CLUSTALG Multiple Sequence Alignment Program: A Hand Worked Example

Julie Thompson-Maaloum
Institute de Géntique et de Biologie Moleculaire et Cellulaire, Université de Strasbourg

Clarke Wilson
Canada Mortgage and Housing Corporation
Ottawa

0.0 Introduction

ClustalG is a 'general' version of the ClustalX multiple sequence alignment program used for analysis of protein and nucleotide molecules. It provides an integrated environment for reading sequence files, performing pairwise, multiple sequence and profile alignments and analysing the results. ClustalG eliminates the specifically biological features of ClustalX and recognizes an expanded alphabet for representing events or activities making it suitable for research in many branches of the social and natural sciences.

ClustalG is available at:

`www.stmarys.ca/partners/iatur/clustalG/`

or from either of:

`cwilson@cmhc-schl.gc.ca`
`andrew.harvey@stmarys.ca`

Input sequences consist of alpahbetic characters that represent events or items ordered in time or one-dimensional space. The algorithms incorporated in ClustalG calculate scores of the similarities among the input character sequences. The raw and aligned sequences are displayed in a window on the screen. A versatile coloring scheme has been incorporated allowing you to highlight features conserved across sequences in the alignment. The pull-down menus at the top of the window allow you to select all the options required for traditional multiple sequence and profile alignment. Alignment quality analysis can be performed and low-scoring segments or exceptional activities can be highlighted.

In order to align just two sequences, it is standard practice to use dynamic programming. This guarantees a mathematically optimal alignment, given a table of scores for matches and mismatches between all sequence elements and penalties for insertions or deletions of different lengths. Attempts at generalising dynamic programming to multiple alignments are limited to small numbers of short sequences. For much more than eight or so proteins of average length, the problem is uncomputable given current computer power. Therefore, all of the methods capable of handling larger problems in practical time scales, make use of heuristics. One of the most widely used approaches to the multiple sequence alignment problem is the "progressive" approach of Feng and Doolittle. One can build up a multiple alignment progressively by a series of pairwise alignments, following the branching order in a guide tree (7). One first aligns the most closely related sequences, gradually adding in the more distant ones.

The ClustalG multiple alignment method can be divided into four phases :

1. pairwise alignment - all pairs of sequences are aligned separately in order to calculate a distance matrix containing the divergence of each pair of sequences;
2. guide tree - a tree is calculated from the distance matrix, describing the approximate groupings of the sequences by similarity;
3. multiple alignment - the sequences are progressively aligned according to the branching order in the guide tree;
4. alignment analysis - a final neighbor-joining tree may be constructed based on the multiple alignment and quality statistics may be calculated.

Complete documentation of the details of the use of ClustalG, including file formats, parameter settings, and output files is provided in the Help function in ClustalG. The Help file can be read and printed with a word processor.

0.1 Sequence Input

The raw unaligned sequences or aligned sequences (profiles) are input using the FILE menu. All the sequences to be aligned should be in a single file, one after another. Seven different sequence input formats are recognised automatically and read by the program: Pearson / Fasta, CLUSTAL, NBRF/PIR, EMBL/Swiss Prot, GCG/MSF, GCG9 RSF and GDE flat file. Of these, the Pearson format is easiest to create and will be used throughout this example. Alignment output may be requested in standard CLUSTAL format (self-explanatory blocked alignments) or in formats compatible with the GDE, PHYLIP or GCG packages. The program offers the user the ability to calculate Neighbour-Joining trees from existing alignments. The trees may be output in the "New Hampshire", nested parenthesis format that is compatible with a number of tree-drawing packages such as DrawTree, TreeView.

Example sequence input file :

>One
ABCDEF
>Two
ADEGH
>Three
BBDBFG
>Four
DEFFFG
>Five
CCDDE
>Six
AABDE

0.2 User defined parameters

Substitution Matrix

The substitution table defines the similarities between the characters used in the sequences. In social science applications, characters normally represent events or activities. By default, an identity matrix is used, in which a match between two identical characters scores 10 and all mismatches score 0. However, ClustalG can also read character similarity scores in one of two different formats: as a similarity matrix or as a parsing file. The matrix format is suitable for small, single character alphabets. If the classification system for sequence elements is large or if multi-letter words are used, the parsing file format is convenient. The ClustalG help file contains descriptions of the matrix and parsing file formats.

Example substitution matrix :

	A	**B**	**C**	**D**	**E**	**F**	**G**	**H**
A	10	0	7	0	0	-1	-1	0
B	0	10	0	0	0	0	0	0
C	7	0	10	0	0	0	0	0
D	0	0	0	10	0	5	0	0
E	0	0	0	0	10	0	0	0
F	-1	0	0	5	0	10	0	0
G	-1	0	0	0	0	0	10	0
H	0	0	0	0	0	0	0	10

In Parsing Format :

```
$matrix
$         10
[A:C]      7
[D:F]      5
[A:F]     -1
[A:G]     -1
?          0
```

ClustalG calculates the mean score for a mismatch, to be used in selecting appropriate gap opening and extension penalties (see below). The mean mismatch score corresponds to the mean off-diagonal score in the matrix representation above. For this example, the mean mismatch score is 0.35.

Penalties for opening and extending a gap: no scaling

Initially, two gap penalties are defined by the user: a gap opening penalty (GOP) which gives the cost of opening a new gap of any length and a gap extension penalty (GEP) which gives the cost of every item in a gap. The user may use these as specified by selecting the no-scaling option in the pairwise parameter screen.

Penalties for opening and extending a gap: automatic scaling

As an alternative to no scaling, ClustalG can automatically attempt to choose appropriate gap penalties for each sequence alignment, depending on the score matrix and the lengths of the sequences.

In order to select gap penalties compatible with the chosen score matrix, the mean score for two mismatched residues (ie. off-diagonal values in the matrix) is used as a scaling factor for the GOP. Also, the alignment scores for both true and false sequence alignments grow with the length of the sequences. The natural logarithm of the length of the shorter sequence is used to increase the GOP with sequence length.

In the example, User GOP = 1.0, User GEP 0.1. With these two modifications, the final gap penalties used by the program are :

(if mean mismatch score is negative)

GOP = 2*(mean absolute mismatch score)*(User_GOP+logn(MIN(L_i,L_j))

(otherwise)

GOP = 2*(User_GOP+logn(MIN(L_i,L_j))
GEP = User_GEP = 0.1

Where GOP, GEP are the gap opening and extension penalties set by the user
L_i,L_j are the lengths of the two sequences to be aligned

In the example, the matrix mismatch score is 0.35,
therefore, GOP = 2* (User_GOP+logn(MIN(L_i,L_j))) = 2*(1.0+1.6) = 5.2

1.0 Phase one: Pairwise Alignment

ClustalG offers three options for the calculation of the maximum similarity score between two sequences or profiles: a fast approximate method based on exact comparisons of short window segments and two dynamic programming options, global alignment (Needleman Wunsch) and local alignment (Smith Waterman). This example shows the local option. The algorithm can be visualised with a path graph (Table 1.1). The goal is to maximise the similarity score for the alignment that ends at each vertex. The similarity score between a pair of sequences is defined as the sum of the similarity scores for all aligned pairs of characters, minus a gap penalty for each gap introduced in either of the sequences. The recursive dynamic programming algorithm can be summarised by the following formula:

$$H_{i,j} = MAX \left\{ \begin{array}{c} 0 \\ H_{i-1,j-1} + S_{i,j} \\ MAX\{H_{i-k,j} - (gop + gep * k)\} \\ MAX\{H_{i,j-l} - (gop + gep * l)\} \end{array} \right\} \qquad \text{Smith-Waterman algorithm}$$

where $H_{i,j}$ is the score for the alignment that ends at vertex i,j
$S_{i,j}$ is the score for aligning the two residues at vertex i,j

Table 1.1 Alignment path for sequences one and two.

	A	B	C	D	E	F
A	10	0	7	0	0	-1
D	0	⬂ 10	⇨ 4.8	➘ 17	⇨ 11.8	⇨ 11.7
E	0	⇩ 4.8	⬂ 10	⇩ 11.8	➘ 27	⇨ 21.8
G	-1	⇩ 4.7	⇩ 4.8	⇩ 11.7	⇩ 21.8	➘ 27
H	0	⇩ 4.6	⇩ 4.7	⇩ 11.6	⇩ 21.7	⇩ 21.8

The maximal score at any vertex is 27, corresponding to two different alignments. The first alignment is selected, as this leads to the highest percent identity score.

```
One   CDE            or        One  CDEF
Two   ADE                      Two  ADEG
```

The Smith-Waterman dynamic programming alignment algorithm is repeated for all pairs of sequences. The results are summarised in the Table 1.2.

SIM=similarity score
ALEN = Alignment length
NID= Number of identical residues
SLEN=Shortest sequence length
%ID=percentage of identical residues

Table 1.2: Pairwise alignment scores

Sequences	Alignment	Similarity calculation	SIM	ALEN	NID	SLEN	%ID
one two	CDE ADE	7+10+10	27	3	2	5	40
one three	BCDEF BBDBF	10+0+10+0+10	30	5	3	6	50
one four	DEF DEF	10+10+10	30	3	3	6	50
one five	ABCDE CCDDE	7+0+0+10+10	27	5	2	5	40
one six	ABCDE AB-DE	10+10+10+10 -(5.2+0.1)	34	5	4	5	80
two three	DE-G DBFG	10+0+10 -(5.2+0.1)	14	4	2	5	40
two four	DE---G DEFFFG	10+10+10 -(5.2+3*0.1)	24	4	3	5	60
two five	AD-E CDDE	7+10+10 -(5.2+0.1)	21	4	2	5	40
two six	A-DE ABDE	10+10+10 -(5.2+0.1)	24	4	3	5	60
Three four	DBFG FFFG	5+0+10+10	25	4	2	6	33
three five	D D	10	10	1	1	5	20
three six	BD BD	10+10	20	2	2	5	40
four five	DE DE	10+10	20	2	2	5	40
four six	DE DE	10+10	20	2	2	5	40
five six	CCDDE AABDE	7+7+0+10+10	34	5	2	5	40

Pairwise percent identity scores are calculated as the number of identical residues in the best alignment divided by the length of the shortest sequence. These scores are then converted to distances by dividing by 100 and subtracting from 1.0 to give the sequence distance matrix.

Table 1.3: Pairwise percent identities:

	one	**two**	**three**	**four**	**five**	**six**
one	100	40	50	50	40	80
two	40	100	40	60	40	60
three	50	40	100	33	20	40
four	50	60	33	100	40	40
five	40	40	20	40	100	40
six	80	60	40	40	40	100

Table 1.4: Sequence Distance Matrix:

	one	**two**	**three**	**four**	**five**	**six**
one	0	0.60	0.50	0.50	0.60	0.20
two	0.60	0	0.60	0.40	0.60	0.40
three	0.50	0.60	0	0.67	0.80	0.60
four	0.50	0.40	0.67	0	0.60	0.60
five	0.60	0.60	0.80	0.60	0	0.60
six	0.20	0.40	0.60	0.60	0.60	0

2.0 Phase Two: Create guide tree

The trees used to guide the final multiple alignment process are calculated from the distance matrix of phase 1 using the Neighbour-Joining method (Saitou and Nei, 1987). This produces unrooted trees with branch lengths proportional to the estimated sequence uniqueness along each branch. Neighbours in the sense of the algorithm are sequence pairs or groups joined through an interior node in a bifurcating tree. Nearest neighbours are not necessarily the sequence pair with the smallest distance. They are the pair that minimizes the sum of branch lengths in the tree.

The algorithm operates by assuming that the initial data structure is a star and no subgroups of individual sequences exist. Distances between any two sequences in the star structure are D_{ij}. The algorithm finds successive pairs of neighbours, which consist initially of sequence pairs and later of sequence groups, until a bifurcating tree has been defined. Given N sequences, the sum of branch lengths is given by equation 3.1.

$$S_{1,2} = \tfrac{1}{2} D_{1,2} + \frac{1}{2(N-2)} \sum_{k=3}^{N} (D_{1,k} + D_{2,k}) + 1/(N-2) \sum_{3<=i<j} D_{i,j} \qquad 3.1$$

The element pair, called 1 and 2, that has the minimum value is selected and branch lengths, L_{1x} and L_{2x} to their node, X, are calculated as in equation 3.2.

$$L_{1x} = (D_{12} + D_{1z} – D_{2z}) / 2$$

$$L_{2x} = (D_{12} + D_{2z} – D_{1z}) / 2 \qquad 3.2$$

In D_{1z} and D_{2z} in equation 3.2, Z represents the set of all sequences excluding 1 and 2. D_{1z} and D_{2z} are defined as in equation 3.3.

$$D_{1z} = (\Sigma_{=3} \; D_{1i}) / N - 2$$

$$D_{2z} = (\Sigma_{i=3} D_{2i}) / N - 2 \qquad 3.3$$

The distance matrix is recalculated after each cycle simply as the mean of the distances from 1 and 2 to each remaining sequence. Other distances from sequences j to k are unchanged from their initial values.

$$D_{1\text{-}2,j} = (D_{1,j} + D_{2,j}) / 2 \qquad 3.4$$

The values of the matrix S_{ij} are defined by the three terms of equation 3.1. The value of S_{16} is the smallest of all S_{ij} and the branch lengths to node A are determined by:

$$\begin{aligned} S_{16} = \; & (.2) / 2 \\ & + (.6+.5+.5+.6+.4+.6+..6+.6) / 8 \\ & + (.6+.4+..6+.67+.6+.6) / 4 \\ = \; & 1.57 \end{aligned}$$

The other 14 branch length sums are larger than 1.57.

2.1 Cycle one:

The first step consists of creating a 'node' joining the two most closely related sequences, One and Six. The fact that they minimize the sum of branch lengths and have the minimum distance ($D_{16} = 0.2$) is coincidental. From equations 3.2 and 3.3 we have

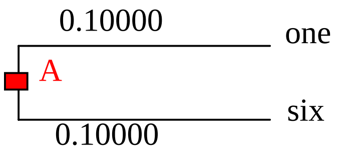
$D_{1z} = (.6+.5+.5+.6) / 4 = .55$

$D_{6z} = (.4+.6+.6+.6) / 4 = .55$

$L_{1A} = (.2+.55-.55) / 2 = .1$

$L_{6A} = (.2+.55-.55) / 2 = .1$

Node A : Sequence: one (L_{1A} = 0.10000) joins Sequence: six (L_{2A} = 0.10000)



The distances from the new node to the remaining sequences are recalculated, and the Sij values are recalculated. The distance matrix for cycle 2 is given in Table 2.1.

Table 2.1: N-1 Distance table

	1-6	**two**	**three**	**four**
two	.5	0		
three	.55	.6	0	
four	.55	.4	.67	0
five	.6	.6	.8	.6

2.2 Cycle two:

While the smallest value in the distance matrix is $D_{2,4}$, $S_{1\text{-}6,3}$ minimizes the sum of tree branch lengths and sequence three is added to the tree at node B. We have:

$$S_{1\text{-}6,3} = (.55)/2 + (.5+.55+.6+.6+.67+.8)/6 + (.4+.6+.6)/3 = 1.43 \text{ and}$$

$$S_{24} = (.4)/2 + (.5+.6+.6+.55+.67+.6)/6 + (.55+.6+.8)/3 = 1.44$$

which is more than $S_{1\text{-}6,3}$. Equations 3.2 and 3.3 give:

$D_{1\text{-}6z} = (.5+.55+.6) / 3 = .55$

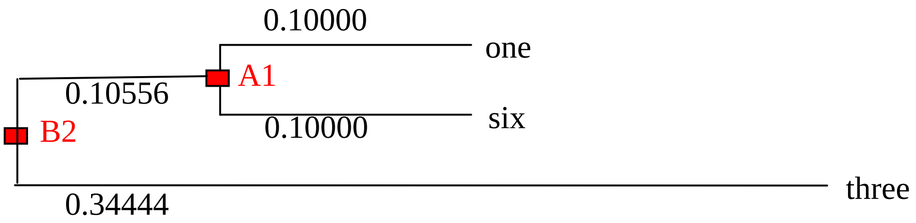
$D_{3z} = (.6+.67+.8) / 3 = .69$

$L_{1\text{-}6B} = (.55+.55-.69) / 2 = 0.205$

$L_{3B} = (.55+.69-.55) / 2 = 0.345$ and

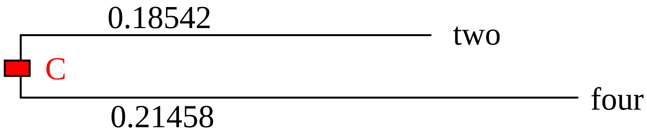
$L_{AB} = L_{1\text{-}6B} - L_{1A} = 0.105$

Node B : Node: A (0.10556) joins Sequence: three (0.34444)



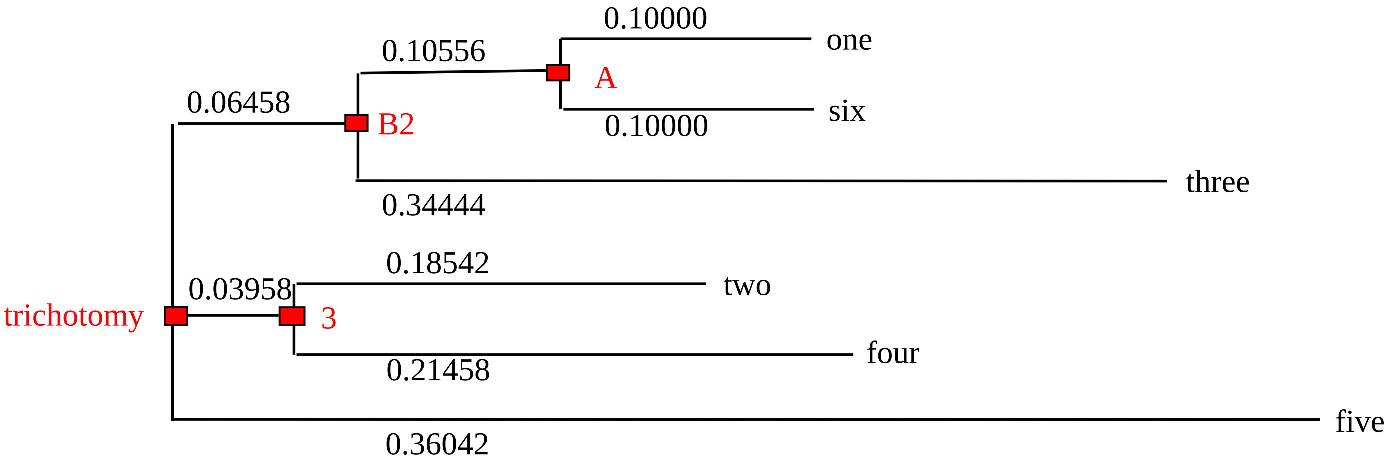
Cycles 2.n are repeated until all sequences are included in the tree.

Node C : Sequence: two (0.18542) joins Sequence: four (0.21458)



Node D (Last cycle, trichotomy) :

Node: 2 (0.06458) joins Node: 3 (0.03958) joins Sequence: five (0.36042)



The tree is written to a file in the New Hampshire file format, compatible with a number of tree drawing packages (Phylip, TreeTool, Njplot, Treeview, etc.). Branch lengths are printed after the sequence labels.

New Hampshire File Format:

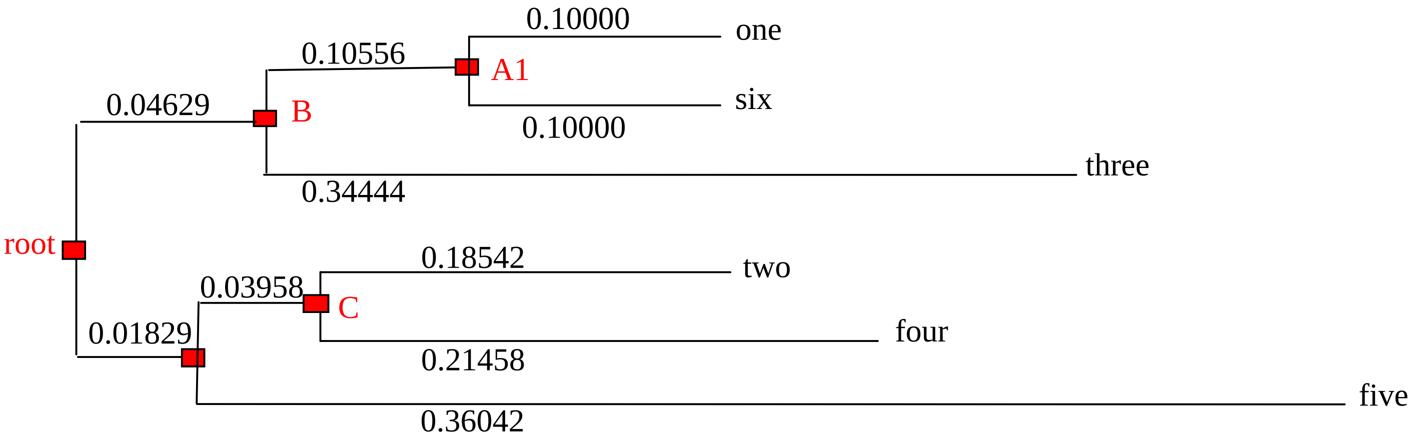
```
(((one:0.10000,
six:0.10000)
:0.10556,
three:0.34444)
:0.06458,
(two:0.18542,
four:0.21458)
:0.03958,
five:0.36042);
```

2.3 Optional weighting of sequences

Sequence data may be derived from surveys of representative samples of a population of interest or from surveys without formal sampling designs. In the former case, no additional weighting is appropriate since the sampling design was adopted to allow some desired mix of respondents to be selected randomly.

In the latter case, a researcher may want to correct for unintended oversampling or misrepresentation in the data set. In this situation, groups of closely related sequences receive lowered weights because they contain much duplicated information. Highly divergent sequences without any close relatives receive high weights. These weights are used in the multiple alignment stage as simple multiplication factors for scoring positions from different sequences or prealigned groups of sequences. The guide tree is first rooted; the root is placed such that the mean branch length on the left side of the root is equal to the mean branch length on the right. The weights are calculated from the branch lengths of the rooted guide tree and normalised such that the sum of the sequence weights is one.

Rooted guide tree:



The sequence weights are dependent upon the distance from the root of the tree but sequences which have a common branch with other sequences share the weight derived from the shared branch. For example, the sequence “four” gets a weight consisting of the length of the branch leading to it that is not shared with any other sequences (0.215) plus half the length of the branch shared with sequence “two” (0.040), plus one third of the length of the branch to the root (0.018). This sums to a total of 0.240.

		Raw weight	Normalised weight
One:	0.100/1 + 0.106/2 + 0.046/3	= 0.168	0.108
Six:	0.100/1 + 0.106/2 + 0.046/3	= 0.168	0.108
Three:	0.344/1 + 0.046/3	= 0.359	0.236
Two:	0.185/1 + 0.040/2 + 0.018/3	= 0.211	0.141
Four:	0.214/1 + 0.040/2 + 0.018/3	= 0.240	0.162
Five:	0.360/1 + 0.018/3	= 0.366	0.243

Note that the sequence closest to the root is sequence two (0.018+0.040+0.185=0.243) and this is where the mulitple alignment phase will begin.

3.0 Phase 3: Build multiple alignment

The basic procedure at this stage is to use a series of pairwise alignments to align larger and larger groups of sequences, following the branching order in the guide tree. We proceed from the tips of the rooted tree towards the root. For each node in the rooted tree, the sequences of the left branch of the node are aligned with those on the right branch.

Each step uses the Needleman-Wunsch global pairwise alignment algorithm to align two sequences or two previous alignments, or to add a single sequence to an alignment. As this process progresses, we eventually place whole alignments on the borders of a comparison table rather than single sequences (as was shown in Table 1.1). To make the required comparisons a profile score matrix ($PS_{r,c}$) that is analogous to the original substitution matrix is required. However, the new matrix defines similarities between positions in two alignments, each position consisting of at least two characters and possibly having several hundred characters and gaps depending on the step number and the total number of sequences in the file. We also redefine penalties for opening and extending gaps. Gaps that are present in older alignments remain fixed.

In order to align alignments (groups of sequences), each alignment is converted into a profile. The profile is a matrix, where each row, r, represents a position in the multiple alignment. The row labels are transposed columns of characters and gaps of the aligned sequences. Each column, c, represents one character from the alphabet. The column labels will be the alphabet (or the part of the alphabet used in the alignment). The profile also contains the position-specific gap opening and extension penalties. For a group of N aligned sequences (r_i, i=1 to N) of length L (c_j, j=1 to L), let a_{ij} be the activity in sequence i at position j.

The first profile uses the frequencies of the events observed in the group of sequences and the original substitution matrix ($S_{i,j}$) :

$$\text{Profile\_1}(\mathrm{r},\mathrm{c}) = \sum_{\mathrm{d}=1}^{\mathrm{D}} \mathrm{w}_{\mathrm{d}} S(event_{\mathrm{d}}, event_{\mathrm{c}}) * (n_r / N)$$

where D is the number of characters in the alphabet, $event_c$ is the event represented by the column in the profile. N is the number of sequences and n_r is the number of non-gap characters at the position. The character weight, w_d, is given by:

$$w_d = \frac{\sum_{i=1}^{N} w_i \delta_d}{\sum_{i=1}^{N} w_i} \qquad \delta_d = \begin{cases} 1 & if \quad a_{ij} = event_d \\ 0 & if \quad a_{ij} \neq event_d \end{cases}$$

where w_i is the weight of sequence *i*, as determined in section 2.3. In the last term, scores for each position are multiplied by the proportion of characters (as opposed to gaps) in the alignment at this position. Gap characters at the ends of the sequences are not considered as gaps by default.

The second profile uses a different calculation from the first, being based only on the observed residue frequencies in the group of sequences:

$$\mathrm{Profile\_2(r,c)} = \sum_{\mathrm{d=1}}^{\mathrm{D}} \mathrm{w_d}$$

The asymmetry in the treatment of Profile1 and Profile2 originates from experience in alignment applications in biology and does not have a theoretical basis.

The profile score ($PS_{i,j}$) for aligning a position from one alignment and one from another is then the product of the two profile rows:

$$\mathrm{PS_{i,j}} = \sum_{d=1}^{D} \mathrm{Profile\_}\ 1(r_i, d) * \mathrm{Profile\_}\ 2(r_j, d)$$

The scores for opening and extending gaps in the profiles are calculated as follows :

Score for opening a gap after position i in Profile_1 and position j in Profile_2:
GOP = GOP1(i) + GOP2(j+1)
Score for opening a gap after position j in Profile_2 and position i in Profile_1:
GOP = GOP1(i+1) + GOP2(j)

Score for extending a gap in profile 1, opposite position j in profile 2:
GEP = GEP2(j)
Score for extending a gap in profile 2, opposite position j in profile 1:
GEP = GEP1(j)

The alignment parameters for the first profile alignment, which is in effect a simple pairwise alignment, are:

Substitution matrix (as for pairwise alignment)

	A	**B**	**C**	**D**	**E**	**F**	**G**	**H**
A	10	0	7	0	0	-1	-1	0
B	0	10	0	0	0	0	0	0
C	7	0	10	0	0	0	0	0
D	0	0	0	10	0	5	0	0
E	0	0	0	0	10	0	0	0
F	-1	0	0	5	0	10	0	0
G	-1	0	0	0	0	0	10	0
H	0	0	0	0	0	0	0	10

Calculation of gap penalties

The default values for User_GOP = 1.0 and User_GEP = 0.05

If MIN(L_i,L_j) < 100, minlen=0;
Otherwise, minlen= logn(MIN(L_i,L_j)

If mean mismatch score is negative :
GOP = mean mismatch score*(User_GOP+minlen)
Otherwise :
GOP = 0.5*(User_GOP+minlen)

In the example, GOP = 0.5* (1.0+0.0) = 0.5 for L=5
GEP = User_GEP = 0.05

End gaps are not penalised.

3.1 Step 1 - align sequence two with sequence 4

Each sequence is converted to a profile and the alignment is constructed using the global Needleman-Wunsch algorithm:

$$H_{i,j} = MAX\left\{\begin{array}{c} H_{i-1,j-1} + S_{i,j} \\ \underset{k}{MAX\left\{H_{i-k,j} - (gop + gep * k)\right\}} \\ MAX\left\{H_{i,j-l} - (gop + gep * l)\right\} \end{array}\right\}$$

Needleman-Wunsch algorithm

where:

$H_{i,j}$ is the score for the alignment that ends at vertex i,j;
$S_{i,j}$ is the substitution score for aligning the two residues at vertex i,j when profiles are individual sequences;
$S_{i,j}$ equals $PS_{i,j}$ for subsequent profile alignments.

Profile_1: two ADEGH

	A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**GOP1**	**GEP1**
A	10	0	7	0	0	-1	-1	0	0.5	0.05
D	0	0	0	10	0	5	0	0	0.5	0.05
E	0	0	0	0	10	0	0	0	0.5	0.05
G	-1	0	0	0	0	0	10	0	0.5	0.05
H	0	0	0	0	0	0	0	10	0	0

Profile_2: four DEFFFG

	A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**GOP2**	**GEP2**
D	0	0	0	1	0	0	0	0	0.5	0.05
E	0	0	0	0	1	0	0	0	0.5	0.05
F	0	0	0	0	0	1	0	0	0.5	0.05
F	0	0	0	0	0	1	0	0	0.5	0.05
F	0	0	0	0	0	1	0	0	0.5	0.05
G	0	0	0	0	0	0	1	0	0	0

Alignment path for profiles 1 and 2:

	A	D	E	G	H
D	0	↘ 10	0	0	0
E	0	⇘ 0	↘ 20	⇘ 0	⇘ 0
F	-1	⇩ 5	⬇ 18.95	⇘ 20	⇨ 19.45
F	-1	⇘ 4	⬇ 18.9	⇩ 18.95	⇘ 20
F	-1	⇘ 4	⬇ 18.85	⇘ ⇩ 18.9	⇘ ⇩ 18.95
G	-1	⇩ 2.95	⇩ 18.8	↘ 28.85	➔ 28.85

The score for the global alignment is found in the bottom, right-hand vertex:

Score = 10+10 – (0.5+0.5+ 3*0.05) +10= 28.85

Global Alignment :

```
two      ADE---GH
four     -DEFFFG-
```

3.2 Step 2 - align profile of sequences (two, four) with sequence five

Position-specific gap opening and extension calculations for profiles

In ClustalG, a table of gap opening and extension penalties for every position in the profile is calculated, in order to make gaps more or less likely at different positions. Both the gap opening and gap extension penalties are reduced at positions containing gaps:

$$GOP(j) = GOP * 0.3 * \frac{Number\_of\_sequences\_with\_residues}{Number\_of\_sequences}$$

$$GEP(j) = \frac{GEP}{2}$$

The gap opening penalty is increased at conserved positions (i.e. with no gaps) near existing gaps:

$$GOP(j) = GOP * (2 + 2 * \left(\frac{gap\_separation - distance\_from\_gap}{gap\_separation} \right)$$

$$GEP(j) = GEP$$

In the example, GOP = 0.5, GEP = 0.05, and end gaps are not penalised. Gap separation parameter = 8 (this is the default).

For the alignment for profile 1:

```
two      ADE---GH
four     -DEFFFG-
```

Column 1 is conserved, since gaps at the ends of sequences are not considered as gaps, and is 3 positions from the gap in column 4. So,

GOP1(1) = GOP*(2+2*(8-3)/8) = 1.625

Column 2 is conserved and is 2 positions from the gap in column 4. So,

GOP1(2) = GOP*(2+2*(8-2)/8) = 1.75

Similarly for column 3, which is 1 position from the gap in column 4:

GOP1(3) = GOP*(2+2*(8-1)/8) = 1.875

For column 4, gaps already exist at this position, so for columns 4, 5, and 6:

GOP1(4) = GOP*0.3*1/2 = 0.75

GEP1(4)=GEP/2 = 0.025

	A -	D D	E E	- F	- F	- F	G G	H -
GOP1	1.625	1.75	1.875	0.075	0.075	0.075	1.875	0
GEP1	0.05	0.05	0.05	0.025	0.025	0.025	0.05	0

The GOP and GEP for each position in both alignments are shown in the profile tables below.

Profile Calculation:

Profile_1 is based on the alignment of sequences two, four, the original substitution matrix and the character weights, w_d:

```
two      ADE---GH
four     -DEFFFG-
```

Recall that the profile cell is determined by a character weight, w_d, the substitution score $S_{i,j}$, and the proportion of non-gap characters at the position. If sequence weights are normalized such that the total weight=1.0, the denominator of the formula for the character weight becomes 1.0. In this example, the weights for sequences two and four are normalized. Sequence weights were derived in section 2.3.

Weight for sequence two = 0.141
Normalized weight for two = 0.141/(0.141+0.162) = 0.465

Weight for sequence four = 0.162
Normalized weight for four = 0.162/(0.141+0.162) = 0.535

Example profile calculations:

Profile_1(1,A) = (0.465*1+0.535*0)* S(A,A) *(1/1) = 0.465*10 = 4.65

Profile_1(1,B) = (0.465*0+0.535*0)* S(A,B) *(1/1) = 0

Profile_1(1,C) = (0.465*1+0.535*0)* S(A,C) *(1/1) = 0.465*7 = 3.255

Profile_1(2,D) = (0.465*1+0.535*1)*S(D,D) *(2/2) = 1*10 = 10

Profile_1(4,A) = (0.465*0+0.535*1)*S(F,A) (1/2) = 0.535*-1 / 2 = -0.267

Profile_1(4,D) = (0.465*0+0.535*1)*S(F,D) (1/2) = 0.535*5 / 2 = 1.33

Profile_1(4,F) = (0.465*0+0.535*1)*S(F,F) *(1/2) = 0.535*10 / 2 = 2.67

Profile_1(7,A) = (0.465*1+0.535*1)*S(G,A) *(2/2) = 1*10 = 2.67

```
Profile 1:  two      ADE---GH
            four     -DEFFFG-
```

	A	B	C	D	E	F	G	H	GOP	GEP
1 A-	4.65	0	3.25	0	0	-0.46	-0.46	0	1.625	0.05
2 DD	0	0	0	10	0	5	0	0	1.75	0.05
3 EE	0	0	0	0	10	0	0	0	1.875	0.05
4 -F	-0.26	0	0	1.33	0	2.67	0	0	0.075	0.025
5 -F	-0.26	0	0	1.33	0	2.67	0	0	0.075	0.025
6 -F	-0.26	0	0	1.33	0	2.67	0	0	0.075	0.025
7 GG	-1	0	0	0	0	0	10	0	1.875	0.05
8 H-	0	0	0	0	0	0	0	4.65	0	0

Profile 2: five CCDDE

	A	B	C	D	E	F	G	H	GOP	GEP
C	0	0	1	0	0	0	0	0	0.5	0.05
C	0	0	1	0	0	0	0	0	0.5	0.05
D	0	0	0	1	0	0	0	0	0.5	0.05
D	0	0	0	1	0	0	0	0	0.5	0.05
E	0	0	0	0	1	0	0	0	0	0

The alignment path for profiles 1 and 2 is constructed as for a pairwise alignment.

PS(1,1) = Profile_1(r1) * Profile_2 (r1) = 3.25
PS(1,j) = Profile_1(rj) * Profile_2 (r1) = 0
PS(2,1) = Profile_1(r1) * Profile_2 (r2) = 3.25 (no gap penalty for the end gap)
PS(2,2) = Profile_1(r2) * Profile_2 (r1) + PS(1,1) = 0 + 3.25

Cell (2,1) is the product of Profile_1 row1 and Profile_2 row2. Cell (2,2) is determined by the substitution value of the product

	1	2	3	4	5	6	7	8
	A	D	E	-	-	-	G	H
	-	D	E	F	F	F	G	-
C	3.25	0	0	0	0	0	0	0
C	3.25	⬂ 3.25	⇨ 0.775	⇨ 0.725	⇨ 0.675	⇨ 0.625	⇨ 0.575	⇨ 0.525
D	⬇ 1.075	⬂ 13.25	⬂ 3.25	⇨ 1.95	⇨ 1.9	⇨ 1.85	⇨ 1.8	⇨ 1.75
D	0	⬊ 11.075	⬂ 13.25	⇨ 11.95	⇨ 11.9	⇨ 11.85	⇨ 11.8	⇨ 11.75
E	0	⇩ 8.78	⬊ 21.075	➔ 21.075	➔ 21.075	➔ 21.075	➔ 21.075	➔ 21.075

The score for the global alignment is found in the bottom, right-hand vertex:

Score = 3.25-(1.625+0.5+0.05)+10+10=21.075

Global Alignment :

```
two                -A-DE---GH
four               ---DEFFFG-
five               CCDDE-----
```

Note that the four gap positions of the two-four alignment have been preserved and that a new gap has been added before the [D] position to accommodate the alignment with sequence five.

In the remainder of the process we align the sequences in the following order:

Step 1: Align Sequence: two with Sequence: four

```
two  ADE---GH
four -DEFFFG-
```

Score : 28.85

Step 2: Align Profile: (two, four) with

```
two  -A-DE---GH
```

Sequence: (five)

```
four ---DEFFFG-
five CCDDE-----
```

Score : 21.075

Step 3: Align Sequence: (one) with
Sequence: (six)

```
one   -ABCDEF
six   AAB-DE-
```

Score : 38.95

Step 4: Align Profile: (one,six) with

Sequence: (three)

```
one    -ABCDEF-
six    AAB-DE--
three  --BBDBFG
```

Score: 25.00

Step 5: Align Profile: (two,four,five) with

Profile: (one,six,three)

```
two       -A--DE---GH
four      ----DEFFFG-
five      CCD-DE-----
one       -ABCDEF----
six       AAB-DE-----
three     --BBDBF--G-
           .. ##:  :
```

Score:19.05

Divergent sequences

The most divergent sequences (most different, on average from all of the other sequences) are usually the most difficult to align correctly. It is sometimes better to delay the incorporation of these sequences until all of the more easily aligned sequences are merged first. This may give a better chance of correctly placing the gaps and matching weakly conserved positions against the rest of the sequences. A choice is offered to set a cut off (default is 40% identity or less with any other sequence) that will delay the alignment of the divergent sequences until all of the rest have been aligned.

Recalculate Pairwise distances

Finally, the pairwise distances are recalculated based on the new multiple alignment using the sum of pairs criterion. For each pair of sequences, the similarity is defined as:

$$\text{Similarity} = \frac{\text{number of identical matches}}{\text{number of residues aligned}}$$

And the pairwise distance is:

$$\text{Distance} = 1 - \text{Similarity}$$

Multiple Alignment:

```
two               -A--DE---GH
four              ----DEFFFG-
five              CCD-DE-----
one               -ABCDEF----
six               AAB-DE-----
three             --BBDBF--G-
                   :: ##:  :
```

	Similarity	**Distance**
Two, four	3/3=1	0
Two,five	2/3=0.667	0.333
Two,one	3/3=1	0
Two,six	3/3=1	0
Two,three	2/3=0.667	0.333
Four,five	2/2=1	0
Four,one	3/3=1	0
Four,six	2/2=1	0
Four,three	3/4=0.75	0.25
Five,one	2/4=0.5	0.5
Five,six	2/5=0.4	0.6
Five,three	1/3=0.333	0667
One,six	4/4=1	0
One,three	3/5=0.6	0.4
Six,three	2/3=0.667	0.333

Distance matrix:

	two	**four**	**five**	**one**	**six**	**three**
two	0.000	0.000	0.333	0.000	0.000	0.333
four	0.000	0.000	0.000	0.000	0.000	0.250
five	0.333	0.000	0.000	0.500	0.600	0.667
one	0.000	0.000	0.500	0.000	0.000	0.400
six	0.000	0.000	0.600	0.000	0.000	0.333
three	0.333	0.2500	0.667	0.400	0.333	0.000

4.0 Phase 4 – Alignment analysis

4.1 Final neighbour-joining tree

Multiple Alignment:

```
two             -A--DE---GH
four            ----DEFFFG-
five            CCD-DE-----
one             -ABCDEF----
six             AAB-DE-----
three           --BBDBF--G-
                 :: ##:  :
```

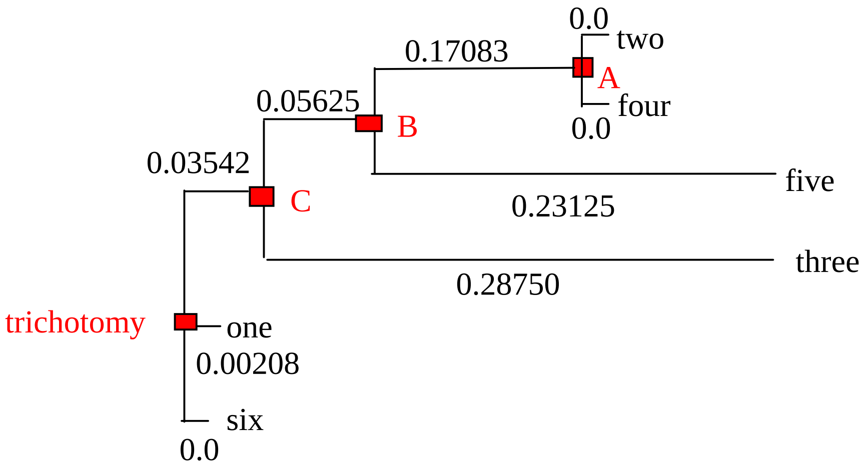
The neighbor-joining algorithm used is exactly the same as that described earlier for the guide tree.

Node A : Sequence: two (0.00000) joins Sequence: four (0.00000)

Node B : Node: A (0.17083) joins Sequence: five (0.23125)

Node C : Sequence: three (0.28750) joins Node: B (0.05625)

Node D : (Last cycle, trichotomy): Node: C (0.03542) joins Sequence: one (0.00208) joins Sequence: six (0.00000)



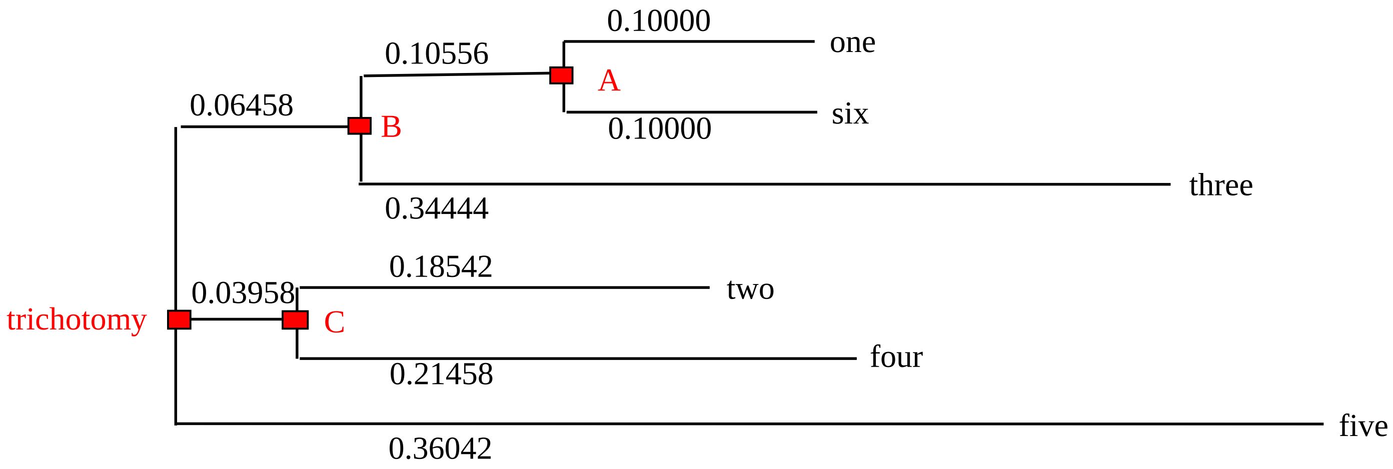
In New Hampshire file format:
((two:0.00000,
((four:0.00000,
five:0.23125)
:0.17083,
three:0.28750)
:0.05625)
:0.03542,
one:0.00208,
six:0.00000);

Comparison of trees before and after multiple alignment

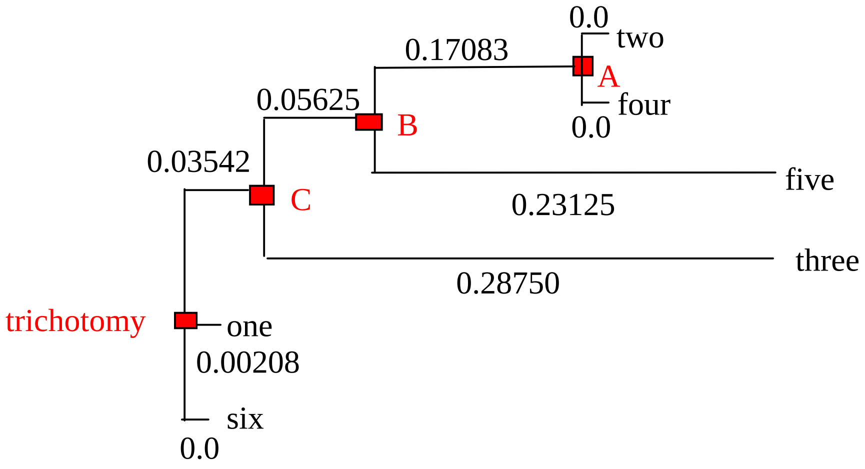
Multiple Alignment:

```
two                -A--DE---GH
four               ----DEFFFG-
five               CCD-DE-----
one                -ABCDEF----
six                AAB-DE-----
three              --BBDBF--G-
                    :: ##:   :
```

Guide tree calculated from pairwise alignments:



Final neighbor-joining tree from multiple alignment:



The main difference is that sequence three has moved from the group with sequences one, six to join the group with two, four and five. Sequence three is the most divergent sequence as it does not contain the DE 'motif' conserved in the other sequences. Sequences one and six share the AB motif, that sequence three does not have. Sequence 3 does contain the G activity present only in sequences two and four.

4.2 Quality analysis

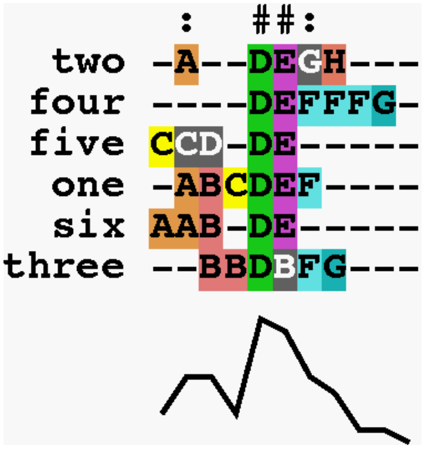
Matrix used for alignment analysis:

$	10
[A:C]	7
[D:F]	5
[A:F]	-1

[A:G]	-1
[G:F]	-1
[G:H]	-1
?	0

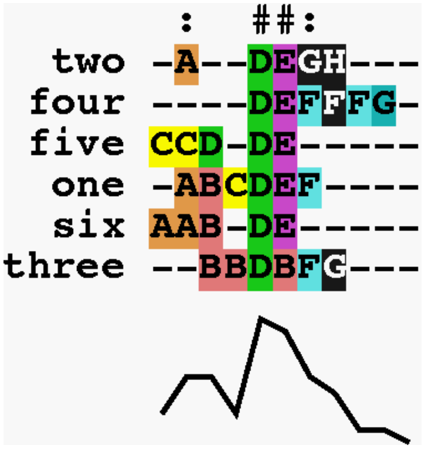
(The aligment and the matrix have been changed from those in the example above, in order to force some errors and to demonstrate the error detection calculations).

exceptional activities



```
       :  ##:
two   -A--DEGH---
four  ----DEFFFG-
five  CCD-DE-----
one   -ABCDEF----
six   AAB-DE-----
three --BBDBFG---
```

low-scoring segments



```
       :  ##:
two   -A--DEGH---
four  ----DEFFFG-
five  CCD-DE-----
one   -ABCDEF----
six   AAB-DE-----
three --BBDBFG---
```

Firstly, ClustalG allows the user to select the colors used to highlight certain activities. In this case, a simple scheme has been used that colors all activities, regardless of the conservation at each position. The conservation is indicated in two ways: the characters #,*,:,. above the alignment, and the conservation profile below the alignment.

Conservation profile

The conservation at each position is estimated based on a distance analysis in an N-dimensional sequence space (where N is the number of characters in the alphabet). For each column in the alignment, a consensus sequence is calculated based on the activities observed in this column. The conservation score is based on the mean distance of each sequence from the consensus position. Thus, if the column is conserved, the distance of each sequence from the consensus is zero, and the conservation score is 1. For less well conserved columns, the distances increase and the score tends towards to zero. The user can select the scoring matrix used to calculate the consensus point and the sequence distances.

Exceptional Activities

For each column in the alignment, those sequences that are found a long way from the consensus point can be highlighted in grey. The scaling factor used for the exception calculation can be adjusted by the user to select the proportion of exceptional activities displayed.

Low scoring segments

The low scoring segment calculation is based on a profile of the alignment. (see profile calculation in multiple alignment section). For each sequence, the scores for matching the sequence against the profile are summed in both forward and backward directions. The low-scoring segments (those segments that score negatively in both directions) can be highlighted in black. The user can select the scoring matrix used to calculate the profile.

5. References

Abbott A, 1995, "Sequence analysis: new methods for old ideas" *Annual Review of Sociology* v.21, pp.93-113

Feng, D F and Doolittle, R F, 1987, progressive multiple alignment, *Journal of Molecular Evolution*, v25, pp. 351-360

Higgins D, and Sharp P, 1988, CLUSTAL: a package for performing multiple sequence alignment on a microcomputer, <u>Gene</u> v.73, pp.237-244.

Needleman S and Wunsch C, 1970 A general method applicable to the search for similarities in the amino acid sequence of two proteins, *Journal of Molecular Biology*, 48:443-453.

Saitou and Nei. Reconstructing phylogenetic trees, *Molecular Biology and Evolution*, vol. 4, no. 4, p. 406.

Sankoff, D. and J. Kruskal, *Time Warps, String Edits and Macromolecules, The Theory and Practice of Sequence Comparison*, Addison-Wesley, Reading Mass., 1983

Thompson, J.D., Higgins, D.G. and Gibson, T.J. 1994 CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, positions-specific gap penalties and weight matrix choice. *Nucleic Acids Research*, 22:4673-4680.

Thompson,J.D., Gibson,T.J., Plewniak,F., Jeanmougin,F. and Higgins,D.G.1997, The ClustalX windows interface: flexible strategies for multiple sequence alignment aided by quality analysis tools. *Nucleic Acids Research*, 24:4876-4882.

Thompson,J.D., Plewniak, F, Jeanmougin, and Poch, O, 1999, A comprehensive comparison of multiple sequence alignment programs, *Nucleic Acids Research*, 27:13, pp. 2682-2690.

Waterman, M. *Introduction to Computational Biology*, Chapman and Hall, London, 1995.

Wilson C, Activity pattern analysis by means of sequence alignment methods, *Environment and Planning A*, 1998, vol. 30, pp. 1017-1038.

Wilson, C. 2001. Activity patterns of Canadian women: an application of ClustalG sequence alignment software, *Transportation Research Reco*rd, number 1777, pages 55-67.