Sequence Analysis '18: Lecture 3

• Pairwise alignment
• Substitution
• Dynamic Programming algorithm
Exact match scoring matrix

To prepare an alignment, we first consider the score for aligning (associating) any one character of the first sequence with any one character of the second sequence.
The cost of mutation is not a constant

**DNA:** A change in the 3rd base in a codon, and sometimes the first base, sometimes conserves the amino acid. No selective pressure.

**Protein:** A change in amino acids that are in the same chemical class conserve their chemical environment. For example: Lys to Arg is conservative because both are positively charged.
The shape and chemical nature of these 20 side chains account for the folding and function of proteins.
Say their full names as fast as possible.

<table>
<thead>
<tr>
<th>M</th>
<th>E</th>
<th>L</th>
<th>I</th>
<th>O</th>
<th>V</th>
<th>N</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>P</td>
<td>K</td>
<td>W</td>
<td>R</td>
<td>H</td>
<td>C</td>
<td>S</td>
</tr>
<tr>
<td>Y</td>
<td>T</td>
<td>G</td>
<td>K</td>
<td>N</td>
<td>W</td>
<td>L</td>
<td>I</td>
</tr>
<tr>
<td>A</td>
<td>M</td>
<td>E</td>
<td>Q</td>
<td>D</td>
<td>N</td>
<td>H</td>
<td>K</td>
</tr>
<tr>
<td>R</td>
<td>V</td>
<td>Y</td>
<td>Q</td>
<td>K</td>
<td>F</td>
<td>H</td>
<td>F</td>
</tr>
<tr>
<td>W</td>
<td>Y</td>
<td>H</td>
<td>F</td>
<td>N</td>
<td>Q</td>
<td>T</td>
<td>S</td>
</tr>
<tr>
<td>A</td>
<td>P</td>
<td>G</td>
<td>L</td>
<td>C</td>
<td>S</td>
<td>N</td>
<td>G</td>
</tr>
<tr>
<td>H</td>
<td>Q</td>
<td>V</td>
<td>I</td>
<td>C</td>
<td>K</td>
<td>R</td>
<td>M</td>
</tr>
</tbody>
</table>
Willie Taylor’s AA Venn Diagram
Conservative amino acid changes means chemically similar side chain

Lys $\leftrightarrow$ Arg

Ile $\leftrightarrow$ Leu

Ser $\leftrightarrow$ Thr  Asp $\leftrightarrow$ Glu  Asn $\leftrightarrow$ Gln

If the “chemistry” of the sidechain is conserved, then the mutation is less likely to change structure/function.
Amino acid substitution matrices

Two 20x20 substitution matrices are used: BLOSUM & PAM.

|    | A | C | D | E | F | G | H | I | K | L | M | N | P | Q | R | S | T | V | W | Y |
| 4  | 0 | -2| -1| -2| 0 | -2| -1| -1| -1| -1| -2| -1| -1| 1 | 0 | 0 | -3| -2|   |   |   |
| 9  | -3| -4| -2| -3| -3| -1| -1| -3| -1| -3| -3| -3| -3| -1| -1| -1| -2| -2|   |   |   |
| 6  | 2 | -3| -1| -1| -3| -1| -4| -3| 1 | -1| 0 | -2| 0 | -1| -3| -4| -3|   |   |   |   |
| 5  | -3| -2| 0 | -3| 1 | -3| -2| 0 | -1| 2 | 0 | 0 | -1| -2| -3| -2|   |   |   |   |
| 6  | -3| -1| 0 | -3| 0 | 0 | -3| -4| -3| -3| -2| -2| -1| 1 | 3 |   |   |   |   |   |
| 6  | -2| -4| -2| -4| -3| 0 | -2| -2| -2| 0 | -2| -3| -2| -3|   |   |   |   |   |
| 8  | -3| -1| -3| -2| 1 | -2| 0 | 0 | -1| -2| -3| -2| 2 |   |   |   |   |   |
| 4  | -3| 2 | 1 | -3| -3| -3| -3| -2| -1| 3 | -3| -1|   |   |   |   |   |
| 5  | -2| -1| 0 | -1| 1 | 2 | 0 | -1| -2| -3| -2|   |   |   |   |   |
| 4  | 2 | -3| -3| -2| -2| -1| 1 | -2| -1|   |   |   |   |   |   |   |   |   |
| 5  | -2| -2| 0 | -1| -1| -1| 1 | -1| -1|   |   |   |   |   |   |   |   |   |
| 6  | -2| 0 | 0 | 1 | 0 | -3| -4| -2|   |   |   |   |   |   |   |   |   |
| 7  | -1| -2| -1| -1| -2| -4| -3|   |   |   |   |   |   |   |   |   |
| 5  | 1 | 0 | -1| -2| -2| -1|   |   |   |   |   |   |   |   |   |
| 5  | -1| -1| -3| -3| -2|   |   |   |   |   |   |   |   |   |   |
| 4  | 1 | -2| -3| -2|   |   |   |   |   |   |   |   |   |   |
| 5  | 0 | -2| -2|   |   |   |   |   |   |   |   |   |   |
| 4  | -3| -1|   |   |   |   |   |   |   |   |   |   |   |
| 11 | 2 |   |   |   |   |   |   |   |   |   |   |   |   |
| 7  |   |   |   |   |   |   |   |   |   |   |   |   |   |

Each number is the score for aligning a single pair of amino acids.

Calculate the score for this alignment:

ACEPGAA
ASDDGTV

BLOSUM62
For protein alignments, first set up the scoring matrix by filling in the appropriate substitution score.

Inexact match using BLOSUM62 substitution matrix:

score = 10
Pseudo code for setting up the alignment matrix

read blosum[1..20][1..20]
read firstseq[1..N]
for (i from 1 to N) firstseq[i] = location(aa,firstseq[i])
read secondseq[1..M]
for (i from 1 to M) secondseq[i] = location(aa,secondseq[i])
alignmentmatrix[1..N][1..M]=0
for (i from 1 to N) do
    for (j from 1 to M) do
        alignmentmatrix[i][j] = blosum[firstseq[i]][secondseq[j]]
    enddo
enddo
write alignmentmatrix[1..N][1..M]
An Alignment as a Path through the Alignment Matrix

Imagine each of these boxes has a score in it. (i.e. from BLOSUM)

There are directions for each step: down, right, or diagonal.

The alignment is complete when we reach the lower right-hand corner box.

Imagine each of these boxes has a score in it. (i.e. from BLOSUM)
arrows

Upper seq advances by one, Lower seq advances by zero. **Gap** in lower seq.

Upper seq advances by zero, Lower seq advances by one. **Gap** in upper seq.

Upper seq advances by one, Lower seq advances by one. **Match**.
definitions

Gap -- an insertion or deletion of any length

gap character -- biologically meaningless spacer used when printing an alignment

Insertion -- location in aligned sequences where a sequence is longer by one or more characters

Deletion -- location in aligned sequences where a sequence is longer by one or more characters
The size of the alignment matrix is \((N+1)x(M+1)\), where \(N\) and \(M\) are the lengths of the two sequences.

Rows on top and left ("gap" rows) have no scores.
An alignment

\[
\begin{array}{ccc}
A & B & C \\
D & E & F \\
\end{array}
\]

\[
\begin{array}{c}
A B \sim C \\
\sim D E F \\
\end{array}
\]
Scoring an alignment with gaps

Start with score = 0
For each step:
  if gap: add a gap penalty
  if match: add $A_{ij}$ value.
program scoremyalignment

    right=1; down=2; diag=3;
    Aseq=0; Bseq=0; gappenalty=2;
    read myfile, A[0..3,0..3]
    walk[1..4]=(right,diag,down,diag)
    score = 0.
    for i from 1 to len(walk) do
        if (walk[i]==right) then
            score = score + gappenalty
            Aseq = Aseq + 1
        elseif (walk[i]==down) then
            score = score + gappenalty
            Bseq = Bseq + 1
        elseif (walk[i]==diag) then
            Aseq = Aseq + 1
            Bseq = Bseq + 1
            score = score + A[Aseq,Bseq]
        endif
    enddo
    write *, "Score =",score
end program scoremyalignment

right, down, diag are arbitrary constants in this case
Aseq, Bseq are counters
myfile contains precalculated alignment scores
walk is a series of arrows

add a penalty for gaps

add the alignment score to the score
only if the arrow was diag
A walk through the alignment matrix
All possible alignments for ABC versus DEF

| ABC~~~ | AB~C      | ABC~~~BC         |
| ~~~~DEF| ~DEF      | DEF~~~           |
| ABC~~~ | A~BC      | A~BC             |
| ~~~DEF | ~DEF      | DEF              |
| ABC~   | A~BC      | ABC~~           |
| ~~~DEF | DE~F      | ~D~EF            |
| ABC    | A~BC      | A~B~C           |
| DEF    | DEF~      | ~DEF~            |
| ~ABC   | AB~C      | A~BC~BC         |
| DEF~   | ~DEF      | ~DEF~            |
| ~~ABC  | AB~C      | A~BC~BC         |
| DEF~~~ | DEF~      | ~DEF~            |

no gaps          one gaps                  two gaps
All possible alignments = all possible paths

starting here

ending here
Each box spawns three paths.

Total number of possible paths of $n$ arrows = $3^n$
The reasoning behind “dynamic programming”

All "paths" through the alignment matrix end in the lower right-hand corner. Imagine there are $3^n$ “walkers” each taking a different path to the lower right box and collecting scores and penalties as it goes. We can ask each walker as it enters the last box what its final score is. Then we pick the walker with the highest score and ask it what path it took. **Optimal? Yes. Efficient? No.** (We needed $3^n$ walkers!)

Consider the step before the last box (e.g. a down arrow). All walkers that traversed that arrow added the same amount, in this case a gap penalty. If it was a diagonal arrow, they would still all add the same amount, a match score. So the highest scoring walker before the arrow is still the highest scoring after the arrow. We could have ignored all of the walkers with sub-optimal scores.

The same argument applies to all boxes back to the beginning. We only need to keep the optimal path. We can ignore the others, since they will always rank lower than the optimal.

**Optimal? Yes. Efficient? Yes.** (Only 3 calculations per box.)
Dynamic Programming algorithm

• For each box, add up the scores of the three walks that end in that box. **Keep just the highest scoring one.**

• Draw an arrow (traceback) from the box that had the highest scoring walk.

• When you reach the last box, trace back along the arrows.

• Convert the arrows into an alignment. This is the optimal alignment.
Dynamic programming algorithm

For all $i=0..i_{\text{max}}$ { For all $j=0..j_{\text{max}}$ {
    
    $S_{i,j} = \text{MAX}\{ S_{i-1,j-1} + s(i,j), \ S_{i-1,j} - w_x, \ S_{i,j-1} - w_y \}$
    
    $T_{i,j} = \text{argmax}\{ S_{i-1,j-1} + s(i,j), \ S_{i-1,j} - w_x, \ S_{i,j-1} - w_y \}$

    
} }

Traceback($T$)

$S(i,j)$ is the score assigned to box $(i,j)$, composed to the maximum of the three values, the alignment score at $(i-1,j-1)$ plus the match score $s(i,j)$, or the alignment score at $(i-1,j)$ minus the gap penalty $w_x$, or the alignment score at $(i,j-1)$ minus the gap penalty $w_y$. In practice $w_x = w_y$.

$T(i,j)$ is a flag pointing back along the arrow.
Forward summation

\[ S_{i,j} = \text{MAX} \{ S_{i-1,j-1} + s(i,j), S_{i-1,j} - w_x, S_{i,j-1} - w_y \} \]

The first row and column (gap rows) are filled in using only one of the arrows, since the other two are out-of-bounds.
Forward summation

\[ S_{i,j} = \text{MAX} \{ S_{i-1,j-1} + s(i,j), \]
\[ S_{i-1,j} - w_x, \]
\[ S_{i,j-1} - w_y \} \]

Think of each arrow as adding a new term, either \( s(i,j) \) or a gap penalty.

Filling in a box requires that the values in the three input boxes be already filled in.
Forward summation

\[ S_{i,j} = \text{MAX} \{ S_{i-1,j-1} + s(i,j), S_{i-1,j} - w_x, S_{i,j-1} - w_y \} \]

The last row and column are filled in normally
Traceback

We save one traceback value in each box. This is a number, letter, or word that represents the arrow direction: i.e. down, right, or diagonal.

\[ T_{i,j} = \arg\max \{ S_{i-1,j-1} + s(i,j), S_{i-1,j} - w_x, S_{i,j-1} - w_y \} \]

\* \argmax() is a function that returns the number of the maximum argument, not the value.
Traceback

Traceback starts from the **last box** where \( i = \) length of \( x \), and \( j = \) length of \( y \).

Each "arrow" points back to the previous box. The result is a series of arrows in reverse order:

\[
\begin{align*}
2 & 1 3 1
\end{align*}
\]

These are reversed: 2131

...and translated to an alignment:

\[
\begin{align*}
2 & 1 3 1 \\
& \downarrow \downarrow \\
 \sim & AB \sim C \\
& \sim \sim \sim \sim DEF
\end{align*}
\]

\[
T_{i,j} = \text{argmax} \{ S_{i-1,j-1} + s(i,j), \quad S_{i-1,j} - w_x, \quad S_{i,j-1} - w_y \}
\]
From arrows to alignment.

Remember that each arrow is one "column" of the alignment.

- Upper seq advances by one, Lower seq advances by zero. **Gap** in lower seq.
- Upper seq advances by zero, Lower seq advances by one. **Gap** in upper seq.
- Upper seq advances by one, Lower seq advances by one. **Match**.
From arrows to alignment.

Remember that each arrow is one "column" of the alignment.

Align ADGTFR with ADTFRE using the following arrows:

A D G T F R ~
A D ~ T F R E
Exercise 3: dynamic programming with proteins

Match = use BLOSUM score in lower right  Gap penalty = $-1$

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>D</th>
<th>G</th>
<th>T</th>
<th>F</th>
<th>R</th>
<th>M</th>
<th>G</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>-2</td>
<td>6</td>
<td>-1</td>
<td>-1</td>
<td>-3</td>
<td>-2</td>
<td>-3</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>G</td>
<td>0</td>
<td>-1</td>
<td>6</td>
<td>-2</td>
<td>-3</td>
<td>-2</td>
<td>-3</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>Y</td>
<td>-2</td>
<td>-3</td>
<td>-3</td>
<td>-2</td>
<td>3</td>
<td>-2</td>
<td>-1</td>
<td>-3</td>
<td>-3</td>
</tr>
<tr>
<td>R</td>
<td>-1</td>
<td>-2</td>
<td>-2</td>
<td>-1</td>
<td>-3</td>
<td>5</td>
<td>-1</td>
<td>-2</td>
<td>-2</td>
</tr>
<tr>
<td>I</td>
<td>-2</td>
<td>-3</td>
<td>-4</td>
<td>-1</td>
<td>0</td>
<td>-3</td>
<td>1</td>
<td>-4</td>
<td>-4</td>
</tr>
<tr>
<td>G</td>
<td>6</td>
<td>-1</td>
<td>6</td>
<td>-2</td>
<td>-3</td>
<td>-2</td>
<td>-3</td>
<td>6</td>
<td>6</td>
</tr>
</tbody>
</table>
Exercise 3, part 1. Instructions

• Prefill the boxes with the appropriate BLOSUM score. 
  *(done. numbers in lower right.)*

• Calculate $S_{i,j} = \max \{ S_{i-1,j-1} + s(i,j), S_{i-1,j} - 1, S_{i,j-1} - 1 \}$. **Ignore arrows that are out-of-bounds.** Fill in boxes from upper left to lower right, as you read.

• As you fill in each $S_{i,j}$ score, also draw one **arrow**(argmax \{ $S_{i-1,j-1} + s(i,j), S_{i-1,j} - 1, S_{i,j-1} - 1$ \}) coming into the box, where arrow(1) = diagonal, arrow(2)=right, arrow(3)=down.

• At the lower right, traceback and draw the arrows.

• Translate the arrows into an alignment.
Exercise 3, part 2: Alignment in Ugene

Type the two short sequences into Ugene using File/New Document from text

in Project panel: Select the file names. Mouse-over-Right-click over one of the selected names and a menu pops up. Select Export/Export sequences as alignment. Give it a name. The sequences appear in an alignment, but the sequences are not aligned.

Mouse-over-Right-click on the alignment. Select Align/Align with Kalign.

Set all gap penalties to 1. Bonus to 0.0

Print or type the alignment on paper.

Do you get the same answer as you did in Part 1? If not, fix your mistakes.
Review

• How is the cost of mutation related to evolutionary time?
• Which amino acids are hydrophobic? Hydrophilic? Charged?
• Explain the relationship between a deletion and an insertion?
• In alignment, what is a column?
• What is a path through an alignment matrix?
• How is the number of arrows in a path related to the number of columns in an alignment?
• What does optimal mean?
• Is the optimal alignment the best alignment?