Chapter 6 Class Notes – Sequence Alignment

From Chap. 5 (p.122): human Hsa6 and 6 mouse genes:

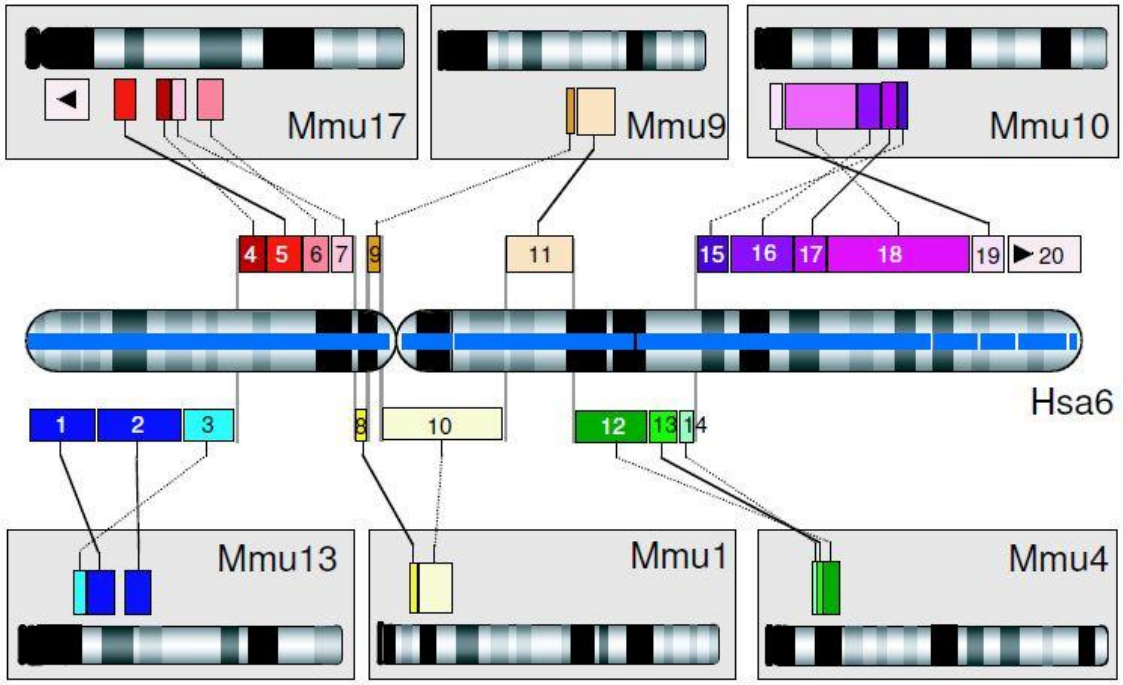


Fig. 5.1. [This figure also appears in the color insert.] Syntenic blocks conserved between human chromosome Hsa6 and mouse chromosomes. Broken lines indicate regions that appear in inverted orders in the two organisms. Reprinted, with permission, from Gregory SG et al. (2002) *Nature* 418:743–750. Copyright 2002 Nature Publishing Group.

p.121: "Clearly, segments of DNA in the two species were rearranged with respect to each other during their independent descent from a common ancestor."

6.1. The Biological Problem: our focus is on shared characters among organisms – to (1) establish evolutionary relationships among organisms, (2) identify functionally conserved sequences (DNA sequences controlling gene expression), (3) identify corresponding genes in model organisms which can be genetically manipulated to develop models for human diseases. Genes/characters in organisms B and C that have evolved from the same ancestral gene/character in A are homologs; thus, "we

might expect that mice would have homologs of human genes for immunoglobulins but would not expect such genes to occur in bacteria." Also, homology may → similarity, but not necessarily vice versa (similarity: degree of match at corresponding positions two DNA/protein sequences, usually expressed as a percentage).

By writing a sequence of gene g_A and of each candidate homolog as strings of characters, with one string above the other, we can determine at which positions the strings do or do not match – this is called an alignment.

Example: which of the following matches is "better" or "best"?

(6.1) ACGTCTAG 2 matches, 5 mismatches, 1 not aligned
ACTCTAG–

(6.2) ACGTCTAG 5 matches, 2 mismatches, 1 not aligned
–ACTCTAG

(6.3) ACGTCTAG 7 matches, 0 mismatches, 1 not aligned
AC–TCTAG

So as to produce global alignments and local alignments, using biological "common sense", we'll assign 'scores' for each of:

- Substitution (point mutation)
- Insertion of short segments --}
- Deletion of short segments ---} Together, called "indels"
- Segmental duplication
- Inversion
- Transposable element insertion
- Translocation

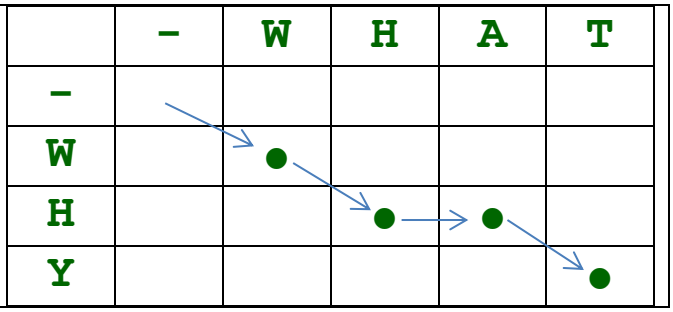
First 3 items above are the most important for present purposes. As noted, insertions and deletions together are called indels; in (6.3) above, we don't know if top strand resulted from insertion of G into ACTCTAG or if the original ancestral sequence was ACGTCTAG and there was a deletion in the second sequence.

6.2. Basic (Motivating) Example: global alignment of "WHAT" and "WHY" using scoring: identity (match) gives $+1$, substitution (mismatch) gives $-\mu$, and indel gives $-\delta$

Result:

WHAT
WH-Y

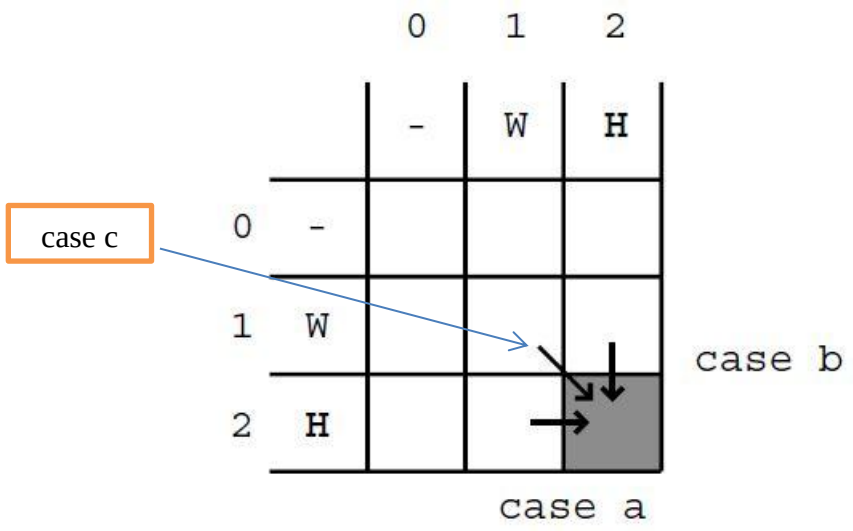
Final score: $1 + 1 - \delta - \mu = 2 - \delta - \mu$



In the above table, we have included one additional row and one additional column for initial indels (−) to allow for the possibility (not applicable here) that alignments do not start at the initial letters (W opposite W in this case). The alignment at left is represented as the series of ● and arrows → through the table at right. "What we seek is the path through the matrix [table] that produces the greatest possible score in the element at the lower right-hand corner." This process of using up all the letters in the search string (first column) and search space (first row) is the process of global alignment.

Dynamic programming: to find the "best" driving route from LA to Chicago, we can break down the trip to a series of choices among cities in segments of the trip. For example, in the last segment, record the times coming in the Chicago from City 1 (Minneapolis; t_1), City 2 (Des Moines; t_2), or City 3 (St. Louis; t_3), and then in to each of these cities from sets of other cities, and so on all the way back to LA. We record this process into an alignment matrix as above at right. "The best alignment is revealed by beginning at the destination (lower right-hand corner matrix element) and working backward, identifying the path that maximizes the score at the end." That said, we do start the process of scoring at the NW corner of the alignment matrix.

To illustrate, suppose the incremental score for matches are $+1$, mismatches are -1, and for indels are -2. Then, consider the following portion of the alignment matrix for the above 'toy' example (top row or search space is from `WHAT`, and left-most column or search string is from `WHY`):



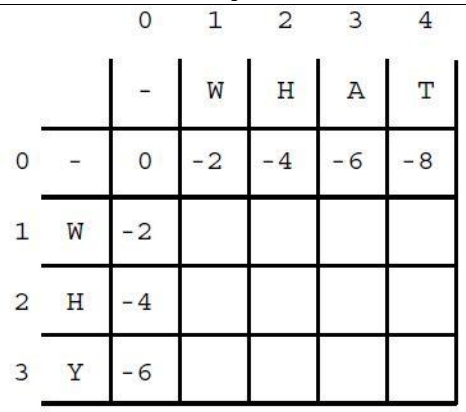
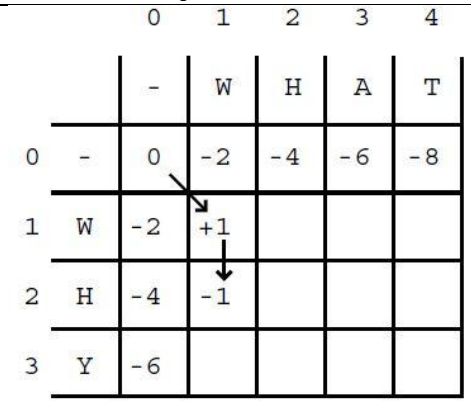
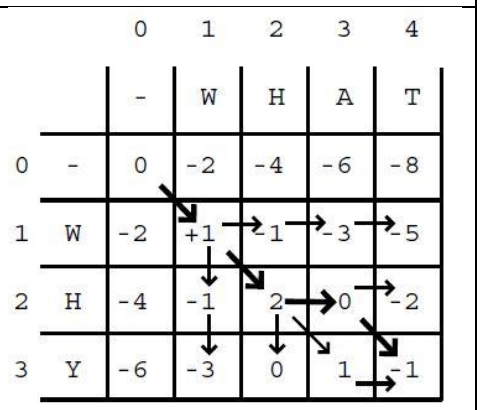
"case a" corresponds to (W) H [AT]
(WH) - [Y]
and the score would be $S_{22} = S_{21} - 2$ (the cost of an indel is -2)

"case b" corresponds to (WH) - [AT]
(W) H [Y]
and the score would be $S_{22} = S_{12} - 2$ (the cost of an indel is -2)

"case c" corresponds to (W) H [AT]
(W) H [Y]
and the score would be $S_{22} = S_{11} + 1$ (increment for match is +1)

Of the above 3 choices for S_{22}, we choose the highest one (i.e., the maximum) and the path associated with it. Suppose this corresponds to S_{11}: we then repeat this process noting the [three] paths that lead into S_{11}, and so on.

Using the same scoring as above, we now return to the full problem and application of this global alignment scoring method:

Initial Set-up	Next Steps	Final Product
		

"The" final answer is that we get two equally-maximal alignments (each with final scores of -1):

```
WHAT        and        WHAT
WH-Y                   WHY-
```

The worst alignments (each with a score of -14) would be

```
WHAT---        or        ---WHAT
----WHY                  WHY----
```

Now, you try: using the scoring $s(a_i, b_j) = +1\ if\ a_i = b_j, s(a_i, b_j) = -1\ if\ a_i \neq b_j, s(a_i, -) = s(-, b_j) = -2$, globally align sequences (search string) $A =$ ATCGT with (search space) $B =$ TGGTG. (The answer is on next page.)

6.3. Global Alignment – Formal Development: we start with sequences from the same alphabet $A = a_1a_2a_3 \ldots a_n$ and $B = b_1b_2b_3 \ldots b_m$. Next, we set up the initial matrix using (6.11) $S_{i,0} = -i\delta, S_{0,j} = -j\delta$. Incremental scores in individual cases are given as follows:

$$s(a_i, b_j) = \textit{the score of aligning } a_i \textit{ with } b_j$$
$$= \begin{cases} 1, & \textit{if } a_i = b_j \\ -\mu, & \textit{if } a_i \neq b_j \end{cases}$$
$$s(a_i, -) = s(-, b_j) = -\delta \ (\textit{for indels})$$

Then use to find the running total score:

$$S_{i,j} = \max \begin{cases} S_{i-1,j-1} + s(a_i, b_j) \\ S_{i-1,j} - \delta \\ S_{i,j-1} - \delta \end{cases}$$

Computational Example 6.1: with the scoring as above, globally align $A =$ ATCGT with $B =$ TGGTG

To start:

		0	1	2	3	4	5
		-	T	G	G	T	G
0	-	0	-2	-4	-6	-8	-10
1	A	-2					
2	T	-4					
3	C	-6					
4	G	-8					
5	T	-10					

Final Answer:

		0	1	2	3	4	5
		-	T	G	G	T	G
0	-	0	-2	-4	-6	-8	-10
1	A	-2	-1	-3	-5	-7	-9
2	T	-4	-1	-2	-4	-4	-6
3	C	-6	-3	-2	-3	-5	-5
4	G	-8	-5	-2	-1	-3	-4
5	T	-10	-7	-4	-3	0	-2

Final Answer:

```
A:  A T C G T -
B:  - T G G T G
```

Final Score: -2

6.4. Local Alignment – Rationale and Formulation: we need an alignment of segments *within* two strings, so we traverse only part of the alignment matrix. With local alignments, we do not apply indel penalties if strings A and B fail to align at the ends – so, all elements in the first row and first column will now be zero. Also, there is no need to continue paths whose scores become too small; thus if the best path to an element from its immediate neighbors above and to the left (including the diagonal) leads to a negative score, we will assign a 0 score to that element. We identify the best local alignment by tracing back from the matrix element having the highest score; this is usually not (but may be) the element in the lower right-hand corner of the matrix.

We again start with sequences from the same alphabet $A = a_1a_2a_3 \ldots a_n$ and $B = b_1b_2b_3 \ldots b_m$: within each of these there are intervals $I \subset A$ and $J \subset B$. The best local alignment score, $M(A, B)$, for strings is

$$M(A, B) = \max\{S(I, J) : I \subset A, J \subset B\}$$

Here, $S(I, J)$ is the score for subsequences I and J. Elements of the alignment matrix are $M_{i,j}$, and as noted $M_{i,0} = M_{0,j} = 0$. Also,

$$M_{i,j} = \max \begin{cases} M_{i-1,j-1} + s(a_i, b_j) \\ M_{i-1,j} - \delta \\ M_{i,j-1} - \delta \\ 0 \end{cases}$$

Thus, the best local alignment score for strings A and B is $M(A, B) = \max_{i,j} M_{i,j}$.

Computational Example 6.2 – Local alignment: find the best local alignment score for $A =$ ACCTAAGG and $B =$ GGCTCAATCA, and using the scoring $s(a_i, b_j) = 2$ if $a_i = b_j, s(a_i, b_j) = -1$ if $a_i \neq b_j$, and $s(a_i, -) = s(-, b_j) = -2$.

		0	1	2	3	4	5	6	7	8	9	10
		-	G	G	C	T	C	A	A	T	C	A
0	-	0	0	0	0	0	0	0	0	0	0	0
1	A	0	0	0	0	0	0	2	2	0	0	2
2	C	0	0	0	2	0	2	0	1	1	2	0
3	C	0	0	0	2	1	2	1	0	0	3	1
4	T	0	0	0	0	4	2	1	0	2	1	2
5	A	0	0	0	0	2	3	4	3	2	1	3
6	A	0	0	0	0	0	1	5	6	4	2	3
7	G	0	2	2	0	0	0	3	4	5	3	1
8	G	0	2	4	2	0	0	1	2	3	4	2

1

The resulting local alignment is enclosed in the box below:

```
A:    A C C T - A A G G -
B:    G G C T C A A T C A
```

Most local alignment programs only report the aligned regions of A and B, that is, the sequences shown in the box above.

We will only report the answer in the box above.

6.5. Number of Possible Global Alignments: We start with sequences from the same alphabet $A = a_1a_2a_3 \ldots a_n$ and $B = b_1b_2b_3 \ldots b_m$, and ask: how many possible global alignments are there for these two strings? The number of alignments, $\#A$, is the sum of the number of alignments having $1, 2 \ \ldots \ \min\ \{m, n\}$ matched pairs. To count the number of ways of having k aligned pairs, we must choose k letters from each sequence. From A this can be done in $\binom{n}{k}$ ways, and from B this can be done in $\binom{m}{k}$ ways. Thus, we get Equation (6.18):

$$\#A = \sum_{k=0}^{\min(n,m)} \binom{n}{k}\binom{m}{k} = \binom{n+m}{\min(n,m)}$$

To illustrate, if $A = a_1a_2a_3a_4$ and $B = b_1b_2b_3$, $n + m = 7$, $\min\ (n, m) = 3$, so $\#A = \binom{7}{3} = \frac{7\times6\times5}{3\times2\times1} = 35$. If, instead, $n = m = 100$ then $\#A > 9 \times 10^{58}$, so we'll seek more efficient methods to do the alignments (see next chapter).

In R, "choose(200,100)" gives 9.054851e+58

To prove Equation (6.18), we introduce:

The Hyper-Geometric (HG) Distribution: sample *without replacement* from a binary population. The population size is N. Of these m are type A (successes) and $(N - m)$ are type B, and we take a sample of size h; the probability of drawing k successes is

$$g(k) = \frac{\binom{m}{k}\binom{N-m}{h-k}}{\binom{N}{h}}$$

Here, $k = 0, 1 \ldots \min\ \{m, h\}$. This distribution will be of interest to us in its own right: let $K \sim HG(h, p)$ for $p = \frac{m}{N}$, $E(K) = hp$, $Var(K) = \frac{N-h}{N-1} hp(1-p)$. Note similarities to the Binomial dist.

Now, back to proving Equation (6.18): let $N = n + m$ and $h = n$, then

$$g(k) = \frac{\binom{m}{k}\binom{n}{n-k}}{\binom{m+n}{n}} = \frac{\binom{m}{k}\binom{n}{k}}{\binom{m+n}{n}}$$

Equation (6.18) is proven by noting that this is a valid probability mass function, and so it must sum to one.

6.6. Scoring Rules: for $s(a_i, b_j)$, instead of using the scoring matrix below at left, it makes more sense to use the one below at right:

		b_j:			
		A	C	G	T
a_i	A	1	-1	-1	-1
	C	-1	1	-1	-1
	G	-1	-1	1	-1
	T	-1	-1	-1	1

		b_j:			
		A	C	G	T
a_i	A	1	-1	-½	-1
	C	-1	1	-1	-½
	G	-½	-1	1	-1
	T	-1	-½	-1	1

This follows since transition mutations (within purines or pyrimidines; A→G, G→A, C→T, T→C) occur about twice as often as do tranversions (A→T, A→C, G→C, etc.); transversions require exchanges of one- and two-ring structures. Thus, the "cost" here for transitions is less (-½) and for transversions is twice as much (-1).

A second issue is that indels are probably not independent – meaning that previously we scored a gap of length k as $w(k) = -k\delta$. A more reasonable score for $\alpha > \beta$ might be $w(k) = -\alpha - \beta(k-1)$.

6.7. Multiple Alignment: the simple technique on p.162 really only works for a small number of sequences. The algorithm and calculation is sped up by employing "pairwise alignments in an incremental fashion: the most similar pair is placed into a fixed alignment, and then the other sequences are included in a stepwise fashion." CRUSTALW[1] computes all pairwise alignments, and using trees or clusters (Chapter 10). A final approach is to use Hidden Markov models (HMMs).

6.8. Implement: as noted, the global and local alignment algorithms we have discussed can be very time intensive for 'real' problems, and in the next chapter, we'll discuss less accurate but much faster alignment methods.

[1] Thompson, J.D., Higgins, D.G. and Gibson, T.J., 1994, CRUSTAL W: Improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice, Nucleic Acids Research, 22: 4673-4680.