Lecture 10

Multiple Sequence Alignment

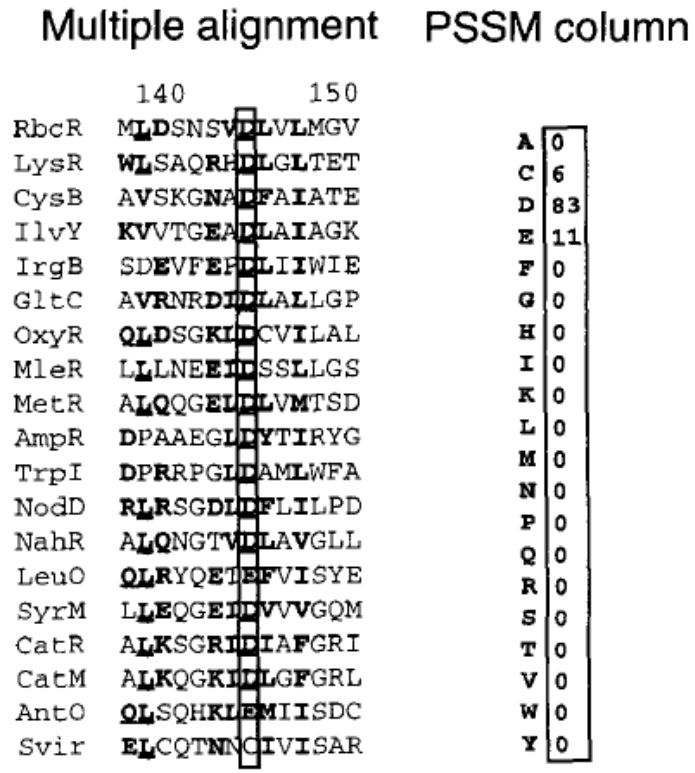
```
3GXK_1|Chains      VGYGDITQVETSGASSKTSRQDKLEYDGVRASHTMAQTDAGRMEKYKSFINNVAKKHVVD 60
5JDO_1|Chain       ---GEIKVELE-------------DSDDVAAACELRAQLAG---------VSIASGILLR 35
6HIT_1|Chains      ------------------------------------------------------XSLS 4
1IRD_1|Chain       ------------------------------------------------------VLS 3
5HU6_1|Chain       ------------------------------------------------------VLS 3
4MQC_1|Chain       ------------------------------------------------------VLS 3
                                                                           :

3GXK_1|Chains      PAVIAAIISRESRA--------GN-------VI---FNTTPPGWGDNYNGFGLMQVDKRY 102
5JDO_1|Chain       PAVIRNATTEFSRKKSEDILAKGGAAVERASAAVDRV----SGLDKTNETAQKVR--KAA 89
6HIT_1|Chains      SKQKATVKDFFSK-MSTRSDDIGAEALSRLVAVYPQTKSYFSHWKDASPGSAPVR----- 58
1IRD_1|Chain       PADKTNVKAAWGK-VGAHAGEYGAEALERMFLSFPTTKTYFPHFD-LSHGSAQVK----- 56
5HU6_1|Chain       PADKTNVKAAWGK-VGAHAGEYGAEALERMFLSFPTTKTYFPHFD-LSHGSAQVK----- 56
4MQC_1|Chain       PADKTNVKAAWGK-VGAHAGEYGAEALERMFLSFPTTKTYFPHFD-LSHGSAQVK----- 56
                             .:        *                              ::

3GXK_1|Chains      HEPRGAWNSEEHIDQATGILVNFIQLI---------------------------------- 129
5JDO_1|Chain       ---AVAHHALEHVKEEVEIVAKKVNEIIELTAGATEHAKGAKANGDASAVKVSNLLARAK 146
6HIT_1|Chains      ------KHGITI----MGGVYDAVGKIDDLKGGL---------------LSLSELHAFM- 92
1IRD_1|Chain       ------GHGKKV----ADALTNAVAHVDDMPNAL---------------SALSDLHAHK- 90
5HU6_1|Chain       ------GHGKKV----ADALTNAVAHVDDMPNAL---------------SALSDLHAHK- 90
4MQC_1|Chain       ------GHGKKM----ADALTNAVAHVDDMPNAL---------------SALSDLHAHK- 90
                           :.          : . :  :
```

Other variations of scoring matrices

- Position Specific Scoring Matrix
- Structural information
- 3D-1D mapping



Features and performance of methods for PSSM construction.

Method	Considers substitutions	Sensitive to number of sequences	Position-specific	Relative performance*
Normalized counts	No	No	No	-
Average score	Yes	No	No	+
Pseudo-counts				
Background	No	Yes	No	+
Substitution	Yes	Yes	No	++
Dirichlet	Yes	Yes	Yes	+++
Position-based†	Yes	Yes	Yes	++++

Position-based Sequence Weights

- Use to
  - reduce redundancy
  - emphasize diversity
- Based on
  - distance between a sequence and an ancestral or generalized sequence
  - weights on the diversity observed at each position in the alignment
- Application is in
  - MSA
  - Sequence searching

The Problem

Given an alignment of several sequences (we will defer until later how to construct such an alignment), how should we define scores for aligning to a single new sequence?

This problem raises three distinct, and deep questions:

1. How does one deal with correlation among the aligned sequences?
2. How many independent observations does an alignment column represent?
3. How does account for small sample size and for prior knowledge?

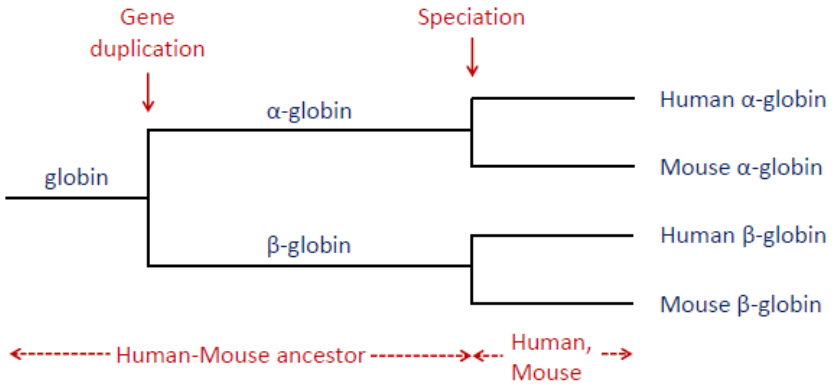
More simply, leaving aside the question of gap scores, how should one score the alignment of a multiple alignment column to a single letter?

We will defer the third question until later.

Questions

- How can one formalize this problem?
- Can one recast the problem of finding appropriate sequence weights as an optimization problem?

Digression: Orthology and Paralogy

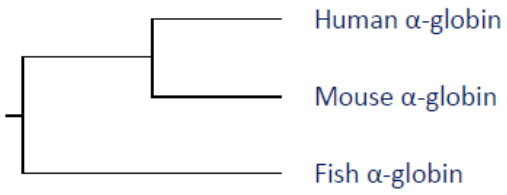


Homology: Two genes or proteins are *homologous* if they share a common ancestor.

Orthology: Two genes or proteins are *orthologous* if they diverged by speciation.

Paralogy: Two genes or proteins are *paralogous* if they diverged by gene duplication.

Sequence Trees and Phylogenetic Trees

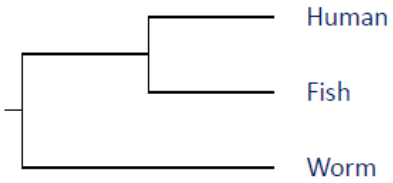


For orthologous genes or proteins, a tree reflecting sequence relationships should be congruent with the *phylogentic tree* of species relationships.

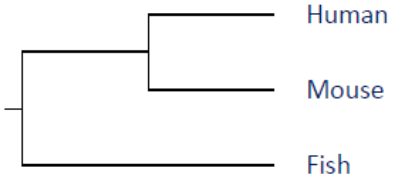
For paralogous genes or proteins, a tree reflecting sequence relationships may be incongruent with the phylogentic tree of species relationships.

Over the course of evolution, it is possible that in a particular protein family different paralogs are lost in different species. In that case there may be no set of orthologs for that family from which a valid phylogenetic tree may be reconstructed.

Weights Depend on a Set of Sequences



Given orthologous genes or proteins from these three organisms, the fish sequence should be downweighted.



Given orthologous genes or proteins from these three organisms, the fish sequence should be upweighted.

In other words, a weight is never *intrinsic* to a sequence. It is associated with a sequence only in the context of a set of other sequences.

First choice: Purging

A simple approach to dealing with sequence correlation is simply removing or ignoring sequences that are more than $X\%$ identical to some sequence already included.

Advantages:

- Very fast and simple.
- Duplicating a sequence does not alter results.

Disadvantages:

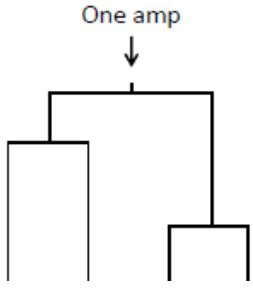
- No definition of what is being optimized.
- Dependent on order in which sequences are considered.
- Some information is clearly lost.

Second choice: Tree based weights

Reformulation: Let $\boldsymbol{T}$ be a continuous one-dimensional quantitative trait that undergoes Brownian motion over the course of evolution. Assume it has value t at the root of a tree, and the values $\vec{t}$ at the tree's leaves.

Question: Given $\vec{t}$, what is the maximum-likelihood estimator for t?

Solution: Let $l_{i,i}$ be the distance from the root to leaf i, and let $l_{i,j}$ be the distance from the root to the last common ancestor of leaves i and j. Then the variance of the random variable t_i is proportional to $l_{i,i}$, and the covariance of t_i and t_j is proportional to $l_{i,j}$. Let $\mathbf{M}$ be the variance-covariance matrix, $\vec{1}$ be a column vector of 1s, and $\vec{w} = (\mathbf{M}^{-1}\vec{1})/(\vec{1}'\mathbf{M}^{-1}\vec{1})$. Then $\hat{t} = \vec{w} \bullet \vec{t}$ is the estimator we seek.



Equivalent to: Make the vertical edges of the tree of resistant wire, and ground the leaves. Apply a voltage so that one amp flows into the root. The current that flows out each leaf is the weight for that leaf.

Felsenstein, J. (1973) "Maximum-likelihood estimation of evolutionary trees from continuous characters." *Am. J. Hum. Genet.* **25**:471-492.

Advantages:

- Well-formulated as an optimization problem.
- Independent of sequence order.
- Uses all information.
- Tree may be *rooted* anywhere, allowing outgroups to contribute.

Possible disadvantages:

- Leaves farther from the root are downweighted.
- Assumes an evolutionary tree relating the sequence.

Major disadvantage:

- Requires the construction of an evolutionary tree, a hard and time-consuming problem.

Lecture 11-12

Multiple Sequence Alignment

```
3GXK_1|Chains      VGYGDITQVETSGASSKTSRQDKLEYDGVRASHTMAQTDAGRMEKYKSFINNVAKKHVVD 60
5JDO_1|Chain       ---GEIKVELE-------------DSDDVAAACELRAQLAG---------VSIASGILLR 35
6HIT_1|Chains      ------------------------------------------------------XSLS 4
1IRD_1|Chain       ------------------------------------------------------VLS 3
5HU6_1|Chain       ------------------------------------------------------VLS 3
4MQC_1|Chain       ------------------------------------------------------VLS 3
                                                                           :

3GXK_1|Chains      PAVIAAIISRESRA--------GN-------VI---FNTTPPGWGDNYNGFGLMQVDKRY 102
5JDO_1|Chain       PAVIRNATTEFSRKKSEDILAKGGAAVERASAAVDRV----SGLDKTNETAQKVR--KAA 89
6HIT_1|Chains      SKQKATVKDFFSK-MSTRSDDIGAEALSRLVAVYPQTKSYFSHWKDASPGSAPVR----- 58
1IRD_1|Chain       PADKTNVKAAWGK-VGAHAGEYGAEALERMFLSFPTTKTYFPHFD-LSHGSAQVK----- 56
5HU6_1|Chain       PADKTNVKAAWGK-VGAHAGEYGAEALERMFLSFPTTKTYFPHFD-LSHGSAQVK----- 56
4MQC_1|Chain       PADKTNVKAAWGK-VGAHAGEYGAEALERMFLSFPTTKTYFPHFD-LSHGSAQVK----- 56
                             .:        *                            ::

3GXK_1|Chains      HEPRGAWNSEEHIDQATGILVNFIQLI---------------------------------- 129
5JDO_1|Chain       ---AVAHHALEHVKEEVEIVAKKVNEIIELTAGATEHAKGAKANGDASAVKVSNLLARAK 146
6HIT_1|Chains      ------KHGITI----MGGVYDAVGKIDDLKGGL---------------LSLSELHAFM- 92
1IRD_1|Chain       ------GHGKKV----ADALTNAVAHVDDMPNAL---------------SALSDLHAHK- 90
5HU6_1|Chain       ------GHGKKV----ADALTNAVAHVDDMPNAL---------------SALSDLHAHK- 90
4MQC_1|Chain       ------GHGKKM----ADALTNAVAHVDDMPNAL---------------SALSDLHAHK- 90
                           :.          : . :  :
```

RECAP

Third choice: Henikoff weights

Central idea:

Averaged over multiple-alignment columns, a sequence this is similar to others will tend to have many letters in common with those sequences.

Method:

i) For each column, divide a total weight of 1 evenly among the letter types that occur at that position, and then divide the weight assigned to each letter type evenly among the sequences that have that letter.

ii) For each sequence, sum its weights from all positions, and normalize.

Example:

Sequences	Calculation	Weight
G C G T T A G C	¼ + ⅓ + ⅓ + ¼ + ¼ + ⅓ + ¼ + ½ = 2½	0.31250
G A G T T G G A	¼ + ⅓ + ⅓ + ¼ + ¼ + ⅓ + ¼ + ¼ = 2¼	0.28125
C G G A C T A A	½ + ⅓ + ⅓ + ½ + ½ + ⅓ + ½ + ¼ = 3¼	0.40625

Henikoff, S. & Henikoff, J.G. (1994) "Position-based sequence weights." *J. Mol. Biol.* **243**:574-578.

GYVGS
GFDGF
GYDGF
GYQGG

Position-based sequence weights for the alignment in Table 2A

A. Position-based residue weights

Residue	Position				
	1	2	3	4	5
G	1/(1*4)			1/(1*4)	1/(3*1)
Y		1/(2*3)			
F		1/(2*1)			1/(3*2)
V			1/(3*1)		
D			1/(3*2)		
Q			1/(3*1)		
S					1/(3*1)

B. Position-based sequence weights

Sequence	Position						
	1	2	3	4	5	Total	Normalized
GYVGS	1/(1*4)	1/(2*3)	1/(3*1)	1/(1*4)	1/(3*1)	4/3	0.267
GFDGF	1/(1*4)	1/(2*1)	1/(3*2)	1/(1*4)	1/(3*2)	4/3	0.267
GYDGF	1/(1*4)	1/(2*3)	1/(3*2)	1/(1*4)	1/(3*2)	3/3	0.200
GYQGG	1/(1*4)	1/(2*3)	1/(3*1)	1/(1*4)	1/(3*1)	4/3	0.267
Total	1	1	1	1	1	5	1.001

Each residue in each position is assigned a weight equal to $1/(r*s)$ where r is the number of different residues in the position and s is the number of times the particular residue appears in the position. The position-based weights are then added for each position in each sequence.

Advantages:

Very fast and simple.

Independent of sequence order.

Uses all information.

Disadvantages:

Ad hoc: no objective function to optimize.

Exact duplication of a sequence does *not* halve its weight. *Why?*

Digression: The Effective Number of Independent Sequences in a Multiple Alignment

Why is this number relevant?

The problem: What, for example, should be the score for aligning a valine to a column of five leucines?

```
...GEALGRLLVVYPWTQ...
...GEALGRLLIVYPWTQ...
...GETLGRLLVVYPWTQ...
...GKALGRLLIVYPWTQ...
...GEALGRLLVVYPWTQ...
..........V..........
```

Here, the sequences in the multiple alignment are virtually identical. There is little reason to score the alignment much differently than that of valine to a single leucine (BLOSUM-62: +1).

```
...GEALGRLLVVYPWTQ...
...KECFTKFLSAHHDIA...
...VVFYTSILEKAPAAK...
...VDILVKFLTGTPAAQ...
...AEGLERTLHSFPTTK...
..........V..........
```

Here, the sequences are very different, providing good evidence that a leucine is highly favored at this position. Thus, the score for aligning a valine should probably be negative.

Work out

12AS_A: AEKAVQVKVKAL
11AS_A: AEKAVQVKVKAL
1IND_H: PEKRLEWVATTL
1EZG_A: NSQHCVKANTCT

Compute the Henikoff weight for the above sequences.

Compute a position specific scoring matrix.

Workout

- Write down the steps to compute the Henikoff weights of a number of sequences taken as input.

Pairwise Sequence Alignment

Global

```
FTFTALILLAVAV
F—TAL-LLA-AV
```

Local

```
FTFTALILL-AVAV
--FTAL-LLAAV--
```

Local Pairwise Sequence Alignment

- Smith-Waterman

Seq1: ACACACTA
Seq2: AGCACACA
S(match) = +2
S(mismatch) = *w(-,b)* = *w(a,-)* = -1

$$H = \begin{pmatrix} & - & A & C & A & C & A & C & T & A \\ - & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ A & 0 & 2 & 1 & 2 & 1 & 2 & 1 & 0 & 2 \\ G & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\ C & 0 & 0 & 3 & 2 & 3 & 2 & 3 & 2 & 1 \\ A & 0 & 2 & 2 & 5 & 4 & 5 & 4 & 3 & 4 \\ C & 0 & 1 & 4 & 4 & 7 & 6 & 7 & 6 & 5 \\ A & 0 & 2 & 3 & 6 & 6 & 9 & 8 & 7 & 8 \\ C & 0 & 1 & 4 & 5 & 8 & 8 & 11 & 10 & 9 \\ A & 0 & 2 & 3 & 6 & 7 & 10 & 10 & 10 & 12 \end{pmatrix} \quad T = \begin{pmatrix} & - & A & C & A & C & A & C & T & A \\ - & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ A & 0 & \nwarrow & \leftarrow & \nwarrow & \leftarrow & \nwarrow & \leftarrow & \leftarrow & \nwarrow \\ G & 0 & \uparrow & \nwarrow & \uparrow & \nwarrow & \uparrow & \nwarrow & \nwarrow & \uparrow \\ C & 0 & \uparrow & \nwarrow & \leftarrow & \nwarrow & \leftarrow & \nwarrow & \leftarrow & \leftarrow \\ A & 0 & \nwarrow & \uparrow & \nwarrow & \leftarrow & \nwarrow & \leftarrow & \leftarrow & \nwarrow \\ C & 0 & \uparrow & \nwarrow & \uparrow & \nwarrow & \leftarrow & \nwarrow & \leftarrow & \leftarrow \\ A & 0 & \nwarrow & \uparrow & \nwarrow & \uparrow & \nwarrow & \leftarrow & \leftarrow & \nwarrow \\ C & 0 & \uparrow & \nwarrow & \uparrow & \nwarrow & \uparrow & \nwarrow & \leftarrow & \leftarrow \\ A & 0 & \nwarrow & \uparrow & \nwarrow & \uparrow & \nwarrow & \uparrow & \nwarrow & \nwarrow \end{pmatrix}$$

Local pairwise alignment

- Align

S = TAATATATTTAT

T = AAGCGAATAATATATTTATACTCAGATTATTGCGCG

Local pairwise alignment

- Initial Seed

```
            TAT
            |||
AAGCGAATAATATATTTATACTCAGATTATTGCGCG
```

- Alignment by expansion of seed

```
       TAATATATTTAT
       ||||||||||||
AAGCGAATAATATATTTATACTCAGATTATTGCGCG
```

An example

- **Input:**

  S_q= AKLMAATCD

  S_i= ….ALPQRKLMMAKLPPRTLQ…..

  Window size $w = 4$

  Threshold $T = 3$ for determining seed points

- **Method:**

  1. Generating subsequences of length 4 ($w = 4$) from target string S_q.

- AKLMAATCD
- AKLM
- KLMA
- LMAA
- MAAT
- AATC
- ATCD

Considering threshold $T=3$, sequences that are considered valid for subsequence KLMA

Sequence	Score
KLMA	4
K*MA	3
KL*A	3
KLM*	3
* indicates a wild card character and can be any alphabet	

- Identifying valid entries in string S_i corresponding to seed KLMA

1100 = 2 X

KLMA

….ALPQ**RKLMMAKLP**PRTLQ…..

KLMA

1110 = 3 ✓

Perform all such string matching's for all such seed points contained extended regions and report the segment with the highest score in string S_i.

What is BLAST?

- **B**asic **L**ocal **A**lignment **S**earch **T**ool

- Calculates similarity for biological sequences.

- Produces local alignments: only a portion of each sequence must be aligned.

- Uses statistical theory to determine if a match might have occurred by chance.

BLAST is a heuristic

- A lookup table is made of all the “words” (short subsequences) in the query sequence. In many types of searches “neighboring” words are included.

- The database is scanned for matching words (“hot spots”).

- Gapped and un-gapped extensions are initiated from these matches.

BLAST method

- It is an alignment heuristic that determines “local alignments” between a query and a database. It uses an approximation of the Smith-Waterman algorithm.

- BLAST consists of two components: a search algorithm and computation of the statistical significance of solutions.

- BLAST uses a heuristic method to find the highest scoring alignment between the query sequence and the search set sequence.

BLAST terminology

- Definition

Let *q* be the query and *d* the database. A segment is simply a substring *s* of *q* or *d*.

A segment-pair (*s*, *t*) (or hit) consists of two segments, one in *q* and one *d*, of the same length.

BLAST terminology

- Example

```
V A L L A R
P A M M A R
```

- We think of *s* and *t* as being aligned without gaps and score this alignment using a substitution score matrix, e.g. BLOSUM or PAM in the case of protein sequences.

- The alignment score for (*s*, *t*) is denoted by $\sigma(s, t)$.

BLAST terminology

- A locally maximal segment pair (LMSP) is any segment pair (*s*, *t*) whose score cannot be improved by shortening or extending the segment pair.

- A maximum segment pair (MSP) is any segment pair (*s*, *t*) of maximal alignment score $\sigma(s, t)$.

- Given a cutoff score *S*, a segment pair (*s*, *t*) is called a high-scoring segment pair (HSP), if it is locally maximal and $\sigma(s,t) \geq S$.

- Finally, a word is simply a short substring of fixed length *w*.

The BLAST algorithm

- **Goal:** Find all HSPs for a given cut-off score.

- Given three parameters, i.e. a word size *w*, a word similarity threshold *T* and a minimum cut-off score *S*. Then we are looking for a segment pair with a score of at least *S* that contains at least one word pair of length *w* with score at least *T*.

The BLAST algorithm

- **Preprocessing:** Of the query sequence *q* first all words of length *w* are generated. Then a list of all *w*-mers of length *w* over the alphabet $\sum$ that have similarity > *T* to some word in the query sequence q is generated.

Example

For the query sequence `RQCSAGW` the list of words of length $w = 2$ with a score $T > 8$ using the BLOSUM62 matrix are:

word	$2-mer$ with score > 8
RQ	RQ
QC	QC, RC, EC, NC, DC, KC, MC, SC
CS	CS,CA,CN,CD,CQ,CE,CG,CK,CT
SA	-
AG	AG
GW	GW,AW,RW,NW,DW,QW,EW,HW,KW,PW,SW,TW,WW

The BLAST algorithm

1. **Localization of the hits:** The database sequence d is scanned for all hits t of w-mers s in the list, and the position of the hit is saved.
2. **Detection of hits:** First all pairs of hits are searched that have a distance of at most A (think of them lying on the same diagonal in the matrix of the SW-algorithm).
3. **Extension to HSPs:** Each such *seed* (s, t) is extended in both directions until its score $\sigma(s, t)$ cannot be enlarged (LMSP). Then all best extensions are reported that have score $\geq S$, these are the HSPs. Originally the extension did not include gaps, the modern BLAST2 algorithm allows insertion of gaps.

The BLAST algorithm

- The list L of all words of length w that have similarity $> T$ to some word in the query sequence q can be produced in $O(|L|)$ time.
- These are placed in a "keyword tree" and then, for each word in the tree, all exact locations of the word in the database d are detected in time linear to the length of d.
- As an alternative to storing the words in a tree, a finite-state machine can be used, which Altschul et al. found to have the faster implementation.

The BLAST algorithm

Use of seeds of length *w* and the termination of extensions with fading scores (**score dropoff threshold X**) are both steps that speed up the algorithm.
Recent improvements (BLAST 2.0):

- Two word hits must be found within a window of *A* residues.
- Explicit treatment of gaps.
- Position-specific iterative BLAST (PSI-BLAST).

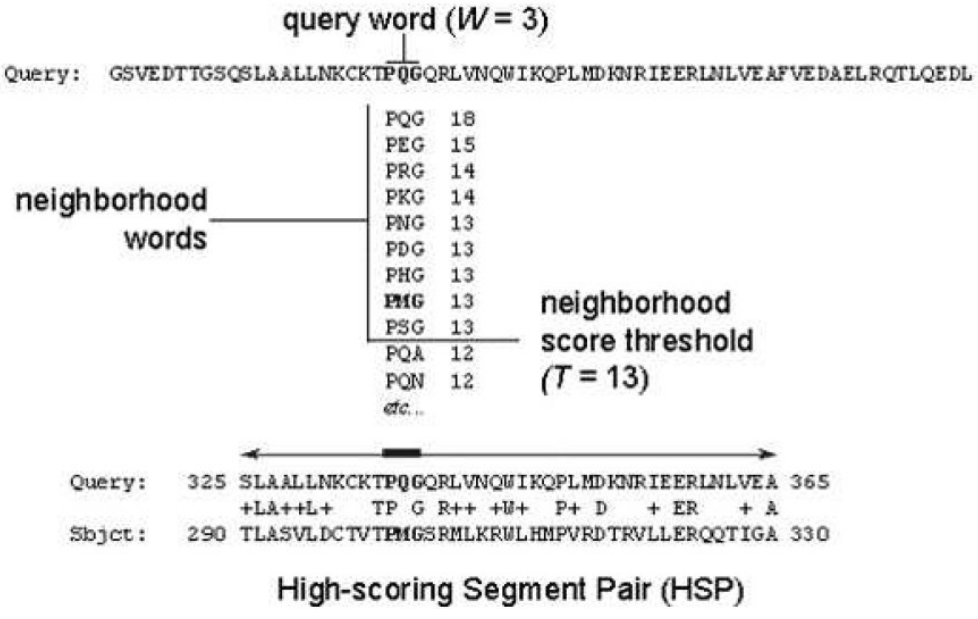
The BLAST algorithm for DNA

For *DNA* sequences, BLAST operates as follows:

- The list of all words of length *w* in the query sequence *q* is generated. In practice, $w = 12$ for DNA.
- The database *d* is scanned for all hits of words in this list. Blast uses a two-bit encoding for DNA. This saves space and also search time, as four bases are encoded per byte.

Note that the "T" parameter dictates the speed and sensitivity of the search.

The BLAST search algorithm



Statistical Significance of HSP

- **Problem:** ***Given an HSP (s,t) with score σ(s,t). How significant is this match (i.e., local alignment)?***

  Given the scoring matrix $S(a, b)$, the expected score for aligning a random pair of amino acid is required to be negative:

$$E = \sum_{a,b \in \Sigma} p_a p_b S(a, b) < 0$$

  The sum of a large number of independent identically distributed (i.i.d) random variables tends to a normal distribution. The maximum of a large number of i.i.d. random variables tends to an extreme value distribution as we will see

Statistical Significance of HSP

HSP scores are characterized by two parameters, K and λ. The parameters K and λ depend on the background probabilities of the symbols and on the employed scoring matrix. λ is the unique value for y that satisfies the equation

$$\sum_{a,b\in\Sigma} p_a p_b e^{S(a,b)y} = 1$$

K and λ are scaling-factors for the search space and for the scoring scheme, respectively.

The number of random HSPs (s, t) with $\sigma(s, t) \geq S$ can be described by a Poisson distribution with parameter $v = Kmne^{-\lambda S}$. The number of HSPs with score $\geq S$ that we *expect* to see due to chance is then the parameter v, also called the *E-value*:

$$E(\texttt{HSPs with score} \geq S) = Kmne^{-\lambda S}$$

BLAST Statistics

- **Score:**
  - A statistical conversion of the score derived by summing using the substitution matrix.
- **Expect (e) value:**
  - Function of the S value and the database size
  - *An e value of 1:* One alignment using a query of this size will by chance produce a S score of this value in a database of this size
  - *e value of –10 (=1x10^{-10}):* Unlikely that random chance lead to this current alignment compared to an alignment with an e value of 1
  - Expect value is specific to a database of a certain size . Thus it may change later because of change in database size.
- **Rules of thumb:**
  - **E value of –30 or less:** Sequences are homologous
  - **E values of –5:** Often considered significant enough when annotating a genome

BLAST output

- Pair-wise report
- Query-anchored report
- Hit-table
- Tax BLAST
- Abstract Syntax Notation 1
- XML

BLAST family of programs

The BLAST family of programs allows all combinations of DNA or protein query sequences with searches against DNA or protein databases:

- Protein-protein (blastp): compares an amino acid sequence against a protein sequence database.
- Nucl.-nucl (blastn): compares a nucleotide query sequence against a nucleotide sequence database (in general optimized for speed, not sensitivity).
- Translated nucl.-protein (blastx): compares the six-frame conceptual translation products of a nucleotide query against a protein sequence database.
- Protein-translated nucl (tblastn): compares a protein query sequence against a sequence database dynamically translated in all six reading frames (useful for searching proteins against EST's).
- Translated nucl-translated nucl. (tblastx): compares the six frame translation of a nucleotide query sequence against the six-frame translations of a nucleotide sequence database.

BLAST webserver

- BLAST
  - http://blast.ncbi.nlm.nih.gov/Blast.cgi
  - http://blast.ncbi.nlm.nih.gov/Blast.cgi?PROGRAM=blastp&PAGE_TYPE=BlastSearch&LINK_LOC=blasthome

Work Out

- **Problem Statement**

  Given a query sequence S_q and a target sequence S_i (such that $|S_q| \ll |S_i|$) find an optimal alignment and alignment score of S_q with S_i.

  - *Additional information/input:* For a window size (*w*) 4 the minimum score (*T*) value is 3. First locate such window and expand on both the sides.