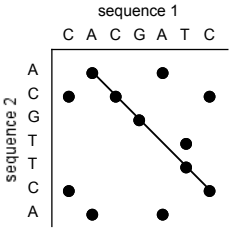
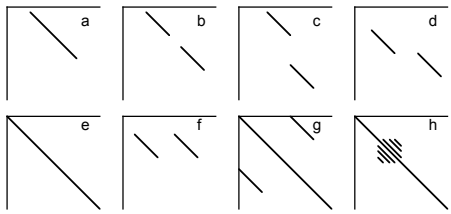
MCB182 Class Notes: Sequence Alignment

Background: Have you ever wondered how a spelling checker works? How does it know if the word is misspelled, and how does it suggest correct spellings? This area of computer science is called inexact (approximate, fuzzy) **string matching**. The bioinformatics equivalent is called sequence alignment. In bioinformatics, we often treat nucleotides and proteins as strings of letters. Even though we know that biological sequences are 3-dimensional entities with physical and chemical properties, it's much easier to process them as 1-dimensional strings. Sequence alignment is used for a variety of bioinformatics tasks. Previously, we have taken for granted that we can assemble a genome and identify repeats. There are also many other tasks that require sequence comparisons such as determining the function of newly discovered proteins, gene finding, constructing phylogenetic trees, and designing oligos. So how do we *know* if two sequences are similar? There are two fundamental concepts: (1) creating alignments (2) determining if alignments are significant. We will first discuss how to create alignments and then next time we will consider their significance.



Dot plots: A simple way to look at the relationship between two sequences is a dot plot (or dot matrix). This is a 2D matrix with a sequence along each axis. Each point in the matrix corresponds to a specific letter in each sequence. Regions of similarity appear as diagonals in the matrix. Rather than draw dots, it easier to draw lines showing just the similar regions. In the 8 graphs shown: (a) an alignment showing a regional similarity between two sequences (b) a section in the middle does not align as well (c, d) the similar regions are separated by a gap (e) a sequence aligned to itself (f) sequence 1 has a duplication (g) a sequence with a repeat aligned to itself (h) a sequence with an SSR aligned to itself. Note that in (c, d) there is either an insertion in one sequence or a deletion in the other. Gaps are therefore often called **indels**.



Pairwise alignment: There are two "flavors" of pairwise alignment: **global** and **local**. In global alignment, the goal is to align every letter of the two sequences. Consider aligning the letters in these two sequences: (1) ACTTTGA (2) TTT. One possible alignment between these is shown as "align 1". Every letter in each sequence is either aligned to another letter or a gap (-) symbol. If the sequences are identical, it is typical to use a | character between the sequences to indicate this. Another common convention is to use the letter. Whenever sequences have unequal lengths, there will be gaps. The gaps can occur anywhere. For example, an alignment between (1) and another sequence (3) ACTGA is shown as "align 2". In local alignment, only the best region is kept. "Align 3" shows two possible local alignments of sequences (1) and (3). Both are equally good. There are also a large number of really poor alignments one could make.

Align 1
```
ACTTTGA
  |||
--TTT--
```

Align 2
```
ACTTTGA
|| | ||
AC-T-GA
```

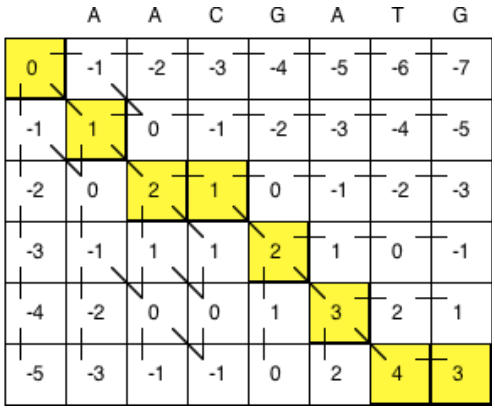
Align 3
```
ACT     TGA
||| or |||
ACT     TGA
```

Alignment scoring: In order to compare alignments to each other, we can give them a **score**. A simple scoring scheme is to give every matching letter a score of +1 and every mismatch or gap a score of -1. Under such a scheme, the scores for alignments 1-3 are: -1, 3, and 3.

Needleman-Wunsch algorithm: To find the best global alignment one uses the N-W algorithm (or some variant of it). The number of possible alignments between two sequences is huge. You can put gaps in either sequence anywhere you like (but not across from each other). A naive alignment algorithm would enumerate all possible gaps and then choose the alignment with the best score. Even with short sequences this quickly becomes unwieldy and in biological sequences, which can be huge in the case of chromosomes, the number of alignments becomes astronomical. N-W uses dynamic programming (DP) to efficiently find a single highest

DP is a class of optimization algorithms where the optimal solution is built up from a large number of sub-optimal solutions. This is not a class on algorithms, so don't get bogged down in the details.

scoring alignment. There may be more than one alignment with the maximum score, but the algorithm usually only returns one of these. To begin the N-W algorithm, the sequences are entered into a matrix (like a dot plot) with an extra 1st column and row. There are 3 steps to the algorithm: (1) initialization (2) fill (3) trace back. In the initialization, the first row and column are set to gap scores. In the fill, a recursive operation is used to update the maximum score of every cell. In the trace back, the alignment is recovered by following the maximum alignment from the bottom right of the matrix through the top left.



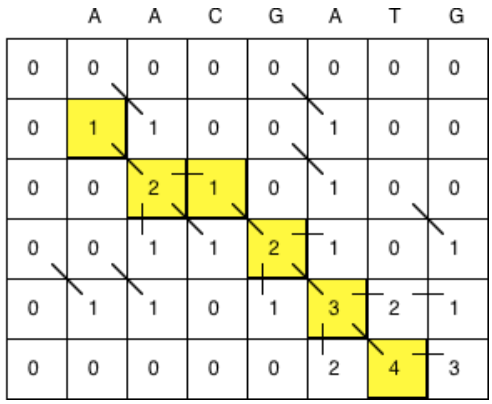
Let's take a close look at the fill. In order to fill a cell, you must have 3 neighboring cells located above, to the left, and diagonally above and left. At the beginning, there is only one cell that can be filled. This is the one that aligns the first A and A in the example. To fill this cell, you must determine the maximum score of 3 possible directions (diagonal, up, and left).

Diagonal score = score of diagonal cell + match or mismatch score (either +1 or -1)
Left score = score of left cell + gap score (-1)
Up score = score of up cell + gap score (-1)

When you move horizontally or vertically, you do not consider wether the nucleotides match or not because this operation introduces a gap character.

The power of DP is that we do this same operation of looking at 3 possible alignments at every position in the matrix. But we are not enumerating all possible alignments, we are always extending the previous maximum alignment.

Smith-Waterman algorithm: To find the maximum scoring local alignment, you can use the exact same procedure as N-W except that (a) any score below 0 is given the score of 0. At the end, the trace back is performed from the highest score in the matrix rather than the last cell of the matrix.



Scoring system: Different match, mismatch, and gap scores will result in different alignments. Try the same sequences with a gap score of -2 and you will get a slightly different alignment.

Computational considerations: The N-W and S-W algorithms as described are not used for aligning long sequences. One reason is that the amount of memory to hold the DP matrix becomes excessive. Each cell in the matrix must hold a score and a directional pointer. This might be 5 bytes of RAM per cell. In order to align two BACs of 100 kb each, you would need about 50 GB of RAM (1e5 x 1e5 x 5). What if you wanted to align some genomes? No computer on the planet has enough RAM. Another reason not to use N-W and S-W is that most of the the space in a DP matrix has a low score. Why align everything rather than just the best parts? Sequence alignment is one of the oldest areas of bioinformatics research, but it is still very active. There are a lot of clever programs that perform alignments very quickly without using much memory. At the root of all these programs is some variant of the S-W algorithm.