Multiple Sequence Alignment

Reference: Gusfield, *Algorithms on Strings, Trees & Sequences*, chapter 14

Some slides from:

- Jones, Pevzner, USC *Intro to Bioinformatics Algorithms*
- Ruzzo, Tompa U. Washington CSE 590bi
- Poch, Strasbourg [www.inra.fr/internet/Projets/agroBI/PHYLO/Poch.ppt](http://www.inra.fr/internet/Projets/agroBI/PHYLO/Poch.ppt)
- A. Drummond, Auckland, NZ

Revised Nov 2015
Multiple Alignment vs. Pairwise Alignment

- Up until now we have only tried to align two sequences.
- What about more than two? And what for?
- A faint similarity between two sequences becomes significant if present in many.
- Multiple alignments can reveal subtle similarities that pairwise alignments do not reveal.
Multiple Alignment vs. Pairwise Alignment

• “Pairwise alignment whispers ... multiple alignment shouts out loud”
  Hubbard, Lesk, Tramontano, Nature Structural Biology 1996.
Multiple Alignment Definition

**Input:** Sequences $S_1, S_2, \ldots, S_k$ over the same alphabet

**Output:** Gapped sequences $S'_1, S'_2, \ldots, S'_k$ of equal length

1. $|S'_1| = |S'_2| = \ldots = |S'_k|

2. Removal of spaces from $S'_i$ gives $S_i$ for all $i$
Example

$S_1 = \text{AGGTC}$

$S_2 = \text{GTTCG}$

$S_3 = \text{TGAAC}$

Possible alignment

$$
\begin{array}{cccccc}
A & G & G & T & - & C \\
- & G & - & T & T & C & G \\
T & G & - & A & A & C & - \\
\end{array}
S'_1 \quad S'_2 \quad S'_3
$$

Possible alignment

$$
\begin{array}{cccccc}
A & G & G & T & - & C \\
G & T & T & - & C & G \\
- & T & G & A & A & C \\
\end{array}
S'_1 \quad S'_2 \quad S'_3
$$

$|S'_1| = |S'_2| = |S'_3|$
Example 1

Multiple sequence alignment of 7 neuroglobins using clustalx

Identify and represent protein families.
Example 2


Identify and represent conserved motifs (conserved common biological function).
Protein Phylogenies - Example 3

Deduce evolutionary history
Motivation again

• Common structure, function or origin may be only weakly reflected in sequence - multiple comparisons may highlight weak signals

• Major uses:
  - Identify and represent protein families
  - Identify and represent conserved seq. elements (e.g. domains)
  - Deduce evolutionary history
MSA: central role in biology

Comparative genomics

Gene identification, validation

RNA sequence, structure, function

Human genetics, SNPs

Phylogenetic studies

Hierarchical function annotation:
- homologs, domains, motifs

Structure comparison, modelling

Interaction networks

Therapeutics, drug design

DBD LBD insertion domain binding sites / mutations
Scoring alignments

- Given input seqs. $S_1, S_2, \ldots, S_k$ find a multiple alignment of optimal score

- Scores preview:
  - Sum of pairs
  - Consensus
  - Tree
Sum of Pairs score

**Def:** Induced pairwise alignment

A pairwise alignment induced by the multiple alignment

Example:

- $x$: AC–GCGG–C
- $y$: AC–GC–GAG
- $z$: GCCGC–GAG

Induces:

- $X'$: ACGCGG–C; $x'$: AC–GCGG–C; $y'$: AC–GCGAG
- $Y'$: ACGC–GAC; $z'$: GCCGC–GAG; $z'$: GCCGCGAG

$$\text{SOP}(M) = \sum_{k<l} \sigma(S'_k, S'_l)$$
SOP Score Example

Consider the following alignment:

```
AC-CDB
--C-ADBD
A-BCDAD
```

Scoring scheme:  
- match - 0
- mismatch/indel - -1

SP score: -3 -5 -4 = -12
Optimal MSA
Alignments = Paths

- Align 2 sequences: ATGC, AATC
Alignment Paths

- Align 2 sequences: ATGC, AATC

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>--</th>
<th>T</th>
<th>G</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>A</td>
<td>A</td>
<td>T</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

x coordinate
Alignment Paths

• Align 2 sequences: ATGC, AATC

\[
\begin{array}{cccccc}
0 & 1 & 1 & 2 & 3 & 4 \\
A & -- & T & G & C \\
0 & 1 & 2 & 3 & 3 & 4 \\
A & A & T & -- & C \\
\end{array}
\]

x coordinate

y coordinate
Alignment Paths

- Resulting path in \((x,y)\) space:

\[(0,0) \rightarrow (1,1) \rightarrow (1,2) \rightarrow (2,3) \rightarrow (3,3) \rightarrow (4,4)\]
Alignments = Paths

• Align 3 sequences: ATGC, AATC, ATGC

<table>
<thead>
<tr>
<th>A</th>
<th>--</th>
<th>T</th>
<th>G</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>A</td>
<td>T</td>
<td>--</td>
<td>C</td>
</tr>
<tr>
<td>--</td>
<td>A</td>
<td>T</td>
<td>G</td>
<td>C</td>
</tr>
</tbody>
</table>
### Alignment Paths

- Align 3 sequences: ATGC, AATC, ATGC

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td></td>
<td>--</td>
<td>T</td>
<td>G</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>A</td>
<td>A</td>
<td>T</td>
<td>--</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>--</td>
<td>A</td>
<td>T</td>
<td>G</td>
<td>C</td>
</tr>
</tbody>
</table>
Alignment Paths

• Align 3 sequences: ATGC, AATC, ATGC

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>A</td>
<td>T</td>
<td></td>
<td>G</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>A</td>
<td>T</td>
<td>--</td>
<td>C</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

x coordinate

y coordinate
Alignment Paths

- Align 3 sequences: ATGC, AATC, ATGC

<table>
<thead>
<tr>
<th>0 1 1 2 3 4</th>
<th>x coordinate</th>
</tr>
</thead>
<tbody>
<tr>
<td>A -- T G C</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>0 1 2 3 3 4</th>
<th>y coordinate</th>
</tr>
</thead>
<tbody>
<tr>
<td>A A T -- C</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>0 0 1 2 3 4</th>
<th>z coordinate</th>
</tr>
</thead>
<tbody>
<tr>
<td>-- A T G C</td>
<td></td>
</tr>
</tbody>
</table>

- Resulting path in \((x,y,z)\) space:

\[(0,0,0) \rightarrow (1,1,0) \rightarrow (1,2,1) \rightarrow (2,3,2) \rightarrow (3,3,3) \rightarrow (4,4,4)\]
2-D vs 3-D Alignment Grid

2-D edit graph

3-D edit graph
Aligning Three Sequences

- Same strategy as aligning two sequences
- Use a 3-D “Manhattan Cube”, with each axis representing a sequence to align
- For global alignments, go from source to sink
3-D cell versus 2-D Alignment Cell

In 2-D, 3 edges in each unit square

In 3-D, 7 edges in each unit cube
Architecture of 3-D Alignment Cell

(i-1,j-1,k-1) (i-1,j-1,k) (i-1,j,k-1) (i-1,j,k) (i,j-1,k-1) (i,j-1,k) (i,j,k-1) (i,j,k)
Architecture of 3-D Alignment Cell

-Cube diagonal: no indels

-Edge: 2 indels

-Face diagonal: 1 indels
Multiple Alignment: Dynamic Programming

\[ s_{i,j,k} = \max \begin{cases} 
  s_{i-1,j-1,k-1} + \delta(v_i, w_j, u_k) \\
  s_{i-1,j-1,k} + \delta(v_i, w_j, _) \\
  s_{i-1,j,k-1} + \delta(v_i, _, u_k) \\
  s_{i,j-1,k-1} + \delta(_, w_j, u_k) \\
  s_{i,j,k} + \delta(v_i, _, _) \\
  s_{i,j-1,k} + \delta(_, w_j, _) \\
  s_{i,j,k-1} + \delta(_, _, u_k) 
\end{cases} \]

- cube diagonal: no indels
- face diagonal: one indel
- edge: two indels

\[ \delta(x, y, z) \] is an entry in the 3-D scoring matrix
Running Time

• For 3 sequences of length $n$, the run time is $O(n^3)$

• For $k$ sequences, build a $k$-dimensional cube, with run time $O(2^k n^k)$ [another $2^k$ factor for affine gaps]

• Impractical for most realistic cases

• NP-hard (Elias’03 for general matrices)
Minimum cost - SOP

We use min cost instead of max score

→ Find alignment of minimal cost

Observe: opt SOP score ≥ sum of optimal pairwise scores
Forward Dynamic Programming

• An alternative approach to DP. Useful for SOP alignment:
  • $D(v)$ – opt value of path source $\rightarrow v$
  • $p(w)$ – best-yet solution of path source $\rightarrow w$
• When $D(v)$ is computed, send its value forward on the arcs exiting from $v$:
  For $v \rightarrow w$: $p(w) = \min\{p(w), D(v) + \text{cost}(v, w)\}$
• Once $p(w)$ has been updated by all incoming edges - that value is optimal; set as $D(w)$
Forward Dynamic Programming (2)

• Maintain a queue of nodes whose D is not set yet
• For the node w at the head of the queue: Set D(w)←p(w) and remove
• ∀ out-neighbor x of w - update p; if x is not in the queue - add it at the end
  - Breaking ties lexicographically
• Same complexity as the regular (backwards) DP
Faster DP Algorithm for SOP alignment
Carillo-Lipman 88

- Idea: after computing D(v), with a little extra computation, we may already know that v will not on any optimal solution.

- $\forall k, l, k < l$ compute $f_{kl}(i,j) = \text{opt pairwise alignment score of suffixes } S_k(i+1,..n_1), S_l(j+1,..n_2)$.

- Use forward DP.

- If $\exists$ a known soln of cost $z$, and if $D(i,j,k) + f_{12}(i,j) + f_{13}(i,k) + f_{23}(j,k) > z$ → Do not send $D(i,j,k)$ forward

- Guarantees opt SOP alignment - no improved time bound, but often saves a lot in practice.
Approximation algorithms
Approximation Algorithms - assumption

We use min cost instead of max score

→ Find alignment of minimal cost

**Assumption:** the cost function $\delta$ is a *distance function*

- $\delta(x,x) = 0$
- $\delta(x,y) = \delta(y,x) \geq 0$
- $\delta(x,y) + \delta(y,z) \geq \delta(x,z)$ (triangle inequality)

$D(S,T)$ - cost of minimum global alignment between $S$ and $T$
The Center Star algorithm
Gusfield 1993

Input: $\Gamma$ - set of $k$ strings $S_1, \ldots, S_k$.

1. Find the string $S^* \in \Gamma$ (center) that minimizes $\sum_{S \in \Gamma \setminus \{S^* \}} D(S^*, S)$

2. Denote $S_1 = S^*$ and the rest of the strings as $S_2, \ldots, S_k$

3. Iteratively add $S_2, \ldots, S_k$ to the alignment as follows:
   a. Suppose $S_1, \ldots, S_{i-1}$ are already aligned as $S'_1, \ldots, S'_{i-1}$
   b. Optimally align $S_i$ to $S'_1$ to produce $S'_i$ and $S''_1$ aligned
   c. Adjust $S'_2, \ldots, S'_{i-1}$ by adding spaces where spaces were added to $S''_1$
   d. Replace $S'_1$ by $S''_1$
The Center Star algorithm (demonstration)

Y′: ATG-A-
X′: A-G-AC
Z′: ATGGA-

Y′′: A-TG-A-
W′: AGTG-A-

Inheriting gaps

Y′: A-TG-A-
X′: A-G-AC
Z′: A-TGGA-
W′: AGTG-A-
The Center Star algorithm

Running time

• Choosing $S_1$ - execute DP for all sequence-pairs - $O(k^2n^2)$

• Adding $S_i$ to the alignment - execute DP for $S_i, S'_1$ - $O(i \cdot n^2)$.

(In the $i^{th}$ stage the length of $S'_1$ can be up-to $i \cdot n$)

$$
\sum_{i=1}^{k-1} O(i \cdot n^2) = O(k^2 n^2)
$$

↑

total complexity
The Center Star algorithm

Approximation ratio

- $M^*$ - An optimal alignment
- $M$ - The alignment produced by the center-star algorithm
- $d(i,j)$ - The distance $M$ induces on the pair $S_i, S_j$

$v(M) = \sum_{i=1}^{k} \sum_{\substack{j=1 \\ j \neq i}}^{k} d(i, j) = 2 \sum_{i<j} d(i, j)$

• recall $D(S, T)$ - min cost of alignment between $S$ and $T$

For all $i$: $d(1,i) = D(S_1, S_i)$

(we perform optimal alignment between $S'_1$ and $S_i$ and $\delta(-,-) = 0$)
The Center Star algorithm

Approximation ratio (2)

\[\nu(M) = \sum_{i=1}^{k} \sum_{j=1, j \neq i}^{k} d(i, j) \leq \sum_{i=1}^{k} \sum_{j=1, j \neq i}^{k} (d(1, i) + d(1, j)) = 2(k-1) \sum_{l=2}^{k} d(1, l) = 2(k-1) \sum_{l=2}^{k} D(S_1, S_l)\]

\[\nu(M^*) = \sum_{i=1}^{k} \sum_{j=1, j \neq i}^{k} d^*(i, j) \geq \sum_{i=1}^{k} \sum_{j=1, j \neq i}^{k} D(S_i, S_j) \geq \sum_{i=1}^{k} \sum_{j=2}^{k} D(S_1, S_j) = k \sum_{j=2}^{k} D(S_1, S_j)\]

\[\frac{\nu(M)}{\nu(M^*)} \leq \frac{2(k-1)}{k} \leq 2\]

Definition of \(S_1\):

\[\forall i: \sum_{j=2}^{k} D(S_1, S_j) \leq \sum_{j=1, j \neq i}^{k} D(S_i, S_j)\]
The Center Star algorithm Theorem (Gusfield 93)

- We have proved:
- The center star algorithm is a polynomial algorithm that guarantees a solution at most twice the optimum of SOP alignment.

- “a 2-approximation”
- “an approximation ratio of 2”
Steiner String and Consensus MA
Consensus error & Steiner string - definitions

- Input: set of k strings $\Gamma = \{S_1, \ldots, S_k\}$.
- $D(X, Y)$ - score of aligning $X$, $Y$.
- $S$ - arbitrary sequence (unrelated to $\Gamma$).

- The consensus error of $S$ relative to $\Gamma$:
  
  $$E(S) = \sum D(S, S_i)$$

- $S^*$ is an optimal Steiner string for $\Gamma$ if it minimizes $E(S)$.
  - Different objective function - linear no of terms.
  - No direct relation to multialign! (for now)
Thm: Assume $D$ satisfies triangle ineq. Then 
$\exists S \in \Gamma$ that guarantees an approximation ratio 2.

Pf: Pick $S \in \Gamma$

\[
E(S) = \sum_{S \neq S_i} D(S, S_i) \leq \sum_{S \neq S_i} (D(S, S^*) + D(S^*, S_i))
\]

\[
= (k - 2)D(S, S^*) + D(S, S^*) + \sum_{S_i \neq S} D(S^*, S_i)
\]

\[
= (k - 2)D(S, S^*) + E(S^*)
\]

Pick $S \in \Gamma$ closest to $S^*$ (not constructively)

\[
E(S^*) = \sum_{S_i \in \Gamma} D(S^*, S_i) \geq k \cdot D(S, S^*)
\]

\[
\frac{E(S)}{E(S^*)} \leq \frac{(k - 2)}{k} + 1 < 2
\]
Optimal Steiner String: Approximation

Resulting algorithm:
Pick $S_c \in \Gamma$ that minimizes $E(S_c)$ ($S_c$ is the center string).

Approximation:
$S_c$ gives a 2-approximation. The center string has a consensus error at most 2 times the consensus error of the optimal Steiner string.
Consensus String

- Given multiple alignment, the *consensus character* $i$ is the character that minimize the summed distance to it from all the characters in column $i$ of the MA.

- For example, given scoring scheme with match = 0, mismatch or indel=1, the consensus character is the most frequent character.

- The *consensus string* derived from alignment is the concentration of consensus characters.
Consensus string versus Steiner string definitions

- $d(S, T)$ - The distance that a given multiple alignment $M$ induces on the pair $S, T$
- $D(S, T)$ - min cost of alignment between $S$ and $T$

- Optimal Steiner string $S^*$: minimizes $\Sigma_i D(S^*, Si)$
- Optimal consensus string $S_{con}$: minimizes $\Sigma_i d(S_{con}, S'i)$
Consensus multiple alignment

- $S_{\text{con}}$: AC-GC-GAG
- x: AC-GCGG-C
- y: AC-GC-GAG
- z: GCCGA-GAG
- u: AC-T-GGCA
- v: -CAGT-GAG
- w: AC-GC-GAG

Alignment error$(M) = \Sigma_i d(S_i, S_{\text{con}})$

The opt consensus MA: one with least alignment error
**Consensus multiple alignment**

**Thm:** optimal solution of consensus

\[ MA = \text{Steiner string (up to spaces)} \]

- Opt consensus string = Steiner string (up to spaces)
- Optimal alignment error = optimal consensus error

**Pf:** ex.

Gusfield theorems 14.7.2 and 14.7.3
A summary

\[ S_c \]
Center string

\[ S^* \]
Optimal Steiner string

\[ S_{\text{con}} \]
Optimal consensus string

Center star algorithm

Consensus error

Alignment error

\[ \Sigma_i D(S_c, S_i) \]

\[ \Sigma_i D(S^*, S_i) \]

\[ \Sigma_i d(S_{\text{con}}, S'_i) \leq \Sigma_i d(S_c, S'_i) \]

\[ = \Sigma_i D(S_c, S_i) \]

\[ \Sigma_i d(S_{\text{con}}, S'_i) \]

\[ \leq \Sigma_i d(Sc, S'_i) \leq \Sigma_i D(Sc, S_i) \]

\[ = \Sigma_i D(Sc, S_i) \]

Optimal (consensus) alignment error

Optimal consensus error

Optimal alignment error
A summary

Optimal SOP alignment

Optimal SOP score

2-approximation

Optimal SOP score

Center star algorithm

Center string

Optimal Steiner string

Optimal consensus string

Optimal consensus alignment

Alignment error

<=

Consensus error

<=

Optimal consensus error

Optimal (consensus) alignment error

2-approximation

Pf: ex.
A summary

Optimal (consensus) alignment error

Optimal consensus string

Center string

Optimal Steiner string

Optimal consensus string

Center star algorithm

Optimal SOP alignment

Optimal SOP score

SOP score

Alignment error

Consensus error

Optimal consensus error

Pf: ex.

2-approximation

Optimal (consensus) alignment error

2-appr
Theorem

• Assuming the triangle inequality, the multiple alignment created by the center star method has:
  - A sum-of-pairs (SOP) score that is never more than 2 times the SOP score of the optimal SOP alignment
  - An alignment error and that is never more than 2 times the alignment error of the optimal consensus alignment.
“Tree alignment”, “phylogenetic alignment”
Scoring alignments

• Given input seqs. $S_1, S_2, ..., S_k$ find a multiple alignment of optimal score

• Scores preview:
  - Sum of pairs
  - Consensus
  - Phylogenetic tree

  Optimal SOP alignment
  Optimal Consensus alignment
  Optimal phylogenetic/tree alignment
Phylogenetic/tree MA

- Input: Tree T, a string for each leaf
- Phylogenetic/tree alignment problem: Assignment of a string to each internal node in T

- Score - sum of scores along edges \( \Sigma_{(i,j) \in T} D(S_i, S_j) \)
- Goal: find tree alignment of optimal score
- Consensus = tree Alignment where T is a star
Tree MA - complexity

- NP-hard
- **Lifted alignment**: efficient heuristic method. Poly time, 2-approximation assuming triangle inequality

- In recitation
Multiple Alignment: Greedy Heuristic
Profile Representation of MA

- A G G C T A T C A C C T G
  T A G - C T A C C A - - - G
  C A G - C T A C C A - - - G
  C A G - C T A T C A C - G G
  C A G - C T A T C G C - G G

A 1 1 .8
C .6 1 .4 1 .6 .2
G 1 .2 .2 .4 1
T .2 1 .6 .2
- .2 .8 .4 .8 .4

• Alternatively, use log odds:
• \( p_i(a) = \) fraction of \( a \)'s in col \( i \)
• \( p(a) = \) fraction of \( a \)'s overall
• \( \log p_i(a)/p(a) \)
Fig. 2. Profile of the *Xenopus laevis* transcription factor TFIIIA zinc finger. The eight repeats of the zinc finger sequence that form the probe are shown descending vertically at the left, labeled with the positions where they occur in the complete sequence. Insertions made to align the sequences are shown as periods. The profile calculated by PROFILEMAKE is shown in the box. The rows correspond to the positions in the aligned sequences, and the columns contain the score for each possible amino acid residue when aligned at that position. The position-specific gap penalties are given in the two right-hand columns. The consensus sequence is shown immediately to the left of the box, and represents the highest scoring column at each row in the profile. In other words, the consensus residue is the amino acid that would receive the highest score when aligned with that position in the aligned probe sequences.
Aligning a sequence to a profile

• Key in pairwise alignment is scoring two letters $x, y$: $\sigma(x, y)$
• For a letter $x$ and a column $C$ in a profile, $\sigma(x, C) =$ value of $x$ in col. $C$
• Invent a score for $\sigma(x, -)$
• Run the DP alg for pairwise alignment
Aligning alignments

• Given two alignments, how can we align them?
• Hint: use DP on the corresponding profiles.

\[
\begin{align*}
&x \quad GGGCACTGCAT \\
y & \quad GGTTACGTC-- \\
z & \quad GGGAACTGCAG \\
w & \quad GGACGTACC-- \\
v & \quad GGACCT------ \\
\end{align*}
\]

\[
\begin{align*}
&x \quad GGGCACTGCAT \\
y & \quad GGTTACGTC-- \\
z & \quad GGGAACTGCAG \\
w & \quad GGACGTACC-- \\
v & \quad GGACCT------ \\
\end{align*}
\]
Profile-profile scoring

• Fix a position in the alignment
  - $p_i$ - prob (i in 1st profile); $q_i$ - in 2nd profile
• Expected score: $\sum_{ij} p_i q_j \sigma(i,j)$
• Other scores in use:
  - Euclidean distance
  - Pearson correlation
  - KL-divergence (relative entropy)
Multiple Alignment: Greedy Heuristic

- Choose most similar pair of sequences and combine into a profile, thereby reducing alignment of $k$ sequences to an alignment of $k-1$ sequences/profiles. Repeat
Progressive Alignment

• A variation of greedy algorithm with a somewhat more intelligent strategy for choosing the order of alignments.
Progressive alignment

Align sequences (pairwise) in some (greedy) order

Decisions

(1) Order of alignments
(2) Alignment of group to group
(3) Method of alignment, and scoring function
Guide tree

does one prefer this?

or this?

A
B
C
D
E

A
B
C
D
E
F
• Popular multiple alignment tool today
• Three-step process
  1.) Construct pairwise alignments
  2.) Build guide tree
  3.) Progressive alignment guided by the tree
Step 1: Pairwise Alignment

- Aligns each pair of sequences, giving a similarity matrix
- Similarity = exact matches / sequence length (percent identity)

\[
\begin{array}{cccc}
  v_1 & v_2 & v_3 & v_4 \\
  v_1 & - & \text{.17} & \text{.87} & \text{.59} \\
  v_2 & \text{.17} & - & \text{.28} & \text{.33} \\
  v_3 & \text{.87} & \text{.28} & - & \text{.62} \\
  v_4 & \text{.59} & \text{.33} & \text{.62} & - \\
\end{array}
\]

(.17 means 17 % identical)
Step 2: Guide Tree

- Use the similarity method to create a guide tree by applying some clustering method*

- Guide tree roughly reflects evolutionary relations

  - *ClustalW uses the neighbor-joining method (to be described later in the course)
Step 2: Guide Tree (cont'd)

Calculate:

\[ \mathbf{v}_{1,3} = \text{alignment} \left( \mathbf{v}_1, \mathbf{v}_3 \right) \]
\[ \mathbf{v}_{1,3,4} = \text{alignment} \left( \left( \mathbf{v}_{1,3}, \mathbf{v}_4 \right) \right) \]
\[ \mathbf{v}_{1,2,3,4} = \text{alignment} \left( \left( \mathbf{v}_{1,3,4}, \mathbf{v}_2 \right) \right) \]
Step 3: Progressive Alignment

- Start by aligning the two most similar sequences
- Using the guide tree, add in the most similar pair (seq-seq, seq-prof or prof-prof)
- Insert gaps as necessary
- Many ad-hoc rules: weighting, different matrices, special gap scores....

Dots and stars show how well-conserved a column is.
Multiple Alignment: History

1975 Sankoff
Formulated multiple alignment problem and gave dynamic programming solution

1988 Carrillo-Lipman
Branch and Bound approach for MSA

1990 Feng-Doolittle
Progressive alignment

1994 Thompson-Higgins-Gibson-ClustalW \(\leftarrow\) >40K citations!
Most popular multiple alignment program

1998 Morgenstern et al.-DIALIGN
Segment-based multiple alignment

2000 Notredame-Higgins-Heringa-T-coffee
Using the library of pairwise alignments

2002 MAFFT
2004 MUSCLE
2005 ProbCons
2011 Clustal Omega

Still a lot to be done!