homologous sites between Alignment 1 and Alignment 2. First, the nucleotide G at site 6 of S1 is now interpreted to be homologous to the nucleotide G at site 7 of S2. Second, the nucleotide A at site 9 of S1 is now interpreted as homologous to the nucleotide A at site 10 of S2. Which of the two alignments makes more sense? If Alignment 1 is correct but we used a nucleotide-based alignment method and end up with Alignment 2, then the estimation of the genetic distance between the two sequences will be biased. The genetic distance measures the evolutionary dissimilarity between two sequences, often estimated by ignoring the indel sites. It is often used as an index of sequence divergence time in molecular phylogenetics, when calibrated by fossils with known divergence time. In this particular case, if we perform site-wise deletion of indels, then S1 and S2 would appear more similar to each other in Alignment 2 than in Alignment 1. Biased estimation of the genetic distance often results in failure in molecular phylogenetic reconstruction.

Given that a protein-coding gene is unlikely to remain functional after two consecutive indel mutations as in Alignment 2, we may argue that Alignment 1 based on the alignment of amino acid sequences is better than Alignment 2. However, there are also cases where a nucleotide-based alignment is better. Now consider the following two protein-coding sequences:

```
      3   6   9  12  15
S1 ATG CCC GTA TAA
S2 ATG CCC GTG TTA TAA
```

Of the following three alignments, designated as Alignment 1, Alignment 2 and Alignment 3, all involving one indel of length 3, which one makes more sense to you?

```
Alignment 1:
S1 ATG CCC GTA --- TAA
S2 ATG CCC GTG TTA TAA
```



```

***  ***  **          ***

Alignment 2:
S1 ATG CCC GT- --A TAA
S2 ATG CCC GTG TTA TAA
   ***  ***  **      *  ***

```

```

Alignment 3
S1 ATG CCC G- --TA TAA
S2 ATG CCC GTG TTA TAA
   ***  ***  *      **  ***

```

Alignment 1 is the outcome of aligning nucleotide sequences against aligned amino acid sequences, and the other two alignments are from nucleotide-based alignments. The three alignments represent three alternative hypotheses, all involving a gap of length 3, but differ in the position of the gap. Alignment 1 has only 11 matches, whereas the other two alignments each have 12 matches. In this case, the two hypotheses represented by alignments 2 and 3 is more likely true than alignment 1. In short, aligning protein-coding sequences against aligned amino acid sequences is not necessarily better than aligning the nucleotide sequences directly unless the latter produces frameshifting deletions or insertions.

## 6. POSTSCRIPT

A student has once told me that it is humiliating to have human gene sequences aligned against those of chimpanzees, monkeys or even snakes and turtles. I suspect that people of the past would also find it humiliating to have our earth displaced from the center of the universe and ranked among the other planets. Indeed it would have been much nicer, at least esthetically more charming and spiritually more enlightening, to have human genes all quite different from genes of all other living creatures. It would also have been nicer to have our earth centered in the universe with all stars and galaxies orbiting around us. That would instill into our mind a certain confidence that a supernatural custodian is looking after our wellbeing. We would have been more coherent when preaching to our children.

But nature, as beautiful as she is, does not always seem to be our maid working according to our dictation or wistful thinking. We are part of nature's creation and have been shaped by the same two sculptors of biodiversity called mutation and selection as all the other creatures. From this perspective we can better appreciate the truth that not only all humans

are created equal, but also are all other creatures populating the earth. Sequence alignment is a powerful tool to help us position ourselves properly in the nature of things.



<http://www.springer.com/978-0-387-71336-6>

Bioinformatics and the Cell  
Modern Computational Approaches in Genomics,  
Proteomics and Transcriptomics

Xia, X.

2007, XVI, 350 p., Hardcover

ISBN: 978-0-387-71336-6