LSM2241
Fundamentals of Sequence Comparison II

Greg Tucker-Kellogg

26 August 2015

Outline

Topic

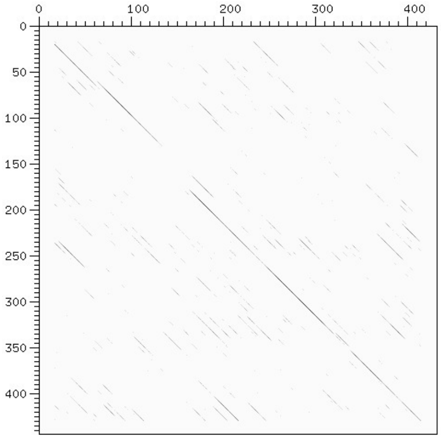
Last week we learned

1. We use pairwise sequence alignment to examine the relationship between two sequences
2. Changes that occur in evolution include substitutions (point mutations) and gaps (insertions and deletions)
3. *Substitution matrices* can be used to score mismatches at each position
4. *log-odds* values can score observed alignments against an expectation of chance
5. Positive score values indicate an alignment is *more likely* to be from homology than by random chance

Topic

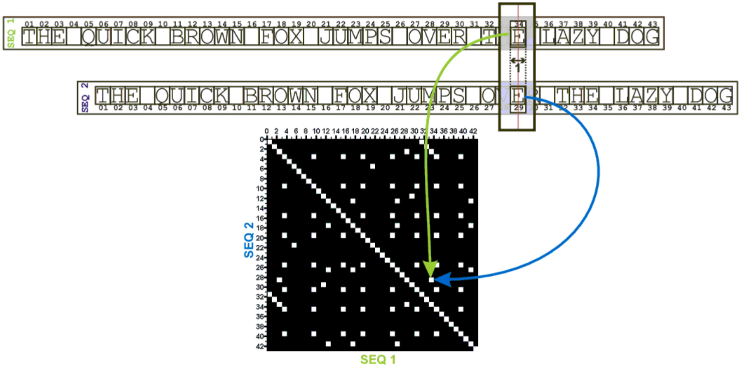
Dotplots: a graphical overview of similarity

- Similarity is plotted for a window across both sequences, without attempt to find best alignment.
- Overall similarity is evident from the main diagonal line
- Local similarity is evident from the diagonal lines elsewhere



A dotplot of two globin protein sequences

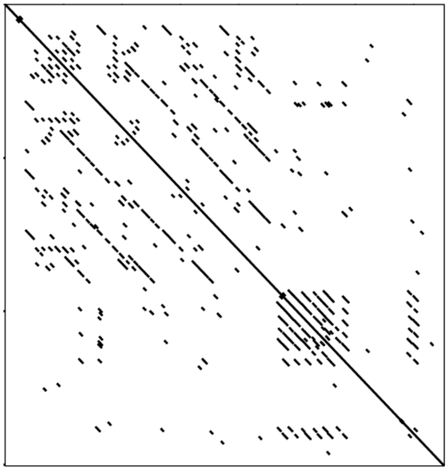
What dotplots are doing


SEQ 1
THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG
1
SEQ 2
THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG
SEQ 2
SEQ 1

In *this plot*, each dot represents an identical letter at the corresponding position. Comparing against the self always gives a main diagonal of dots, which is the alignment of a sequence with itself.

What dotplots might be doing

- using sliding windows instead of individual positions
- scoring windows by similarity instead of identity
- Representing similarity using grays or range of colours
- using BLOSUM or PAM scoring matrices

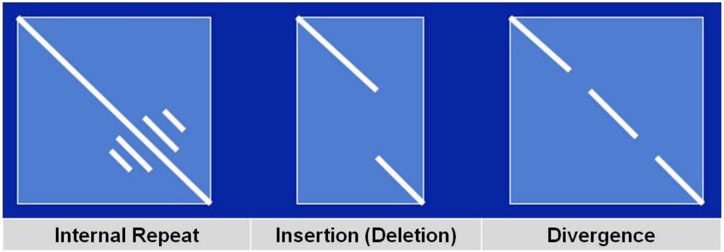


P23014.2 Drosophila melanogaster SLIT protein aligned against itself (BLOSUM62, window size 10, threshold 23)

What dotplots are NOT doing

- Finding optimal alignments
- trying to be clever about gaps

Different dotplot patterns represent different things



- Dotplots give an informative picture of patterns of sequence similarity, even without an optimal alignment
- Even when you have an optimal alignment, dotplots can tell you what you have missed from it

Topic

Alignments are scored by adding every position

- Consider the example alignment at right
- Let's score every match (identity) position as +1, every mismatch as -1 and ignore all gaps.
- This would give an alignment score of 4

```
ATGGCGT
||| *||
ATG-AGT

A T G G C G T
| | |   * | |
A T G - A G T
1+1+1  -1+1+1 = 4
```

What's the substitution matrix?

What we just did corresponds to a substitution matrix of $\rightarrow$

	A	C	G	T
A	**1**	-1	-1	-1
C	-1	**1**	-1	-1
G	-1	-1	**1**	-1
T	-1	-1	-1	**1**

We still need

- A way to score gaps
- A way to identify the optimal alignment

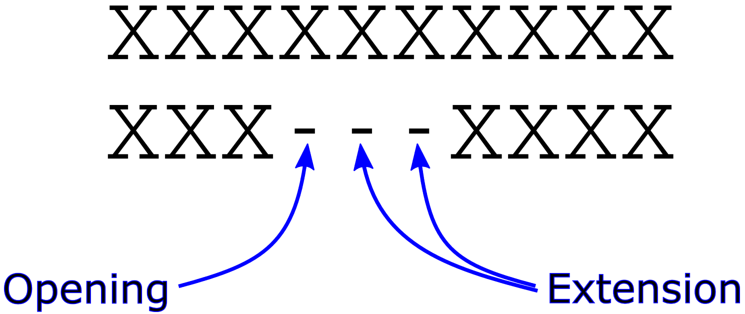
How do we deal with gaps?

The usual way to penalize gaps is one of the following:

1. Penalize with a simple *linear* penalty d for each residue. For a gap of length g, the total penalty would be $-gd$
2. Penalize using an *affine score*, which has one cost d for creating a gap and another cost e for extending it by one residue. The total cost is

$$\gamma(g) = -d - (g-1)e$$

In practice



Generally, *opening* a gap is more costly than extending it

But how do we find the optimal alignment?

- Let's align the following sequences:
  - `SIMILAR`
  - `SIMMARE`
- Let's agree to use a simple scoring system:
  - +5 for any exact match
  - -5 for any mismatch
  - -3 for any gap residue
- Now let's use a *computer* to score every possible alignment and pick the best one!

How many possible alignments are there ?

Suppose we are aligning two sequences of the same length n, the number of alignments is

$$\binom{2n}{n} = \frac{(2n)!}{(n!)^2} \simeq \frac{2^{2n}}{\sqrt{2\pi n}}$$

sequence length	# of possible alignments
8	70
20	184,756
100	1×10^{29}
200	9×10^{58}

This is impossible even for short sequences!

Topic

Global pairwise alignment

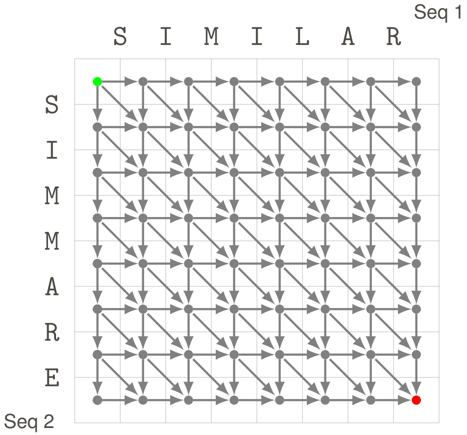
The problem is solved by breaking it into pieces

- Start at one end of both sequences.
- The best alignment score from that end to any position is the best score that can
  1. Get to a position one step earlier
  2. Take that last step

The possibilities seem large

BUT!

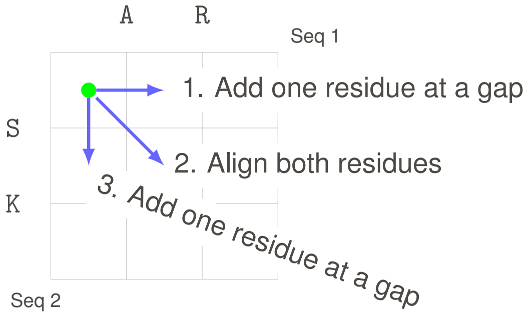
- Since are at most three choices *at* any position
- There are at most three ways *to reach* any position



Lay out the sequences on a grid

- Build an alignment by moving from one cell to the next
- At each step there are at most three possible choices

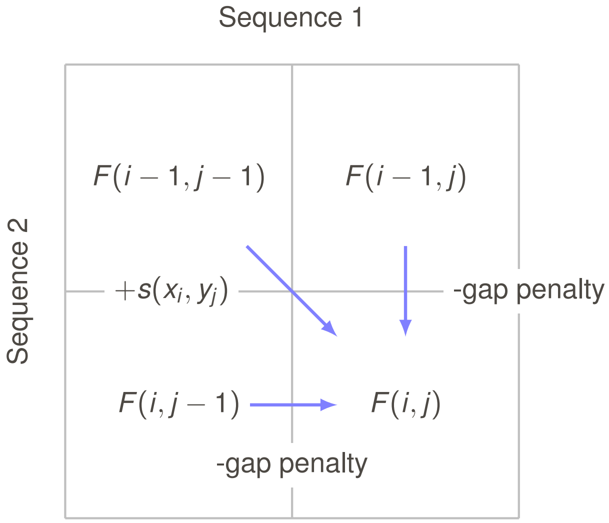
1	Sequence 1	A
	Sequence 2	-
2	Sequence 1	A
	Sequence 2	S
3	Sequence 1	-
	Sequence 2	S



The algorithm of Needleman and Wunsch 1970

1. Create a matrix F of size $n \times m$ for two sequences of length n and m
2. Fill the matrix from one end to the other with the optimal score for alignments up to that point
    - Leave a trail for how you got to each step
3. Trace back through the matrix for the optimal alignment

What is the optimal score at any step?



So the best choice to any point is

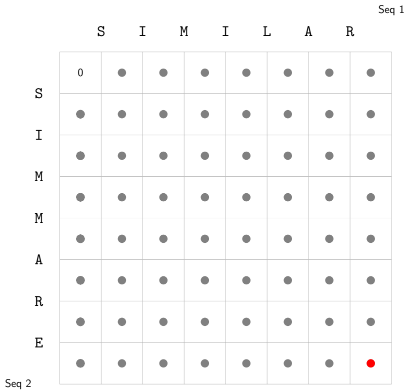
$$F(i,j) = \max \begin{cases} F(i-1, j-1) + s(x_i, y_i) \\ F(i-1, j) - \text{gap penalty} \\ F(i, j-1) - \text{gap penalty} \end{cases}$$

(These are called recurrence relations)

Let’s do it!

Remember

- Start at the upper left, one cell at a time
- Starting score = 0
- Match = +5
- Mismatch = -5
- Gap cost = 3
- *Trace your steps*



The global alignment

It's in there!

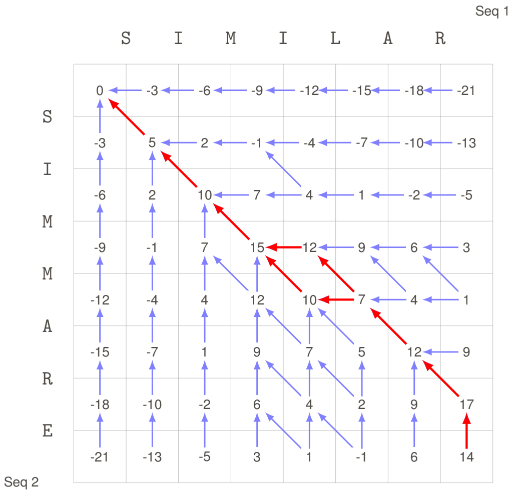
If we start at the end and retrace our steps, we get an alignment that is *optimal* under the scoring scheme

```
search type
Needleman Wunsch

 1 SIMILAR-  7
   ||| |||
 1 SIM-MARE  7

 1 SIMILAR-  7
   |||| ||
 1 SIMM-ARE  7

Alignment score is 14
```



		S	I	M	I	L	A	R
	0	-3	-6	-9	-12	-15	-18	-21
S	-3	5	2	-1	-4	-7	-10	-13
I	-6	2	10	7	4	1	-2	-5
M	-9	-1	7	15	12	9	6	3
M	-12	-4	4	12	10	7	4	1
A	-15	-7	1	9	7	5	12	9
R	-18	-10	-2	6	4	2	9	17
E	-21	-13	-5	3	1	-1	6	14

What about local alignments?

This is the algorithm of Smith and Waterman 1981

The only differences for local alignments are that

1. No cells are allowed to be less than zero, and
2. We start the traceback at the maximum scoring cell of the alignment matrix, rather than at the end
3. We stop the traceback when we reach a cell with a score of 0

The recurrence relations for building the alignment matrix are now:

$$F(i,j) = \max \begin{cases} F(i-1,j-1) + s(x_i, y_i) \\ F(i-1,j) - \text{gap penalty} \\ F(i,j-1) - \text{gap penalty} \\ 0 \end{cases}$$

For *this* example. . . .

It's a little dull

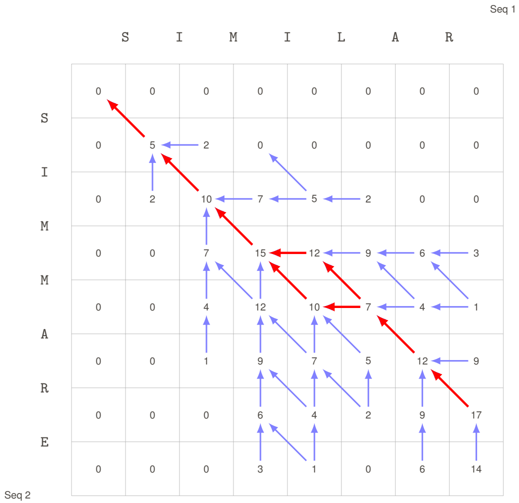
- The best local alignment is a subset of the global alignment
- We lose one residue at the end

```
search type
Smith Waterman

  1 SIMILAR  7
    ||| |||
  1 SIM-MAR  7

  1 SIMILAR  7
    |||| ||
  1 SIMM-AR  7

 Alignment score is 17
```



Dynamic programming

- Our procedures for finding optimal pairwise alignments are *bioinformatics algorithms*
- Needleman-Wunsch (for global alignment) and Smith-Waterman (for local alignment) are *dynamic programming* algorithms.
- In dynamic programming (DP) algorithms like these, we use recursion and/or storage of partial results to get the answer.

Topic

Digression: The Big O

When considering how hard a problem is for a computer, we can describe how the time to solve a problem grows relative to the size of the input. If we call the size n, we say[1]

Complexity	Description	consequences
$O(c)$	constant time	depends on c !
$O(\log(n))$	logarithmic time	big problems are not much harder than small problem
$O(n)$	linear time	Twice the problem is twice the time
$O(n\log(n))$	linearithmic time	Doable
$O(n^2)$	quadratic time	1000 operations = hours or days of time
$O(n^3)$	cubic	problematic
$O(c^n)$	exponential time	intractable

What have we gained from DP for pairwise alignment?

- By using a dynamic programming algorithm, we've turned an impossible problem (of time complexity $O(c^n)$) into a tractable one (of time complexity $O(n^2)$) for global alignment[2],
- But using dynamic programming to align more than two sequences is a problem
    - For three sequences of length n, the problem becomes $O(n^3)$
    - For m sequences of length n, the problem becomes $O(n^m)$

Sequences occur in families, not pairs

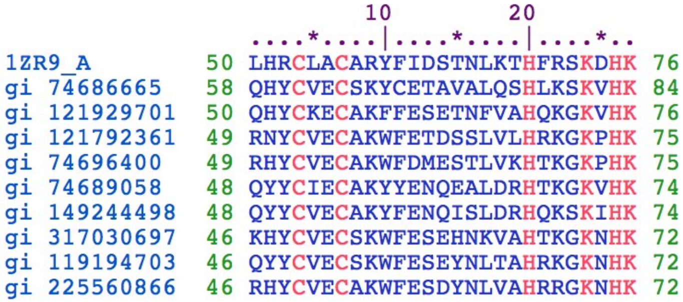
- If we want to understand how sequence relates to function, we need to understand the *patterns* of change.
- Not all positions in a domain of a family behave equally.
- Some can be highly conserved
- Some can be weakly conserved

So we need to look at families

Definition (Multiple Sequence Alignment)

1. An alignment of three or more sequences of DNA, RNA or protein
2. The process for producing such an alignment

An example, a Zinc Finger RNA binding domain[3]

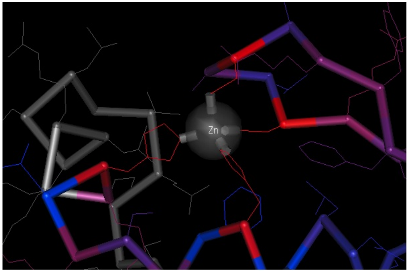


```
                                10        20
                       ....*....|....*....|....*..
1ZR9_A           50 LHRCLACARYFIDSTNLKTHFRSKDHK 76
gi 74686665      58 QHYCVECSKYCETAVALQSHLKSKVHK 84
gi 121929701     50 QHYCKECAKFFESETNFVAHQKGKVHK 76
gi 121792361     49 RNYCVECAKWFETDSSLVLHRKGKPHK 75
gi 74696400      49 RHYCVECAKWFDMESTLVKHTKGKPHK 75
gi 74689058      48 QYYCIECAKYYENQEALDRHTKGKVHK 74
gi 149244498     48 QYYCVECAKYFENQISLDRHQKSKIHK 74
gi 317030697     46 KHYCVECSKWFESEHNKVAHTKGKNHK 72
gi 119194703     46 QYYCVECSKWFESEYNLTAHRKGKNHK 72
gi 225560866     46 RHYCVECAKWFESDYNLVAHRRGKNHK 72
```

Multiple sequence alignment for PFAM12171, invariant positions in red

The conserved residues are key to function

- Each invariant H and C directly binds a zinc atom that characterizes the stucture
- Sequence $\rightarrow$ structure $\rightarrow$ function
- multiple sequence alignment highlights positions likely to be important for function



zinc binding site of $1ZR9\backslash_A$, a C2H2 Type Zinc Finger RNA Binding Protein

Multiple sequence alignment is not exact

- Once we get a lot of sequences, what worked perfectly for pairs of sequences is no longer practical
- We need to use heuristic methods.

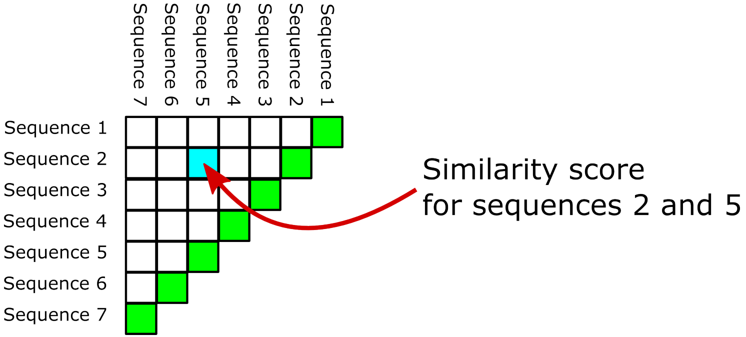
MSA by progressive alignment

For a set of *m* sequences of average length *n*

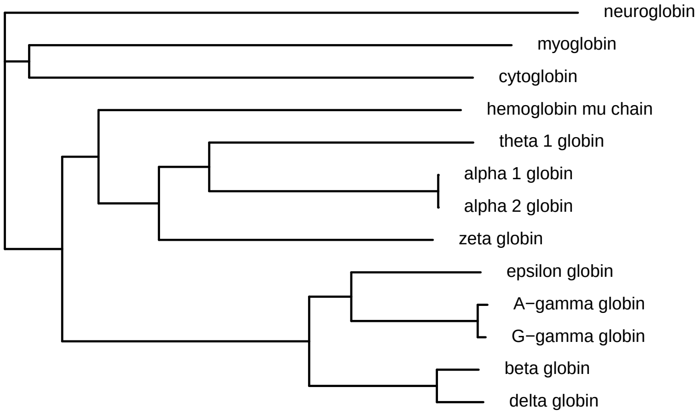
1. Measure pairwise similarity between sequences in the set
2. Use the similarities to arrange the sequences in a *tree*, where neighbors of the tree have high scores
3. Start with the closest pair of sequences, and add one sequence at a time to create a multiple alignment

This is the method used by the Clustal family of programs (**Higgins1996**). The T-Coffee method is related, but uses a slightly different method of constructing the guide tree

Step 1 of MSA: perform pairwise alignment on all pairs of sequences



Step 2 of progressive: arrange the sequences in a *guide tree*



sequences close to each other in the guide tree are similar to each other

Step 3 of progressive MSA: add one sequence at a time based on the guide tree

1. alpha 1 globin + alpha 2 globin
2. A-gamma globin + G-gamma globin
3. beta globin + delta globin
4. (A-gamma globin + G-gamma globin) + espilon globin
5. etc.

Examine your multiple alignment

```
                     80          90
                     .           .
delta_globin         DN...LKGTFSQLSELHCDK     96
beta_globin          DN...LKGTFATLSELHCDK     96
G-gamma_globin       DD...LKGTFAQLSELHCDK     96
A-gamma_globin       DD...LKGTFAQLSELHCDK     96
epsilon_globin       DN...LKPAFAKLSELHCDK     96
alpha_2_globin       DD...MPNALSALSDLHAHK     91
alpha_1_globin       DD...MPNALSALSDLHAHK     91
theta_1_globin       DD...LPHALSALSHLHACQ     91
zeta_globin          DD...IGGALSKLSELHAYI     91
hemoglobin_mu_chain  DN...LRAALSPLADLHALV     90
cytoglobin           HDPDKVSSVLALVGKAHALK    116
myoglobin            GH...HEAEIKPLAQSHATK     97
neuroglobin          EDLSSLEEYLASLGRKHR.A     98
consensus            dd...l...l..LselH..k
```

Risks of progressive alignment

- Any mistakes made early get incorporated into later stages of building the MSA. The early alignments are embedded forever and never change
- *Iterative* methods try to address this by re-evaluating the early alignments to increase an overall score for the MSA
- MUSCLE is a popular such method

Topic

Where we left off

Comparing sequences with dotplots

Optimal pairwise alignment

Pairwise alignment by dynamic programming

Multiple Sequence Alignment

Roundup and next week

What have we learned?

- Knowing already how to score matches and mismatches (last week), we learned how to score gaps
- How to find the optimal global alignment under a scoring system using dynamic programming (Needleman-Wunsch)
- How to find the optimal local alignment (using Smith-Waterman)
- How pairwise alignment methods are exmamples of *dynamic programming* algorithms
- Why DP algorithms cannot be used to solve multiple sequence alignment or database search problems
- How approximate methods can build multiple sequence alignments

Next week: BLAST

- BLAST is used to search sequence databases based on a sequence
- BLAST is probably the most widely used tool in bioinformatics
- We will learn how to use it, how it works, and how to interpret the results of using it.

Bibliography

Needleman, S B and C D Wunsch (1970). “A general method applicable to the search for similarities in the amino acid sequence of two proteins.” In: *Journal of Molecular Biology* 48.3, pp. 443–453 (cit. on p. 23).

Smith, T F and M S Waterman (1981). “Identification of common molecular subsequences.” In: *Journal of Molecular Biology* 147.1. Ed. by Zvi Griliches And Michael D Intriligator, pp. 195–197 (cit. on p. 28).