Multiple Alignment
Multiple Alignment versus Pairwise Alignment

- Up until now we have only tried to align two sequences.
Multiple Alignment versus Pairwise Alignment

- Up until now we have only tried to align two sequences.
- What about more than two? And what for?
Multiple Alignment versus 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
Generalizing the Notion of Pairwise Alignment

• Alignment of 2 sequences is represented as a 2-row matrix
• In a similar way, we represent alignment of 3 sequences as a 3-row matrix

\[
\begin{align*}
\text{AT} & \quad \_ & \quad \text{G} & \quad \text{C} & \quad \text{G} & \quad \_ \\
\text{A} & \quad \_ & \quad \text{C} & \quad \text{G} & \quad \text{T} & \quad \_ & \quad \text{A} \\
\text{A} & \quad \text{T} & \quad \text{C} & \quad \text{A} & \quad \text{C} & \quad \_ & \quad \text{A} \\
\end{align*}
\]

• Score: more conserved columns, better alignment
Alignments = Paths in...

- Align 3 sequences: ATGC, AATC, ATGC
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></td>
<td>A</td>
<td>--</td>
<td>T</td>
<td>G</td>
<td>C</td>
<td>x coordinate</td>
</tr>
<tr>
<td></td>
<td>A</td>
<td>A</td>
<td>T</td>
<td>--</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>--</td>
<td>A</td>
<td>T</td>
<td>G</td>
<td>C</td>
<td></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>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>
<tr>
<td></td>
<td>--</td>
<td>A</td>
<td>T</td>
<td>G</td>
<td>C</td>
<td></td>
</tr>
</tbody>
</table>

\( x \) coordinate

\( y \) coordinate
Alignment Paths

- Align 3 sequences: ATGC, AATC, ATGC

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

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

(0,0,0) → (1,1,0) → (1,2,1) → (2,3,2) → (3,3,3) → (4,4,4)
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
2-D vs 3-D Alignment Grid

2-D edit graph

3-D edit graph
2-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
Multiple Alignment: Dynamic Programming

• \( s_{i,j,k} = \max \) \[
\begin{align*}
  s_{i-1,j-1,k-1} &+ \delta(v_i, w_j, u_k) \\
  s_{i-1,j-1,k} &+ \delta(v_i, w_j, _) \\
  s_{i-1,j,k-1} &+ \delta(v_i, _, u_k) \\
  s_{i,j-1,k-1} &+ \delta(_, w_j, u_k) \\
  s_{i-1,j,k} &+ \delta(v_i, _, _) \\
  s_{i,j-1,k} &+ \delta(_, w_j, _) \\
  s_{i,j,k-1} &+ \delta(_, _, u_k)
\end{align*}
\]
cube diagonal: no indels
face diagonal: one indel
edge diagonal: two indels

• \( \delta(x, y, z) \) is an entry in the 3-D scoring matrix
Multiple Alignment: Running Time

• For 3 sequences of length $n$, the run time is $7n^3; O(n^3)$

• For $k$ sequences, build a $k$-dimensional Manhattan, with run time $(2^k-1)(n^k); O(2^kn^k)$

• Conclusion: dynamic programming approach for alignment between two sequences is easily extended to $k$ sequences but it is impractical due to exponential running time
Multiple Alignment Induces Pairwise Alignments

Every multiple alignment induces pairwise alignments

\[ x: \text{AC--GCGG--C} \]
\[ y: \text{AC--GC--GAG} \]
\[ z: \text{GCCGC--GAG} \]

Induces:

\[ x: \text{ACGCGG--C}; \quad y: \text{AC--GCGG--C}; \quad z: \text{GCCGC--GAG} \]
\[ y: \text{ACGC--GAC}; \quad z: \text{GCCGCGGAG} \]
Reverse Problem: Constructing Multiple Alignment from Pairwise Alignments

Given 3 arbitrary pairwise alignments:

\[
\begin{align*}
  x : & \text{ACGCTGG--C;} & x : & \text{AC-GCTGG--C;} & y : & \text{AC-GC-GAG} \\
  y : & \text{ACGC--GAC;} & z : & \text{GCCGCA--GAG;} & z : & \text{GCCGCAGAG}
\end{align*}
\]

can we construct a multiple alignment that induces them?
Reverse Problem: Constructing Multiple Alignment from Pairwise Alignments

Given 3 arbitrary pairwise alignments:

\[ x: \text{ACGCTGG-C}; \quad x: \text{AC-GCTGG-C}; \quad y: \text{AC-GC-GAG} \]
\[ y: \text{ACGC--GAC}; \quad z: \text{GCCGCA--GAG}; \quad z: \text{GCCGCA-GAG} \]

Can we construct a multiple alignment that induces them?  

**NOT ALWAYS**

Pairwise alignments may be inconsistent
Inferring Multiple Alignment from Pairwise Alignments

- From an optimal multiple alignment, we can infer pairwise alignments between all pairs of sequences, but they are not necessarily optimal.
- It is difficult to infer a “good” multiple alignment from optimal pairwise alignments between all sequences.
Combining Optimal Pairwise Alignments into Multiple Alignment

Can combine pairwise alignments into multiple alignment

Can **not** combine pairwise alignments into multiple alignment
Profile Representation of Multiple Alignment

- 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

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>1</th>
<th>.8</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>.6</td>
<td>1</td>
<td>.4</td>
</tr>
<tr>
<td>G</td>
<td>1</td>
<td>2</td>
<td>.2</td>
</tr>
<tr>
<td>T</td>
<td>.2</td>
<td>1</td>
<td>.6</td>
</tr>
<tr>
<td>-</td>
<td>.2</td>
<td>.8</td>
<td>.4</td>
</tr>
</tbody>
</table>
Profile Representation of Multiple Alignment

In the past we were aligning a sequence against a sequence.

Can we align a sequence against a profile?

Can we align a profile against a profile?
Aligning alignments

- Given two alignments, can we align them?

\[
\begin{align*}
x & \quad \text{GGGCACTGCAT} \\
y & \quad \text{GGTTACGTC--} \quad \text{Alignment 1} \\
z & \quad \text{GGGAACTGCAG} \\
w & \quad \text{GGACGTACC--} \quad \text{Alignment 2} \\
v & \quad \text{GGACCT-----}
\end{align*}
\]
Aligning alignments

- Given two alignments, can we align them?
- Hint: use alignment of corresponding profiles

\[
\begin{align*}
x & \quad \text{GGGCACTGCAT} \\
y & \quad \text{GGTTACGTC}\quad \text{Combined Alignment} \\
z & \quad \text{GGGAACGTGCAG} \\
w & \quad \text{GGACGTACC}\quad \\
v & \quad \text{GGACCT}\\n\end{align*}
\]
Multiple Alignment: Greedy Approach

- Choose most similar pair of strings and combine into a profile, thereby reducing alignment of $k$ sequences to an alignment of $k-1$ sequences/profiles. Repeat
- This is a heuristic greedy method

\[
\begin{align*}
&u_1 = \text{ACGTACGTACGT…} \\
&u_2 = \text{TTAATTAATTA…} \\
&u_3 = \text{ACTACTACTACT…} \\
&\vdots \\
&u_k = \text{CCGGCCGGCCGG…} \\
\end{align*}
\]
Greedy Approach: Example

- Consider these 4 sequences

  $s_1$ GATTCA  
  $s_2$ GTCTGA  
  $s_3$ GATATT  
  $s_4$ GTCAGC
Greedy Approach: Example (cont’d)

- There are $\binom{4}{2} = 6$ possible alignments

\[
\begin{align*}
  s2 & \quad \text{GTCTGA} &  s1 & \quad \text{GATTCA--} \\
  s4 & \quad \text{GTCAGC (score = 2)} &  s4 & \quad \text{G-T-CAGC (score = 0)} \\
  s1 & \quad \text{GAT-TCA} &  s2 & \quad \text{G-TCTGA} \\
  s2 & \quad \text{G-TCTGA (score = 1)} &  s3 & \quad \text{GATAT-T (score = -1)} \\
  s1 & \quad \text{GAT-TCA} &  s3 & \quad \text{GAT-ATT} \\
  s3 & \quad \text{GATAT-T (score = 1)} &  s4 & \quad \text{G-TCAGC (score = -1)}
\end{align*}
\]
Greedy Approach: Example (cont’d)

$s_2$ and $s_4$ are closest; combine:

\[
\begin{array}{c}
  s_2 & GTCTGA \\
  s_4 & GTCAGC \\
\end{array}
\]

new set of 3 sequences:

\[
\begin{array}{c}
  S_1 & GATTCA \\
  S_3 & GATATT \\
  S_{2,4} & GTCT/aGa/c \ (profile) \\
\end{array}
\]
Progressive Alignment

- *Progressive alignment* is a variation of greedy algorithm with a somewhat more intelligent strategy for choosing the order of alignments.
- Progressive alignment works well for close sequences, but deteriorates for distant sequences
  - Gaps in consensus string are permanent
  - Use profiles to compare sequences
ClustalW

- Popular multiple alignment tool today
- ‘W’ stands for ‘weighted’ (different parts of alignment are weighted differently).
- Three-step process
  1.) Construct pairwise alignments
  2.) Build Guide Tree
  3.) Progressive Alignment guided by the tree
Step 1: Pairwise Alignment

- Aligns each sequence against each other giving a similarity matrix
- Similarity = exact matches / sequence length (percent identity)

\[
\begin{array}{cccc}
  \mathbf{v}_1 & \mathbf{v}_2 & \mathbf{v}_3 & \mathbf{v}_4 \\
  \mathbf{v}_1 & - & & \\
  \mathbf{v}_2 & .17 & - & \\
  \mathbf{v}_3 & .87 & .28 & - \\
  \mathbf{v}_4 & .59 & .33 & .62 & - \\
\end{array}
\]

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

- Create Guide Tree using the similarity matrix
  - ClustalW uses the neighbor-joining method
  - Guide tree roughly reflects evolutionary relations
Step 2: Guide Tree (cont’d)

Calculate:

$V_{1,3} = \text{alignment}(V_1, V_3)$

$V_{1,3,4} = \text{alignment}((V_{1,3}, V_4)$

$V_{1,2,3,4} = \text{alignment}((V_{1,3,4}, V_2)$
Step 3: Progressive Alignment

• Start by aligning the two most similar sequences
• Following the guide tree, add in the next sequences, aligning to the existing alignment
• Insert gaps as necessary

<table>
<thead>
<tr>
<th></th>
<th>FOS_RAT</th>
<th>FOS_MOUSE</th>
<th>FOS_CHICK</th>
<th>FOSB_MOUSE</th>
<th>FOSB_HUMAN</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PEEMSVTS-LDLTGGGLPEATTPESEEAFTLPLLNDPEPK-PSLEPVKNISNMELKAEPFD</td>
<td>PEEMSVAS-LDLTGGGLPEASTPESEEAFTLPLLNDPEPK-PSLEPVKSISNVELKAEPFD</td>
<td>SEELAAATALDLG----APSPAAEEAFAPLMTEAPPVPPKEPSG--SGLELKAEPFD</td>
<td>PGPGPLAEVRDLPG----STSAKEDGFGLWLLPPP---LPPPFPQ</td>
<td>PGPGPLAEVRDLPG----SAPAKEDGFSLWLLPPP---LPPPFPQ</td>
</tr>
</tbody>
</table>

Dots and stars show how well-conserved a column is.
Multiple Alignments: Scoring

• Number of matches (multiple longest common subsequence score)

• Entropy score

• Sum of pairs (SP-Score)
Multiple LCS Score

- A column is a “match” if all the letters in the column are the same.

  AAA
  AAA
  AAT
  ATC

- Only good for very similar sequences.
Entropy

- Define frequencies for the occurrence of each letter in each column of multiple alignment
  - $p_A = 1$, $p_T = p_G = p_C = 0$ (1st column)
  - $p_A = 0.75$, $p_T = 0.25$, $p_G = p_C = 0$ (2nd column)
  - $p_A = 0.50$, $p_T = 0.25$, $p_C = 0.25$, $p_G = 0$ (3rd column)
- Compute entropy of each column

\[
- \sum_{X = A, T, G, C} p_X \log p_X
\]
Entropy: Example

\[
\text{entropy} \begin{pmatrix} A \\ A \\ A \\ A \end{pmatrix} = 0 \quad \text{Best case}
\]

\[
\text{entropy} \begin{pmatrix} A \\ T \\ G \\ C \end{pmatrix} = -\sum \frac{1}{4} \log \frac{1}{4} = -4\left(\frac{1}{4}\right) - 2 = 2 \quad \text{Worst case}
\]
Multiple Alignment: Entropy Score

Entropy for a multiple alignment is the sum of entropies of its columns:

$$\sum_{\text{over all columns}} \sum_{\chi=A,T,G,C} p_{\chi} \log p_{\chi}$$
Entropy of an Alignment: Example

**column entropy:**

\[-( p_A \log p_A + p_C \log p_C + p_G \log p_G + p_T \log p_T)\]

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>A</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>C</td>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td>A</td>
<td>C</td>
<td>G</td>
<td>G</td>
</tr>
<tr>
<td>A</td>
<td>C</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>

- Column 1 = \[-[1*\log(1) + 0*\log0 + 0*\log0 + 0*\log0]\]
  = 0

- Column 2 = \[-\left(\frac{1}{4}\right)*\log\left(\frac{1}{4}\right) + \left(\frac{3}{4}\right)*\log\left(\frac{3}{4}\right) + 0*\log0 + 0*\log0\]
  = \[-\left(\frac{1}{4}\right)*(-2) + \left(\frac{3}{4}\right)*(-0.415)\] = +0.811

- Column 3 = \[-\left(\frac{1}{4}\right)*\log\left(\frac{1}{4}\right) + \left(\frac{1}{4}\right)*\log\left(\frac{1}{4}\right) + \left(\frac{1}{4}\right)*\log\left(\frac{1}{4}\right) + \left(\frac{1}{4}\right)*\log\left(\frac{1}{4}\right)\]
  = 4* \[-\left(\frac{1}{4}\right)*(-2)\] = +2.0

- Alignment Entropy = 0 + 0.811 + 2.0 = +2.811
Multiple Alignment Induces Pairwise Alignments

Every multiple alignment induces pairwise alignments

\[
\begin{align*}
  x & : \text{AC--GCGG--C} \\
  y & : \text{AC--GC--GAG} \\
  z & : \text{GCCGC--GAG}
\end{align*}
\]

Induces:

\[
\begin{align*}
  x & : \text{ACGCGG--C} \; \quad x : \text{AC--GCGG--C} \; \quad y : \text{AC--GCGGAG} \\
  y & : \text{ACGC--GAC} \; \quad z : \text{GCCGC--GAG} \; \quad z : \text{GCCGCCGAG}
\end{align*}
\]
Inferring Pairwise Alignments from Multiple Alignments

- From a multiple alignment, we can infer pairwise alignments between all sequences, but they are not necessarily optimal.

- This is like projecting a 3-D multiple alignment path on to a 2-D face of the cube.
Multiple Alignment Projections

A 3-D alignment can be projected onto the 2-D plane to represent an alignment between a pair of sequences.

All 3 Pairwise Projections of the Multiple Alignment
Sum of Pairs Score (SP-Score)

- Consider pairwise alignment of sequences $a_i$ and $a_j$
  imposed by a multiple alignment of $k$ sequences
- Denote the score of this suboptimal (not necessarily optimal) pairwise alignment as $s^*(a_i, a_j)$
- Sum up the pairwise scores for a multiple alignment:
  $$s(a_1, ..., a_k) = \sum_{i,j} s^*(a_i, a_j)$$
Computing SP-Score

Aligning 4 sequences: 6 pairwise alignments

Given \( a_1, a_2, a_3, a_4 \):

\[
s(a_1 \ldots a_4) = \Sigma s^*(a_i, a_j) = s^*(a_1, a_2) + s^*(a_1, a_3)
+ s^*(a_1, a_4) + s^*(a_2, a_3)
+ s^*(a_2, a_4) + s^*(a_3, a_4)
\]
SP-Score: Example

\[ a_1 \text{ ATG–C–AAT} \]
\[ \text{. A–G–CATAT} \]
\[ a_k \text{ ATCCCATTT} \]

To calculate each column:

\[ s(a_1 \ldots a_k) = \sum_{i,j} s^*(a_i, a_j) \left( \begin{array}{c} n \\ 2 \end{array} \right) \text{Pairs of Sequences} \]

\[
\begin{array}{c}
A \\
1 \\
A
\end{array} \quad \begin{array}{c}
G \\
\mu \\
C
\end{array}
\]

Column 1

Score = 3

\[
\begin{array}{c}
G \\
\mu \\
G
\end{array} \quad \begin{array}{c}
C \\
\mu \\
A
\end{array}
\]

Column 3

Score = \(1 - 2\mu\)
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
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

2004 MUSCLE

What’s next?
Problems with Multiple Alignment

- Multidomain proteins evolve not only through point mutations but also through domain duplications and domain recombinations.
- Although MSA is a 30 year old problem, there were no MSA approaches for aligning rearranged sequences (i.e., multi-domain proteins with shuffled domains) prior to 2002.
- Often impossible to align all protein sequences throughout their entire length.
POA vs. Classical Multiple Alignment Approaches

Some content from Chris Lee, POA, UCLA

http://www.bioinformatics.ucla.edu/poa/Poa_Tutorial.html
Alignment as a Graph


B

C

D

• Conventional Alignment
• Protein sequence as a path
• Two paths
• Combined graph (partial order) of both sequences
Solution: Representing Sequences as Paths in a Graph

Each protein sequence is represented by a path. Dashed edges connect “equivalent” positions; vertices with identical labels are fused.
Partial Order Multiple Alignment

Two objectives:
• Find a graph that represents domain structure
• Find mapping of each sequence to this graph

Solution
• PO-MSA (Partial Order Multiple Sequence Alignment) – for a set of sequences $S$ is a graph such that every sequence in $S$ is a path in $G$. 
Partial Order Alignment (POA) Algorithm

Aligns sequences onto a directed acyclic graph (DAG)

1. Guide Tree Construction
2. Progressive Alignment Following Guide Tree
3. Dynamic Programming Algorithm to align two PO-MSAs (PO-PO Alignment).
PO-PO Alignment

- We learned how to align one sequence (path) against another sequence (path).
- We need to develop an algorithm for aligning a directed graph against a directed graph.
Dynamic Programming for Aligning Two Directed Graphs

\[
S(n, o) = \max_{\begin{array}{c} p \rightarrow n \\ q \rightarrow o \end{array}} \begin{cases} 
S(p, q) + s(n, o) \\
S(p, o) + \Delta(n) \\
S(n, q) + \Delta(o), 
\end{cases}
\]

- \(S(n, o)\) – the optimal score
- \(n\): node in \(G\)
- \(o\): node in \(G'\)

**Scoring:**

- *match/mismatch*: aligning two nodes with score \(s(n, o)\)
- *deletion/insertion*:
  - omitting node \(n\) from the alignment with score \(\Delta(n)\)
  - omitting node \(o\) with score \(\Delta(o)\)
Row-Column Alignment

Input Sequences:
1. A
2. C
3. C

Row-column alignment:
1. A  B  GAP
2. C  GAP  D
3. C  B  GAP
POA Advantages

• POA is more flexible: standard methods force sequences to align linearly
• PO-MSA representation handles gaps more naturally and retains (and uses) all information in the MSA (unlike linear profiles)
A-Bruijn Alignment (ABA)

- POA - represents alignment as directed graph; no cycles
- ABA - represents alignment as directed graph that may contain cycles
ABA

(a)

(b)

(c)
ABA vs. POA vs. MSA
Advantages of ABA

• ABA more flexible than POA: allows larger class of evolutionary relationships between aligned sequences
• ABA can align proteins that contain duplications and inversions
• ABA can align proteins with shuffled and/or repeated domain structure
• ABA can align proteins with domains present in multiple copies in some proteins
ABA: multiple alignments of protein

ABA handles:
- Domains not present in all proteins
- Domains present in different orders in different proteins