

# 7.91 – Lecture #2 Michael Yaffe

## More Pairwise Sequence Comparisons

ARDFSHGLLENKLLGCDSMRWE  
GRDYKMALLEQWILGCD-MRWD

- and -

## Multiple Sequence Alignment

ARDFSHGLLENKLLGCDSMRWE  
GRDYKMALLEQWILGCD-MRWD  
SRDW--ALIEDCMV-CNFFRWD

Reading:

This lecture: Mount pp. 8-9, 65-89, 96-115, 140-155, 161-170

# Outline

- Recursion and dynamic programming
- Applied dynamic programming: global alignments: Needleman-Wunsch
- Applied dynamic programming: local alignments – Smith-Waterman
- Substitution matrices: PAM, BLOSUM, Gonnet
- Gaps - linear and affine
- Alignment statistics
- What you need to know to optimize an alignment

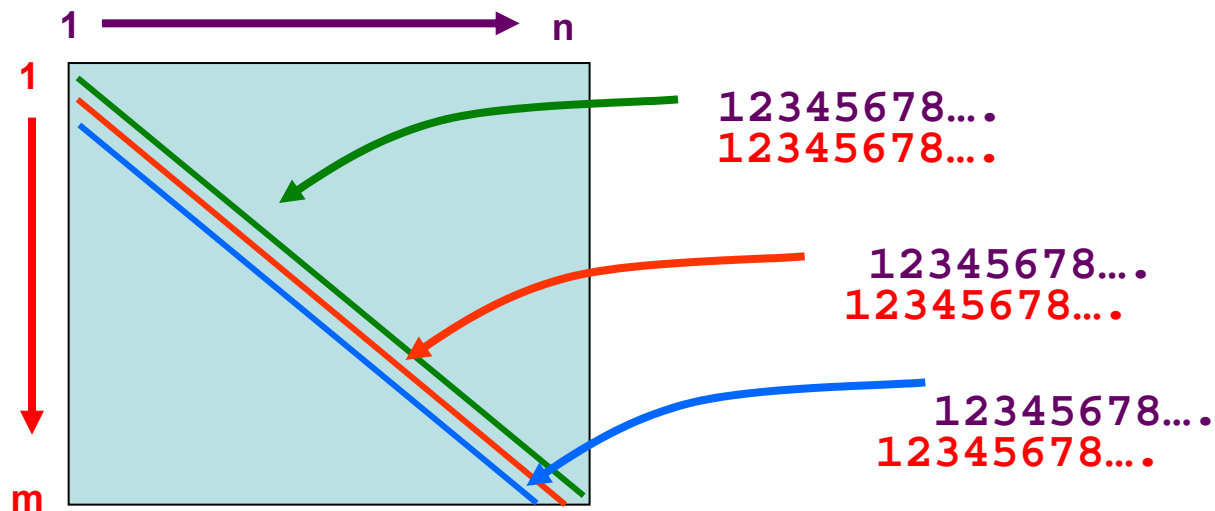
# Outline (cont)

- Multiple sequence alignments: MSA, Clustal
- Block analysis
- Position-Specific Scoring Matrices (PSSM)

## Examples

$O(n^k)$  is “polynomial time” as long as  
 $k \leq 3$  .....tractable

Consider our un-gapped dot matrix  
Global alignment:



.....essentially an  $O(mn)$  problem

## O.K. Examples

**$O(n)$  better than  $O(n \log(n))$ , better than  $O(n^2)$ , better than  $O(n^3)$**

## Terrible Examples

**$O(k^n)$  = exponential time....horrible!!!!**

**NP problems- no known polynomial time  
Solutions = non-deterministic polynomial  
Problems.**

# Recursion and Dynamic Programming

Aligning two protein sequences without gaps – roughly an  $O(mn)$  problem.  
With gaps – becomes computationally astronomical, and cannot be done by direct comparison methods. ( $= 2^{2L}/\sqrt{2\pi L}$ ;  $L$ =sequence length)

Alternative is to compare all possible pairs of characters (matches and mismatches, and also take gaps into account as well, while keeping the number of comparisons manageable. The approach is called dynamic programming. Mathematically proven to produce optimal alignment

Need a substitution or similarity matrix and some way to account for gaps.

Example of how to score an alignment: Write down two sequences:

|                        |   |   |   |     |   |   |
|------------------------|---|---|---|-----|---|---|
| sequence#1             | V | D | S | -   | C | Y |
| sequence#2             | V | E | S | L   | C | Y |
| Score from sub. Matrix | 4 | 2 | 4 | -11 | 9 | 7 |

Score =  $\Sigma(\text{AA pair scores}) - \text{gap penalty} = 15$

# BLOSUM 62 Scoring Matrix

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

**Scoring system should: favor matching identical or related amino acids  
Penalize for poor matches and for gaps.**

**To get a good scoring system need to know: how often a particular amino acid  
Pair is found in related proteins compared with its occurrence by chance. This  
Is the information contained in the substitution matrix  
.....and when a gap would be a better choice**

### **Deriving realistic substitution matrices:**

**First need to know frequency of one amino acid substituting for another  
In related proteins [=P(ab)] c/w the chance that substituting one for the other  
occurred by chance, based on the relative frequencies of each amino acid  
in proteins, q(a) and q(b). Call this the “odds ratio”:  $P(ab)/q(a)q(b)$**

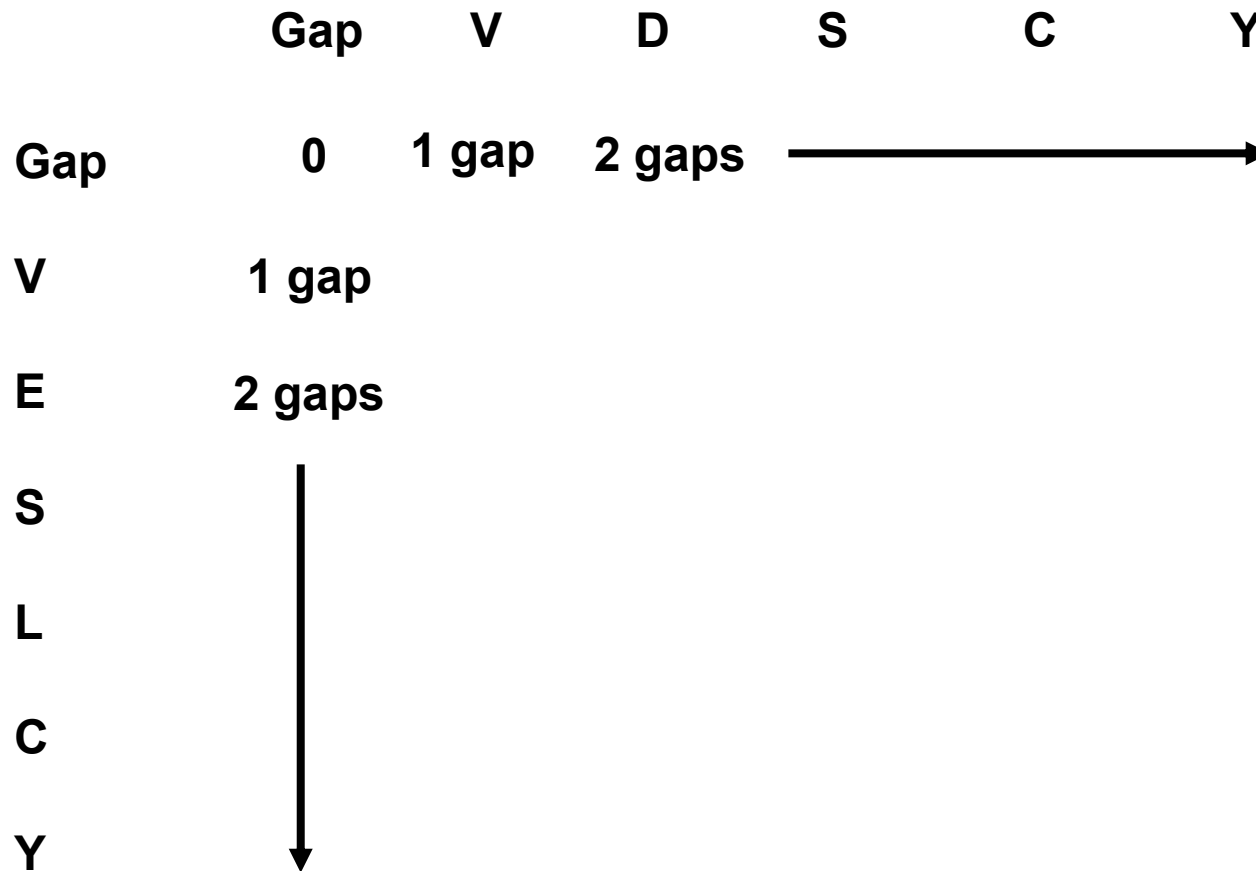
**If we do this for all positions in an alignment, then the total probability will  
be the product of the odds ratios at each position....but multiplication is  
computationally expensive....so....take the **log (odds ratio)** and add them instead.**

**Matrices like PAM and BLOSUM matrices are derived from these log odds ratios  
And contain positive and negative numbers reflecting likelihood of amino  
Acid substitutions in related proteins.**



## To do Dynamic Programming:

First write one sequence across the top, and one down along the side



**Note – linear gap penalty:  $\gamma(n)=nA$ , where  $A$ =gap penalty**

## To do Dynamic Programming:

First write one sequence across the top, and one down along the side

|     |     | i = 0 | 1                     | 2   | 3   | 4   | 5   |
|-----|-----|-------|-----------------------|-----|-----|-----|-----|
| j = |     | Gap   | V                     | D   | S   | C   | Y   |
| 0   | Gap | 0     | -8                    | -16 | -24 | -32 | -40 |
| 1   | V   | -8    | <b>S<sub>ij</sub></b> |     |     |     |     |
| 2   | E   | -16   |                       |     |     |     |     |
| 3   | S   | -24   |                       |     |     |     |     |
| 4   | L   | -32   |                       |     |     |     |     |
| 5   | C   | -40   |                       |     |     |     |     |
| 6   | Y   | -48   |                       |     |     |     |     |

So scoring  $S_{ij}$  requires that we know  $S(i-1, j-1)$  and  $S(i, j-1)$  and  $S(i-1, j)$ ...  
Therefore recursive. We use the solutions of smaller problems to solve larger ones.  
AND we store how we got to the  $S_{ij}$  score, i.e. the intermediate solutions in a tabular matrix. Computer scientists call this dynamic programming, where “programming” means the matrix, not some kind of computer code.

To do Dynamic Programming:

First write one sequence across the top, and one down along the side

|     |     | i = 0 | 1        | 2   | 3   | 4   | 5   |
|-----|-----|-------|----------|-----|-----|-----|-----|
| j = |     | Gap   | V        | D   | S   | C   | Y   |
| 0   | Gap | 0     | -8       | -16 | -24 | -32 | -40 |
| 1   | V   | -8    | $S_{ij}$ |     |     |     |     |
| 2   | E   | -16   |          |     |     |     |     |
| 3   | S   | -24   |          |     |     |     |     |
| 4   | L   | -32   |          |     |     |     |     |
| 5   | C   | -40   |          |     |     |     |     |
| 6   | Y   | -48   |          |     |     |     |     |

Global alignments: Needleman-Wunsch-Sellers  
 $O(n^2)$  using linear gap penalty

$$S_{ij} = \max \text{ of: } \begin{cases} S_{i-1, j-1} + \sigma(x_i, y_j) \text{ (diagonal)} \\ S_{i-1, j} - A \text{ (from left to right)} \\ S_{i, j-1} - A \text{ (from top to bottom)} \end{cases}$$

To do Dynamic Programming:

First write one sequence across the top, and one down along the side

|     |     | i = 0 | 1        | 2   | 3   | 4   | 5   |
|-----|-----|-------|----------|-----|-----|-----|-----|
| j = | Gap |       | V        | D   | S   | C   | Y   |
| 0   | Gap | 0     | -8       | -16 | -24 | -32 | -40 |
| 1   | V   | -8    | $S_{ij}$ |     |     |     |     |
| 2   | E   | -16   |          |     |     |     |     |
| 3   | S   | -24   |          |     |     |     |     |
| 4   | L   | -32   |          |     |     |     |     |
| 5   | C   | -40   |          |     |     |     |     |
| 6   | Y   | -48   |          |     |     |     |     |

Global alignments: Needleman-Wunsch-Sellers  
 $O(n^2)$  using linear gap penalty

$$S_{ij} = \max \text{ of: } \begin{cases} S_{i-1, j-1} + \sigma(x_i, y_j) \text{ (diagonal)} \\ S_{i-1, j} - A \text{ (from left to right)} \\ S_{i, j-1} - A \text{ (from top to bottom)} \end{cases}$$

To do Dynamic Programming:

First write one sequence across the top, and one down along the side

|            |     | <b>i = 0</b> | <b>1</b> | <b>2</b> | <b>3</b> | <b>4</b> | <b>5</b> |
|------------|-----|--------------|----------|----------|----------|----------|----------|
| <b>j =</b> | Gap |              | V        | D        | S        | C        | Y        |
| <b>0</b>   | Gap | 0            | -8       | -16      | -24      | -32      | -40      |
| <b>1</b>   | V   | -8           | <b>4</b> |          |          |          |          |
| <b>2</b>   | E   | -16          |          |          |          |          |          |
| <b>3</b>   | S   | -24          |          |          |          |          |          |
| <b>4</b>   | L   | -32          |          |          |          |          |          |
| <b>5</b>   | C   | -40          |          |          |          |          |          |
| <b>6</b>   | Y   | -48          |          |          |          |          |          |

To do Dynamic Programming:

First write one sequence across the top, and one down along the side

|     |     | i = 0 | 1  | 2   | 3   | 4   | 5   |
|-----|-----|-------|----|-----|-----|-----|-----|
| j = |     | Gap   | V  | D   | S   | C   | Y   |
| 0   | Gap | 0     | -8 | -16 | -24 | -32 | -40 |
| 1   | V   | -8    | 4  | -3  | -8  |     |     |
| 2   | E   | -16   |    |     |     |     |     |
| 3   | S   | -24   |    |     |     |     |     |
| 4   | L   | -32   |    |     |     |     |     |
| 5   | C   | -40   |    |     |     |     |     |
| 6   | Y   | -48   |    |     |     |     |     |

Global alignments: Needleman-Wunsch-Sellers  
 $O(n^2)$  using linear gap penalty

$$S_{ij} = \max \text{ of: } \left\{ \begin{array}{l} S_{i-1, j-1} + \sigma(x_i, y_j) \text{ (diagonal)} \\ S_{i-1, j} - A \text{ (from left to right)} \\ S_{i, j-1} - A \text{ (from top to bottom)} \end{array} \right.$$

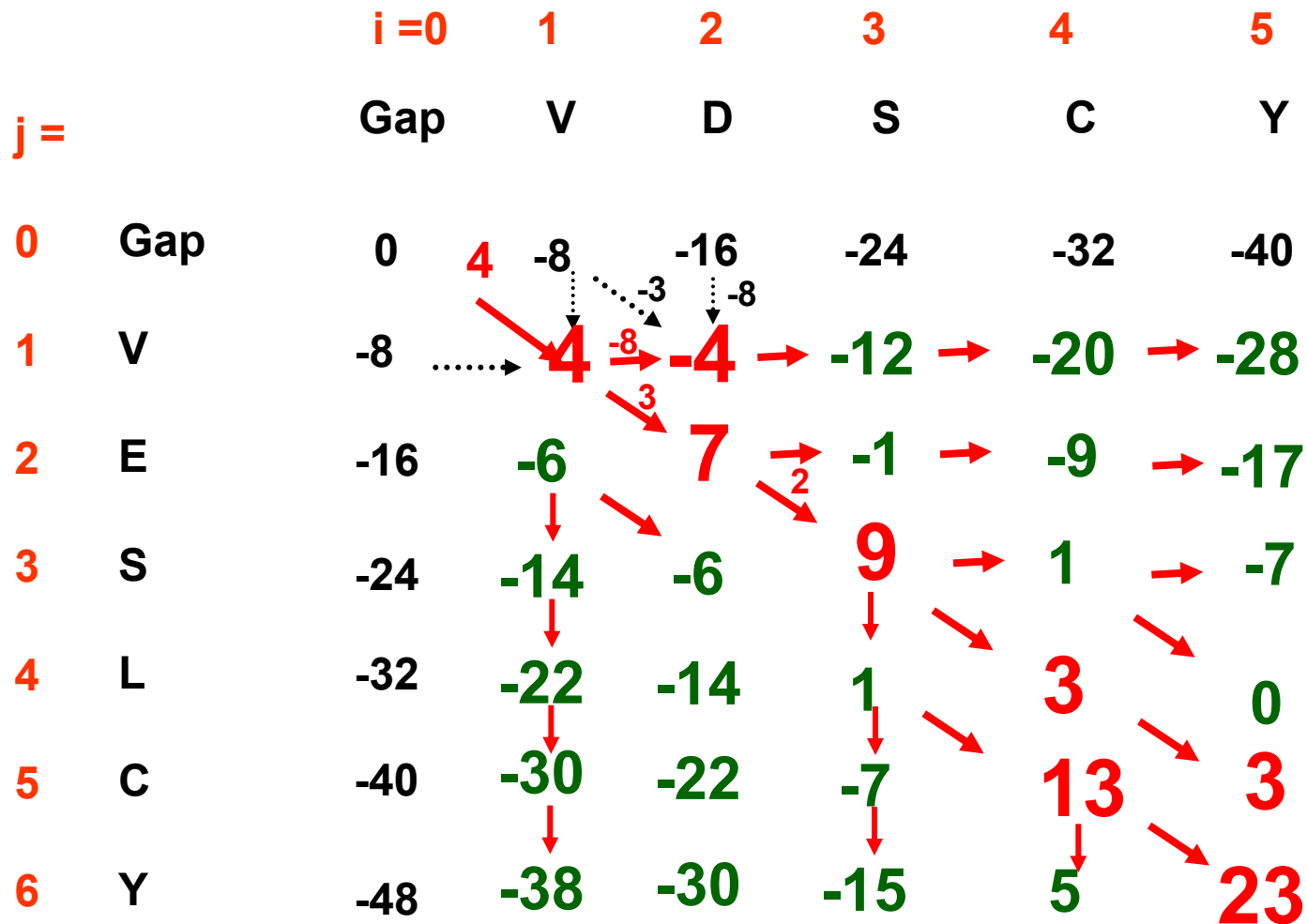
## To do Dynamic Programming:

First write one sequence across the top, and one down along the side

|            |     | <b>i = 0</b> | <b>1</b> | <b>2</b> | <b>3</b> | <b>4</b> | <b>5</b> |     |
|------------|-----|--------------|----------|----------|----------|----------|----------|-----|
| <b>j =</b> |     | Gap          | V        | D        | S        | C        | Y        |     |
| <b>0</b>   | Gap | 0            | 4        | -8       | -16      | -24      | -32      | -40 |
| <b>1</b>   | V   | -8           | 4        | -4       |          |          |          |     |
| <b>2</b>   | E   |              |          |          |          |          |          |     |
| <b>3</b>   | S   |              |          |          |          |          |          |     |
| <b>4</b>   | L   |              |          |          |          |          |          |     |
| <b>5</b>   | C   |              |          |          |          |          |          |     |
| <b>6</b>   | Y   |              |          |          |          |          |          |     |

To do Dynamic Programming:

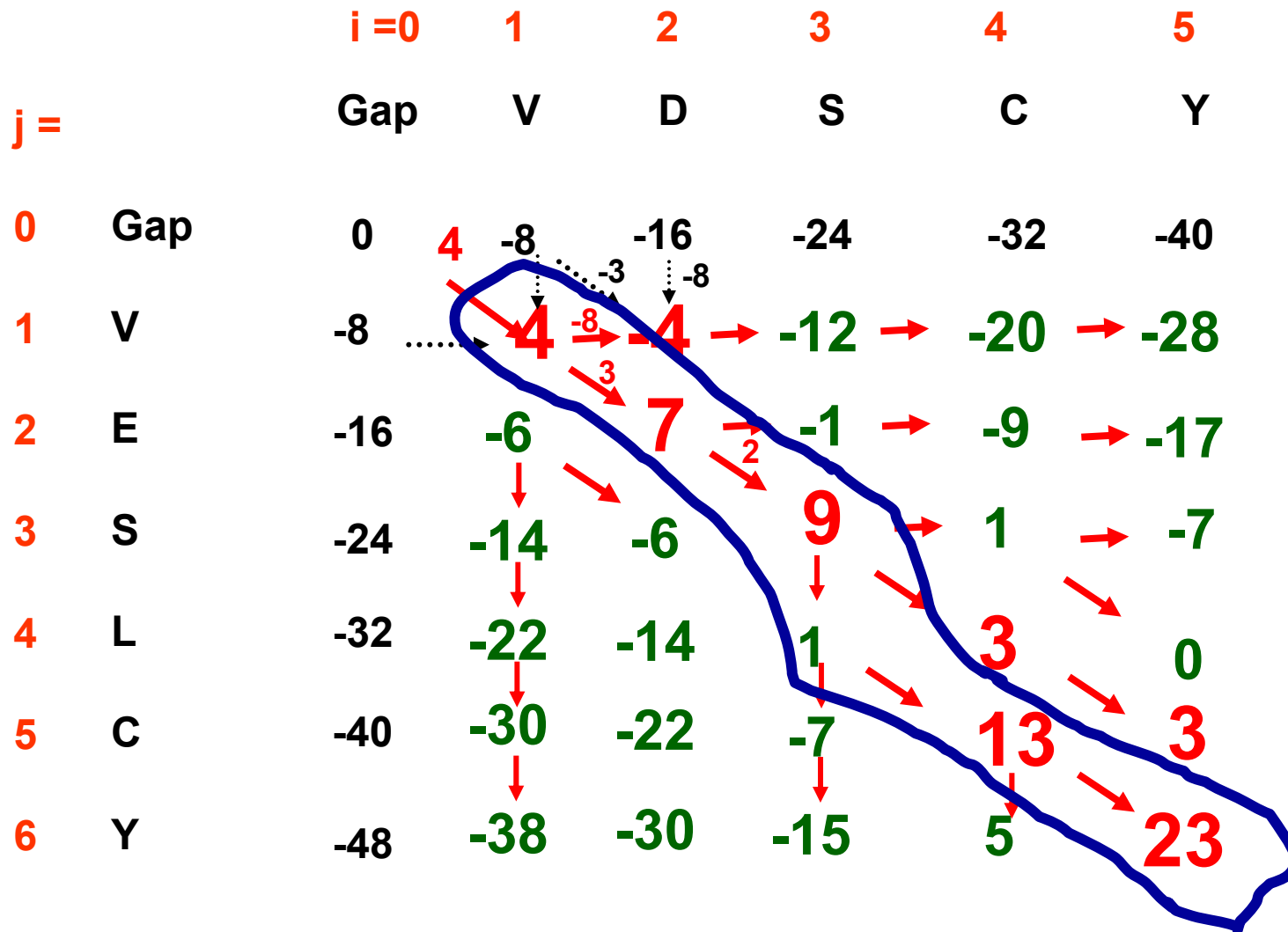
First write one sequence across the top, and one down along the side





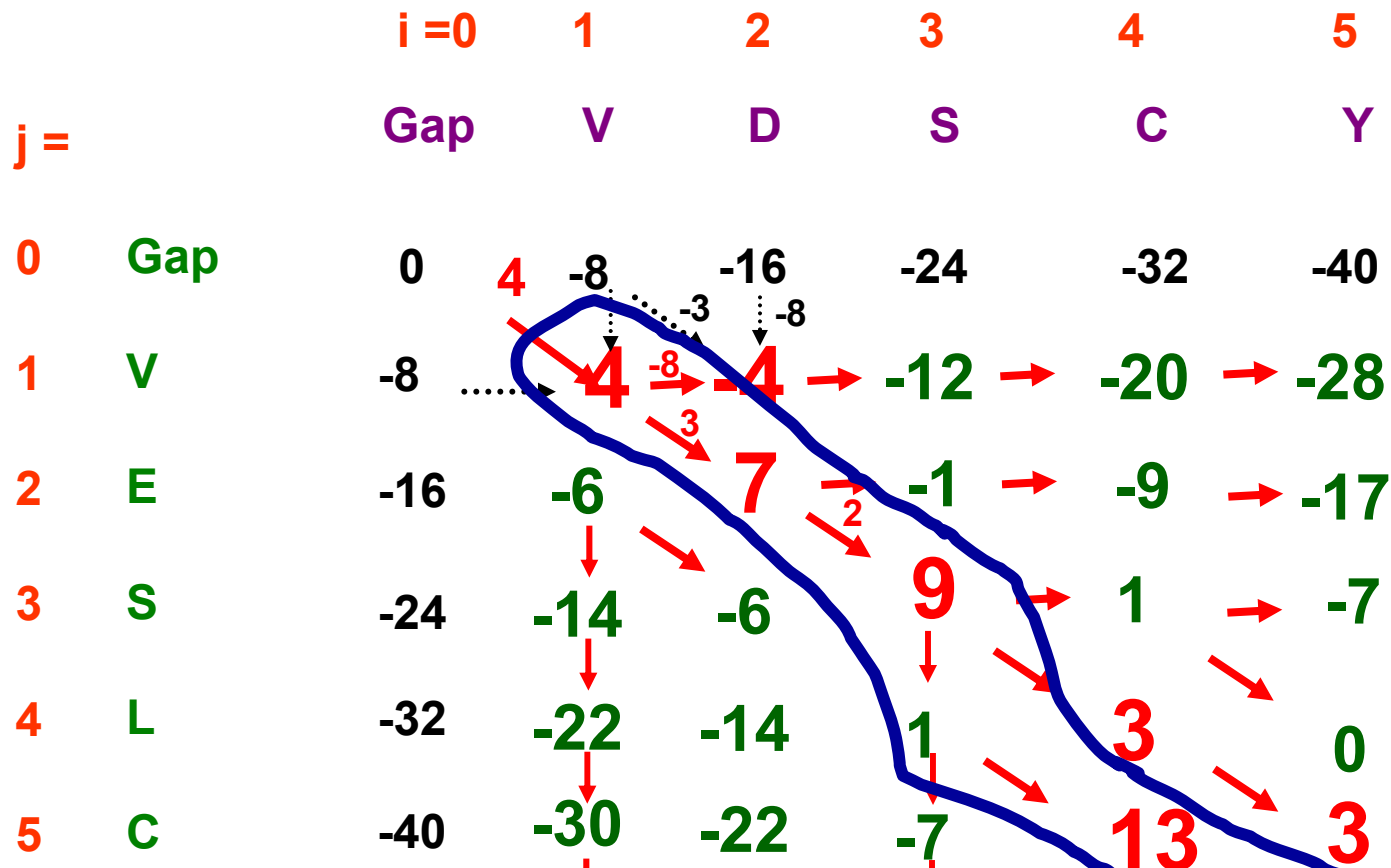
# The Traceback:

After the alignment square is finished, start at the lower right and work backwards following the arrows to see how you got there...



# The Traceback gives the alignment:

V D S - C Y  
V E S L C Y



“Life must be lived forwards and understood backwards.”

- Søren Kierkegaard

# Local Alignment

- Temple Smith and Michael Waterman, 1981 – modified Needleman-Wunsch-Sellers

**Local alignment is the best scoring alignment of a substring in sequence x to a substring in sequence y.**

**KEY ELEMENT IS NOT TO FORCE THE ALIGNMENT TO GO TO THE ENDS OF THE SEQUENCES.**

**For sequence x, residues 1, 2, 3.....N, can pick up to  $\sim N^2$  substrings, i.e. start point a= 1,2.....N and end point b= 1, 2.....n. Same for sequence y,  $\sim M^2$  substrings. For any two substrings, we have the old  $O(mn)$  alignment problem, so the total number of possible alignments is  $\sim N^2M^2(NM)=O(M^3N^3)$ - UGLY!!!! Solveable in polynomial time, but a big polynomial!!!**

# Local Alignment

- Again, dynamic programming comes to the rescue!

**Same basic scheme for dynamic programming as before  
except....**

**Similarity matrix MUST include negative values for mismatches**

**--AND--**

**\*\*\*When the value calculated for a position in the scoring matrix is  
Negative, the value is set to zero.**

**THIS TERMINATES THE ALIGNMENT**

## Smith-Waterman:

Write one sequence across the top, and one down along the side

|            |     | <b>i = 0</b> | <b>1</b>                   | <b>2</b> | <b>3</b> | <b>4</b> | <b>5</b> |
|------------|-----|--------------|----------------------------|----------|----------|----------|----------|
| <b>j =</b> | Gap | 0            | 0                          | 0        | 0        | 0        | 0        |
| <b>0</b>   | Gap | 0            | 0                          | 0        | 0        | 0        | 0        |
| <b>1</b>   | V   | 0            | <b><math>S_{ij}</math></b> |          |          |          |          |
| <b>2</b>   | E   | 0            |                            |          |          |          |          |
| <b>3</b>   | S   | 0            |                            |          |          |          |          |
| <b>4</b>   | L   | 0            |                            |          |          |          |          |
| <b>5</b>   | C   | 0            |                            |          |          |          |          |
| <b>6</b>   | Y   | 0            |                            |          |          |          |          |

Local alignments: Smith-Waterman

$$S_{ij} = \max \left\{ \begin{array}{l} S_{i-1, j-1} + \sigma(x_i, y_j) \text{ (diagonal)} \\ S_{i-1, j} - A \text{ (from left to right)} \\ S_{i, j-1} - A \text{ (from top to bottom)} \\ 0 \end{array} \right.$$

**Programs for Global and Local alignments:  
Biology workbench**

**<http://workbench.sdsc.edu/>**

**Bill Pearson's Web Page**

**<http://fasta.bioch.virginia.edu/>**

**NCBI, Expassy**

# Amino Acid Substitution Matrices

Margaret Dayhoff, 1978, PAM Matrices

**\*\*Evolutionary model\*\*** based on a small data set.

Assumes symmetry:  $A \rightarrow B = B \rightarrow A$

Assumes amino acid substitutions observed over short periods of time can be extrapolated to long periods of time

71 groups of protein sequences, 85% similar

1572 amino acid changes.

Functional proteins → "Accepted" mutations by natural selection

**PAM1 matrix means 1% divergence between proteins - i.e. 1 amino acid change per 100 residues. Some texts re-state this as the probability of each amino acid changing into another is ~ 1% and probability of not changing is ~99%**

# Construction of a Dayhoff Matrix: PAM1

**Step 1**: Measure pairwise substitution frequencies for each amino acid within families of related proteins

... . GDS**F**H**Y**FV**S**H**G**... . .  
... . GDS**F**H**Y**YV**S****F**G... . .  
... . GDS**Y**H**Y**FV**S****F**G... . .  
... . GDS**F**H**Y**FV**S****F**G... . .  
... . GDS**F**H**F**FV**S****F**G... . .

900 Phe (F)....+ another 100 probable Phe but...

100 Phe (F) → 80 Tyr (Y), 3 Trp (W), 2 His (H)....

Gives  $f_{ab}$ , i.e.  $f_{FY}=80$   
 $f_{FW}=3$

....by evolution!



# Do this for all 20 amino acids

|   | C        | D        | E        | F | G | H | I | ..... |
|---|----------|----------|----------|---|---|---|---|-------|
| C | $f_{CC}$ | $f_{CD}$ | $f_{CE}$ |   |   |   |   |       |
| D | $f_{DC}$ |          |          |   |   |   |   |       |
| E | $f_{EC}$ |          |          |   |   |   |   |       |
| F |          |          |          |   |   |   |   |       |
| G |          |          |          |   |   |   |   |       |

Gives  $f_{ab}$  = pairwise exchange frequency

**Implicit assumption - 1st order Markov chain transition model**

**Step 2: Calculate the relative probabilities of pairwise exchange of a→b**

**P<sub>a</sub>** = Probability of amino acid a

**f<sub>ab</sub>** = number of substitutions between a and b

**f<sub>a</sub>** = total number of substitutions involving amino acid a

$$\mathbf{f}_a = \sum_{a \neq b} \mathbf{f}_{ab}$$

**f** = total number of mutations in the group of related sequences =  $\sum_a \mathbf{f}_a$

Define relative probability **M'ab** as:

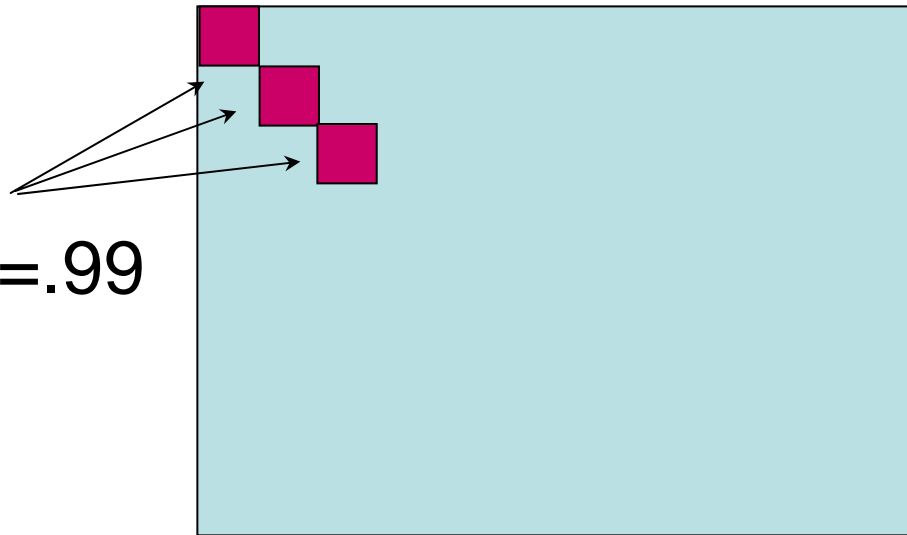
$$\mathbf{M}'_{ab} = \text{Pr}(a \rightarrow b) = \mathbf{f}_{ab} / \mathbf{f}$$

**Step 3: Scale relative probabilities to obtain a 1% total chance of any amino acid changing into a different amino acid**

**i.s. scale  $M'$  to ensure that:**

$$\sum_{a \neq b} P_a M_{ab} = .01$$

i.e.  $\sum P_a M_{aa} = .99$



**Step 4**: This involves defining a “relative mutability”  
index  $\mathbf{m}_a$  for each amino acid

$$\mathbf{m}_a = \frac{\mathbf{f}_a}{100 \mathbf{f P a}}$$

*Number of mutations involving amino acid a*

*“exposure of ‘a’ to mutation”  
Prob(a)\* total number of  
mutations weighted per 100 sites*

Set adjusted probability  $\mathbf{M}_{ab} = \mathbf{M}'_{ab} \mathbf{m}_a$

**Step 5**: Calculate evolutionary distance scale  
so that only 1/100 amino acids change

*$M_{aa}$  reflects amino acid conservation*

$$M_{aa} = 1 - \sum_b M_{ab}$$

*example*

$$M_{FF} = 1 - (\text{adjusted probability of Phe mutations})$$

***\*\*Use a scale factor  $\lambda$  so that  $M_{aa}$  is  $\sim 0.99$   
i.e. chance of it mutating is  $\sim 1\%$***

***i.e. this defines a PAM1 matrix....***

$$\mathbf{M}_{aa} = 1 - \lambda \sum_{a \neq b} \mathbf{M}_{ab} = \sim 0.99$$

***$\lambda$  is our evolutionary scale factor***

**... and for any particular mutation probability,  
 $\lambda \mathbf{M}_{ab}$  reflects the normalized measure of how likely  
amino acid b will replace amino acid a over 1 PAM**

# Real PAM1 values

## Amino Acid Change   PAM 1 Probability Score

|     |        |
|-----|--------|
| F→A | 0.0002 |
| F→R | 0.0001 |
| F→N | 0.0001 |
| F→D | 0.0000 |
| F→C | 0.0000 |
| F→Q | 0.0000 |
| F→E | 0.0000 |
| F→G | 0.0001 |
| F→H | 0.0002 |
| F→I | 0.0007 |
| F→L | 0.0013 |
| F→K | 0.0000 |
| F→M | 0.0001 |
| F→F | 0.9946 |
| F→P | 0.0001 |
| F→S | 0.0003 |
| F→T | 0.0001 |
| F→W | 0.0001 |
| F→Y | 0.0021 |
| F→V | 0.0001 |

**SUM = 1.0**

*Note – this is really just  
1 column in a much bigger  
probability matrix*

|   | ..... E | F      | G | ..... |
|---|---------|--------|---|-------|
| A |         | 0.0002 |   |       |
| C |         | 0.0000 |   |       |
| D |         | 0.0000 |   |       |
| E |         | 0.0000 |   |       |
| F |         | 0.9946 |   |       |
| G |         | 0.0001 |   |       |

Next, assume that mutations at each site are independent of previous mutations. Therefore, calculate changes predicted for more distantly related proteins that have undergone  $N$  mutations/100 amino acids by multiplying the PAM1 matrix against itself  $N$  times.

**Example: PAM2 matrix:**

$$\begin{array}{c|ccc} & \text{aa1} & \text{aa2} & \text{aa3} \\ \hline \text{aa1} & a & b & c \\ \text{aa2} & d & e & f \\ \text{aa3} & g & h & i \end{array} \quad \times \quad \begin{array}{c|ccc} & \text{aa1} & \text{aa2} & \text{aa3} \\ \hline \text{aa1} & a & b & c \\ \text{aa2} & d & e & f \\ \text{aa3} & g & h & i \end{array}$$

$$\begin{array}{c|ccc} & \text{aa1} & \text{aa2} & \text{aa3} \\ \hline \text{aa1} & A & B & C \\ \text{aa2} & D & E & F \\ \text{aa3} & G & H & I \end{array}$$

$$A = a^2 + bd + cg + \dots$$

$$B = ab + be + ch + \dots$$

$$C = ac + bf + ci + \dots$$

$$D = da + ed + fg + \dots$$





| <u>Amino Acid Change</u> | <u>PAM 1 Score</u> | <u>PAM 250 Score</u> |
|--------------------------|--------------------|----------------------|
| F→A                      | 0.0002             | 0.04                 |
| F→R                      | 0.0001             | 0.01                 |
| F→N                      | 0.0001             | 0.02                 |
| F→D                      | 0.0000             | 0.01                 |
| F→C                      | 0.0000             | 0.01                 |
| F→Q                      | 0.0000             | 0.01                 |
| F→E                      | 0.0000             | 0.01                 |
| F→G                      | 0.0001             | 0.03                 |
| F→H                      | 0.0002             | 0.02                 |
| F→I                      | 0.0007             | 0.05                 |
| F→L                      | 0.0013             | 0.13                 |
| F→K                      | 0.0000             | 0.02                 |
| F→M                      | 0.0001             | 0.02                 |
| F→F                      | 0.9946             | 0.32                 |
| F→P                      | 0.0001             | 0.02                 |
| F→S                      | 0.0003             | 0.03                 |
| F→T                      | 0.0001             | 0.03                 |
| F→W                      | 0.0001             | 0.01                 |
| F→Y                      | 0.0021             | 0.15                 |
| F→V                      | 0.0001             | 0.05                 |

These are the  $M_{ab}$  values!  
i.e. the chance that one amino acid will replace another at 250 PAMs in two proteins that are evolutionarily related to each other!

**SUM = 1.0**

## ***PAM 250 matrix – 250% expected change***

**Sequences still ~ 15-30 % similar, i.e. Phe will match Phe ~ 32% of the time  
Ala will match Ala ~ 13% of the time**

### **Expected % similarity**

**Other PAM matrices:**

|                      |   |                                  |
|----------------------|---|----------------------------------|
| <b>PAM 120 – 40%</b> | } | <b>Use for similar sequences</b> |
| <b>PAM 80 – 50%</b>  |   |                                  |
| <b>PAM 60 – 60%</b>  |   |                                  |

**PAM250 – 15-30% similarity.**



# Where do the numbers in the PAM250 Matrix table come from?

---

## Step 6: Calculate relatedness odds

*Chance that two amino acids in a sequence alignment come from related proteins via evolution versus the chance that they are from two unrelated proteins aligned by chance.*

$M_{ab}$  = prob. that b replaces a in related proteins

-vs -

$P_a^{\text{ran}}$  = prob. that b replaces a because the proteins are completely unrelated...i.e. a was there by chance

Now,  $P_a^{\text{ran}} = f_a$ , the frequency of occurrence of amino acid a

# Where do the numbers in the PAM250 Matrix table come from?

---

**Step 6**: Calculate relatedness odds

Relative odds of evolution rather than chance:

$$R_{ij} = \frac{M_{ij}}{f_i}$$

# Where do the numbers in the PAM250 Matrix table come from?

---

**Step 7**: Calculate log (relatedness odds) and multiply by 10 to clear fractional values

Example: Phe→Tyr (which must = Tyr→Phe)

$$R_{ij} = \frac{M_{ij}}{f_i}$$

$$M_{FY} = 0.15$$

$$f_{\text{Phe}} = 0.04$$

$$\text{So } R_{FY} = 0.15 / 0.04 = 3.75$$

$$\text{Log}_{10} R_{FY} = \text{Log}_{10} (3.75) = 0.57$$

$$10 \times 0.57 = 5.7$$

Likewise

$$M_{YF} = 0.20$$

$$f_{\text{Tyr}} = 0.03$$

$$\text{So } R_{YF} = 6.7$$

$$\text{Log}_{10} (6.7) = 0.83$$

$$10 \times 0.83 = 8.3$$

So average =  $(5.7 + 8.3) / 2 = 7$ ....the number in the PAM250 table!





# ***Remember...***

*Saw last time how to use these  
numbers + dynamic  
programming in order to “score”  
an alignment...*

***But we have to use the right matrix!!!***

***PAM 250 matrix – 250% expected change***

Sequences still ~ 15-30 % similar, i.e. Phe will match Phe ~ 32% of the time  
Ala will match Ala ~ 13% of the time

**Expected % similarity**

Other PAM matrices: PAM 120 – 40%  
PAM 80 – 50%  
PAM 60 – 60% } Use for similar sequences

**PAM250 – 15-30% similarity.**

***Use the correct PAM matrix for alignments based on how similar the sequences to be aligned are! But wait.....how do we know that in the first place? Usually don't!!!!.***

***So..... try PAM200, PAM120, PAM60, PAM80, and PAM30 matrix and use the one that gives the highest ungapped alignment score***

# Alternative amino acid matrices

Problems with Dayhoff:

- Based on amino acids, not nucleotides.
- Assumes evolutionary model with explicit phylogenetic relationships, and circular arguments: alignment → matrices; matrices → new alignments.
- Based on a small set of closely related molecules.
- **Gonnett, Cohen & Benner**
  - All against All database matching using DARWIN
  - 1,700,000 matches
  - Compile mutation matrices at different PAMs DIRECTLY*
- **BLOSUM = Blocks Amino Acid Substitution Matrices-Henikoff&Henikoff 1992**
  - based on a much larger dataset from ~500 Prosite families identified by Bairoch using conserved amino acid patterns “blocks” that define each family.

Typically used for multiple sequence alignment.  
AA substitutions noted, log odds ratios derived.

for example...Block patterns 60% identical give rise to Blosum60 matrix, etc....i.e. conservation of functional blocks based on un-gapped alignments.  
Blosum62 - best match between information content and amount of data

*Not based on explicit evolutionary model*

# GAPS

AKHFRGCVS  
AKKF--CVG

- **Linear Gap Penalty**

$$W_n = n\gamma,$$

**$n = \#$  of gaps,  $\gamma =$  gap penalty**

- **Affine gap penalty**

$$W_n = g + n\gamma,$$

**$n = \#$  of gaps,  $\gamma =$  gap extension penalty,  
and  $g =$  gap opening penalty**

[Search](#)

[Set subsequence](#) From: \_\_\_\_\_ To: \_\_\_\_\_

[Choose database](#)

[Do CD-Search](#)

Now: **BLAST!** or [Reset query](#) [Reset all](#)

**Options** for advanced blasting

[Limit by entrez query](#) \_\_\_\_\_ or select from:

[Composition-based statistics](#)

[Choose filter](#)  Low complexity  Mask for lookup table only  Mask lower case

[Expect](#)

[Word Size](#)

[Matrix](#)  [Gap Costs](#)

[Search](#)

[Set subsequence](#) From: \_\_\_\_\_ To: \_\_\_\_\_

[Choose database](#) | nr | \_\_\_\_\_ | \_\_\_\_\_

[Do CD-Search](#)

Now: **BLAST!** or **FASTA** **FASTX** **FASTQ**

### Options for advanced blasting

[Limit by entire query](#) \_\_\_\_\_ or select from: All organisms | \_\_\_\_\_

[Composition-based statistics](#)

[Choose filter](#)  Low complexity  Mask for lookup table only  Mask lower case

[Expect](#) \_\_\_\_\_ 10

[Word Size](#) | 3 | \_\_\_\_\_

[Matrix](#) | BLOSUM62 | \_\_\_\_\_ | [Gap Costs](#) | [Existence](#) | [Extension](#) | 1 | \_\_\_\_\_

# Simplified Alignment Statistics

- How can we tell how good an alignment is based on its score? What are the chances that two random sequences would give a similar score when they were aligned?
- Consider an easier problem – what is the longest run of heads I will get in a random series of coin tosses?  
Fair coin  $p=0.5$ , Erdős and Rényi – longest run =  $\log_{1/p}(n)$   
here this is  $\log_2(n)$ . If  $n=100$ , longest run is 6.65
- For two sequences of length  $n$  and  $m$ , we're doing  $nm$  comparisons, so the longest length of the predicted match would be  $\log_{1/p}(mn)$
- More precisely, the expectation value, or the mean of the longest match turns out to be  $E(M) \sim \log_{1/p}(Kmn)$  where  $K$  is a constant that depends on amino acid composition.  
**....OK, this is really only true for ungapped local alignments and I'm neglecting edge effects and mismatches**

# A few notes...

- $E(M) \sim \log_{1/p}(Kmn)$  means that the match length gets bigger as the log of the product of the sequence lengths. Using the amino acid substitution matrices, we can turn these match lengths into alignment scores,  $S$ .
- More commonly see two parameters used:  $\lambda = \ln(1/p)$  and the parameter  $K$  we already talked about.
- We want to know the number of High-Scoring Pairs, HSP, (i.e. high scoring runs of amino acids).
- This number of HSPs,  $E$ , that exceed some score  $S$  is given by  $E = Kmne^{-\lambda S}$
- So we can evaluate how good a sequence scores, (i.e. its  $S$ ) by looking at how many HSPs (i.e.  $E$  value) we would expect for that score.

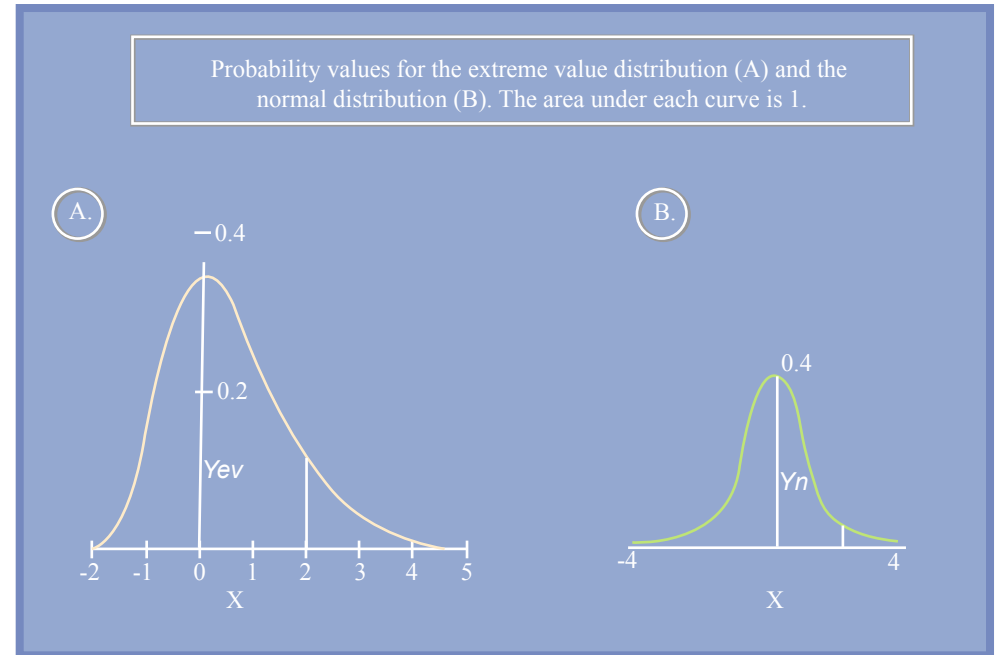


## **Notes (cont).**

- **Where do we get that distribution function that tells us how  $E$  and  $\sigma$  are related? Need to look at the scores in some model of aligned random sequences.....**

# Notes (cont)

- The random sequence alignment scores would give rise to an “extreme value” distribution – like a skewed gaussian.
- Called Gumbel extreme value distribution



For a normal distribution with a mean  $m$  and a variance  $\sigma$ , the height of the curve is described by  $Y=1/(\sigma\sqrt{2\pi}) \exp[-(x-m)^2/2\sigma^2]$

For an extreme value distribution, the height of the curve is described by  $Y=\exp[-x-e^{-x}]$  ...and  $P(S \geq x) = 1 - \exp[-e^{-\lambda(x-u)}]$  where  $u=(\ln Km)/\lambda$

Can show that mean extreme score is  $\sim \log_2(nm)$ , and the probability of getting a score that exceeds some number of “standard deviations”  $x$  is:  
 $P(S \geq x) \sim Kmne^{-\lambda x}$ . \*\*\*  $K$  and  $\lambda$  are tabulated for different matrices \*\*\*\*

For the less statistically inclined:  $E \sim Kmne^{-\lambda S}$

- **Two ways to get the  $K$  and  $\lambda$  parameters:**


**1- For many amino acid substitution matrices, Altschul and Gish have tabulated their score distribution for 10,000 random amino acid sequences using various gap penalties**

**2- Even better! Calculate the distribution for the two sequences you are aligning by keeping one of them fixed and scrambling the other one – this preserves BOTH sequence length and amino acid composition!**

## Example

SEQ1: FWLEVEGNSMTAPTG  
SEQ2: FWLDVQGDSMTAPAG

USE BLOSUM62 MATRIX



**Biology WorkBench**  
click here to  
toggle between  
menus and buttons

*WE  
Moved!* <http://workbench.sdsc.edu/>

Version 3.2

**BL2SEQ**  
Compare proteins to each other with BLAST

Seq1 (User Entered)

- and -

Seq2 (User Entered)

Query: >Seq1  
Length = 15  
Reference: Query= Seq1  
(15 letters)

>Seq2  
Length = 15

Score = 30.4 bits (67), Expect = 3e-07  
Identities = 11/15 (73%), Positives = 14/15 (93%)

Query: 1 FWLEVEGNSMTAPTG 15  
FWL+V+G+SMTAP G  
Sbjct: 1 FWLDVQGDSMTAPAG 15

| Lambda | K     | H     |
|--------|-------|-------|
| 0.319  | 0.135 | 0.464 |

| Gapped<br>Lambda | K      | H     |
|------------------|--------|-------|
| 0.267            | 0.0410 | 0.140 |

Matrix: BLOSUM62  
Gap Penalties: Existence: 11, Extension: 1  
Number of Hits to DB: 9  
Number of Sequences: 0  
Number of extensions: 1  
Number of successful extensions: 1  
Number of sequences better than 10.0: 1  
Number of HSP's better than 10.0 without gapping: 1  
Number of HSP's successfully gapped in prelim test: 0  
Number of HSP's that attempted gapping in prelim test: 0  
Number of HSP's gapped (non-prelim): 1  
length of query: 15  
length of database: 15  
effective HSP length: 0  
effective length of query: 24  
effective length of database: 15  
effective search space: 360  
effective search space used: 360

Raw score = 67

Bit score:  $S = \frac{\lambda R - \ln K}{\ln(2)}$

$$S = \frac{(.267)(67) - \ln(.0410)}{0.693} = 30.4$$

$E \sim Kmne^{-\lambda S}$  which is  
Equivalent to:

$$E = mn2^{-s}$$

$$E = (24)(15)(2^{-30.4}) = 2.54e-07$$

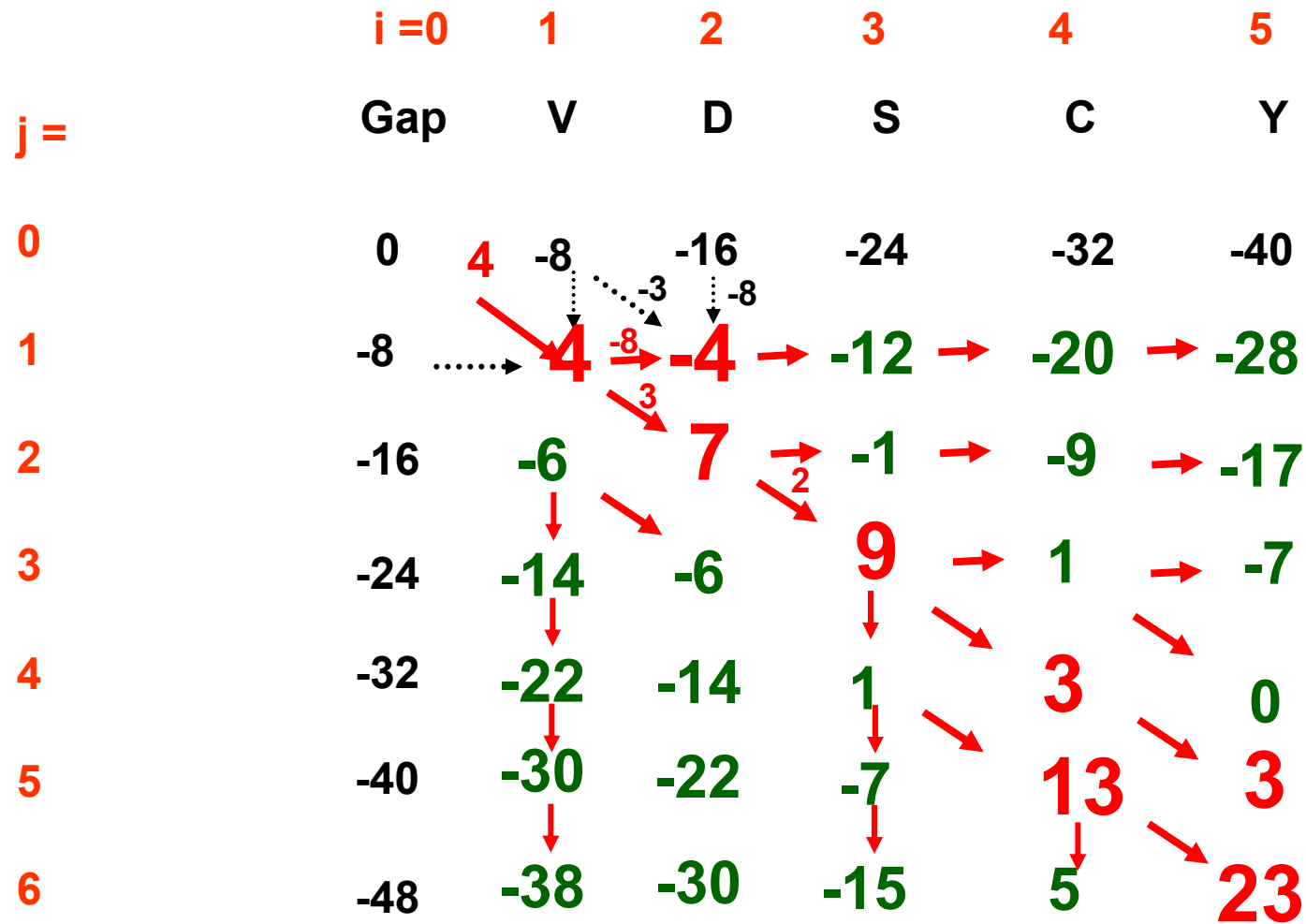
# Multiple Sequence Alignments

- Sequences are aligned so as to bring the greatest number of single characters into register.
- If we include gaps, mismatches, then even dynamic programming becomes limited to ~ 3 sequences unless they are very short....need an alternative approach...

**Why?**

# Consider the 2 sequence comparison

.....an  $O(mn)$  problem – order  $n^2$



## For 3 sequences....

ARDFSHGLENKLLGCDSMRWE  
GRDYKMALLEQWILGCD-MRWD  
SRDW--ALIEDCMV-CNFFRWD

*An  $O(mnj)$  problem !*

Consider sequences each 300 amino acids

2 sequences –  $(300)^2$

3 sequences –  $(300)^3$

but for  $v$  sequences –  $(300)^v$

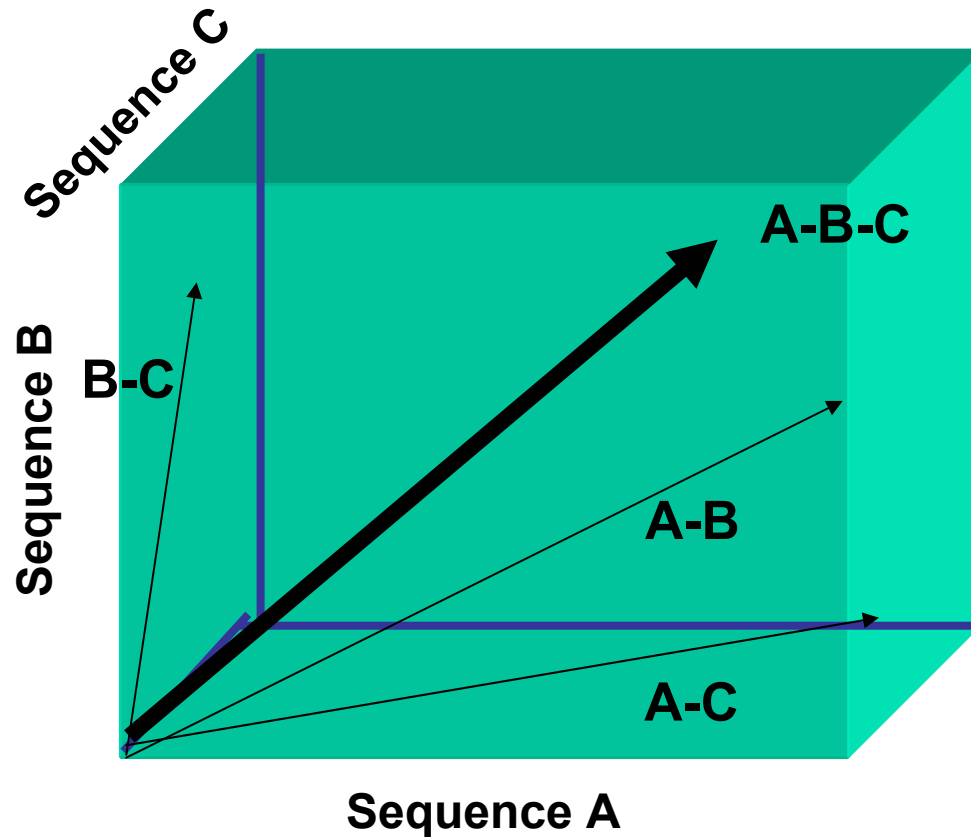
Uh Oh !!!

Our polynomial problem  
Just became exponential!



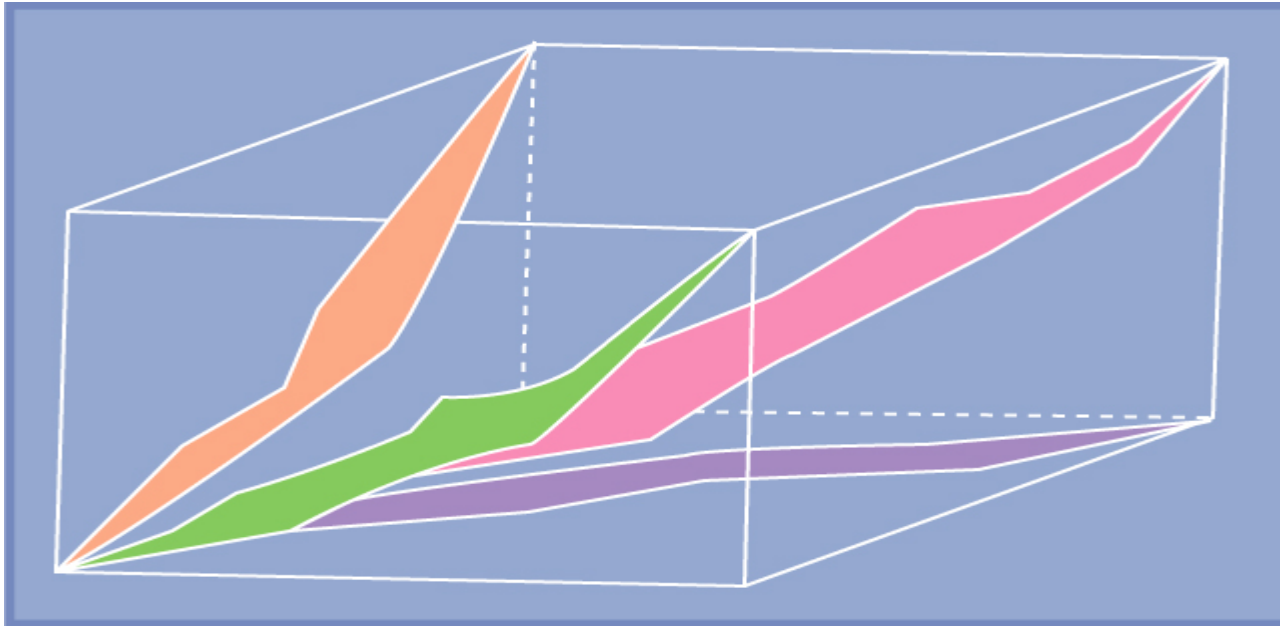
# Consider pairwise alignments between 3 sequences

**Carillo and Lipman – Sum of Pairs method**



*Do we need to  
Score each node?*

**Get the multiple alignment score within the cubic lattice by  
Adding together the scores of the pairwise alignments...**

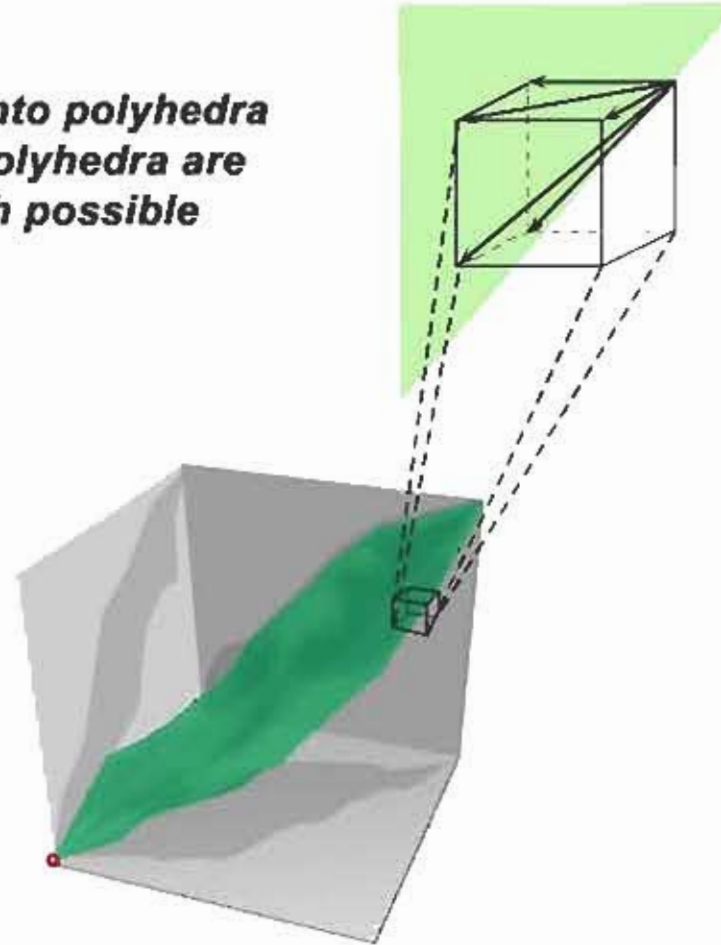


**In practice, doesn't give optimal alignment...But we're close!**

**Seems reasonable that the optimal alignment won't be far from the diagonal we were on...so we just set bounds on the location of the msa within the cube based on each pairwise-alignment.**

**Then just do dynamic programming within the volume defined by the pre-imposed bounds**

***...the volume is broken into polyhedra  
and the borders of the polyhedra are  
defined by paths through possible  
alignments***



**Still takes too long for more than three  
sequences...need a better way!**

- **Progressive Methods of Multiple Sequence Alignment**

**Concept – simple:**

**1-Use DP to build pairwise alignments of most closely related sequences**

**2- Then progressively add less related sequences or groups of sequences...**

# ClustalW

*Higgins and Sharp 1988*

- 1- Do pairwise analysis of all the sequences (you choose similarity matrix).
- 2- Use the alignment scores to make a phylogenetic tree.
- 3- Align the sequences to each other guided by the phylogenetic relationships in the tree.

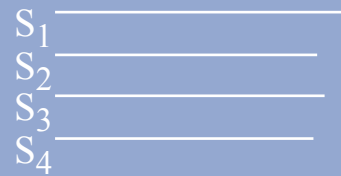
**New features: Clustal  $\boxtimes$  ClustalW (allows weights)  $\boxtimes$  ClustalX (GUI-based**

*Weighting is important to avoid biasing an alignment by many sequence Members that are closely related to each other evolutionarily!*

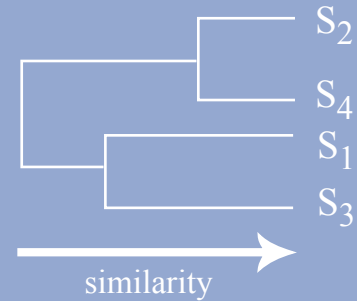
# Steps in Multiple Alignment

## Pairwise Alignment

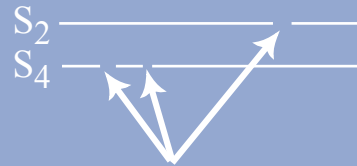
Example - 4 sequences  $S_1$   $S_2$   $S_3$   $S_4$



6 pairwise comparisons  
then cluster analysis



## Multiple alignment following the tree from A



align most similar pair

Gaps to optimize alignment



align next most similar pair

New gap to optimize  
alignment of  $(S_2S_4)$  with  $(S_1S_3)$



align alignments-preserve gaps

# Progressive Alignments

**Note that the final msa is EXTREMELY DEPENDENT on the initial pairwise sequence alignments!**

**If the sequences are close in evolution, and you can see the alignment – GREAT!**

**If the sequences are NOT close in evolution, and you CANNOT See the alignment – errors will be propagated to the final msa**

**Has led to other approaches to do msa's that aren't so Dependent on initial states....i.e. genetic algorithm**

# Finding patterns (i.e. motifs and domains) in Multiple Sequence Analysis

---

## Block Analysis, Position Specific Scoring Matrices (PSSM)

BUILD an msa from groups of related proteins

**BLOCKS** represent a conserved region in that msa that is **LACKING IN GAPS** – i.e. no insertions/deletions

The **BLOCKS** are typically anywhere from 3-60 amino acids long, based on exact amino acid matches – i.e. alignment will tolerate mismatches, but doesn't use any kind of PAM or BLOSUM matrix...in fact they generate the BLOSUM matrix!

**A single proteins contain numerous such BLOCKS separated by stretches of intervening sequences that can differ in length and composition.**

These blocks may be whole domains, short sequence motifs, key parts of enzyme active sites etc, etc.

**BLOCKS database....so far exploration limited. Lots of stuff to probe!**



# Can use these conserved BLOCKS to derive a PSSM

- The dirty secret behind prosite! Scansite! And in a twised way Psi-BLAST!

12345.....11  
...GDSFHVSHG...  
...GDAFHYSFG...  
...GDSYHFLSFG...  
...SDSFHYFMSFG...  
...GDSFHFFASFG...

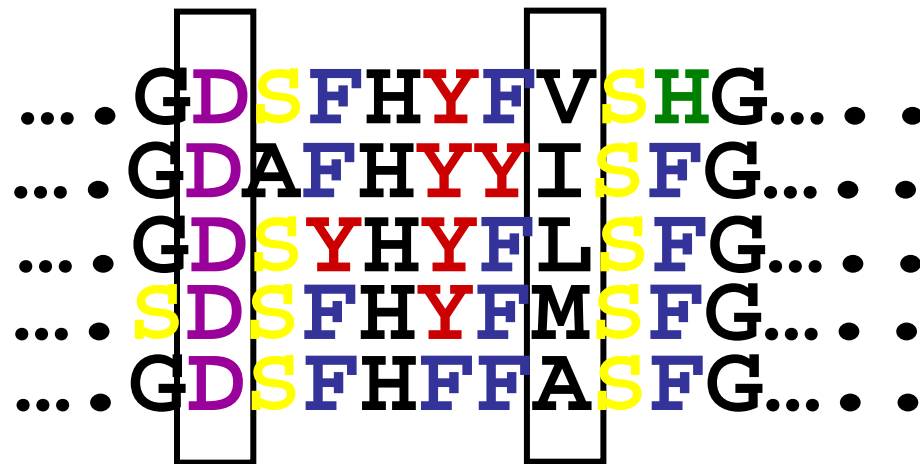
Now build a matrix with 20 amino acids as the columns, and 11 rows  
For the positions in the BLOCK



We can now use the PSSM to search a database for other proteins that have the BLOCK (or motif).

**Problem 1 – We need to think about what kind of information is Contained within the PSSM.**

→Leads to concepts of Information Content & Entropy (next time)



**Problem 2 –The PSSM must accurately represent the expected BLOCK Or motif....and we have only limited amounts of data! Is it a good statistical Sampling of the BLOCK/motif? Is it too narrow because of small dataset? Should we broaden it by adding extra amino acids that we choose using Some type of randomization scheme (called adding pseudocounts). If so, How many should we add?**