Pairwise Sequence Alignment

contd….

Heuristic Alignment Algorithms

Dynamic programming algorithms though very sensitive, are **not the fastest** available sequence alignment methods

- time complexity ~ $O(nm)$, and in many cases **speed is the issue**, e.g., database searches

Protein database contain ~100M residues, this requires ~10^{11} matrix cells to be evaluated to search complete database for a query of length 1000; DNA Dbs are even larger

At 10M matrix cells per second this would be 10^4 secs, or **~ 3 hrs** for a single search

Heuristic Alignment Algorithms

For homology-based gene identification and annotation purposes one routinely screens genes against a database of proteins.

- need for **algorithms faster** than pure DP

Approx. methods can detect close relationships well and quickly but fail to identify very distant relationships.

Heuristic Alignment Algorithms

Goal of such methods - to search as small a fraction as possible of the cells in the DP matrix, while still looking at all high scoring alignments.

A typical approximation approach - take a small integer *k*, and determine all instances of each *k*-tuple of residues in the probe sequence that occur in any sequence in the database

A candidate sequence in the database would contain many matching *k*-tuples, with equivalent spacing in probe and candidate sequences.

- perform DP only for probable homologies.

Heuristic Alignment Algorithms

For selected candidate sequences, approximate optimal alignment calculations are carried out, with the “time” and “space” saving restriction

- paths through the matrix considered are restricted to bands around the diagonals containing the matching *k*-tuples.

e.g. of Heuristic approaches: BLAST, FASTA

Another computational resource that can limit dynamic programming alignment is memory usage

- memory requirement for $F(i, j) \sim O(nm)$, product of sequence lengths

For two protein sequences (few hundred residues long) - manageable on a desktop computers

But if one or both the sequences are genomic DNA sequences (hundreds of thousands of bases long), required memory for the full matrix can exceed machine's physical capacity.

Assignment

Find out the size of protein database, UniProt, and nucleotide database, GenBank.

Compute No. of matrix cells to be computed using DP for:

(1) searching protein database, UniProt, and nucleotide database, GenBank and the time required assuming query sequence of length 1000 bases.

(2) Comparing Human Chr 1 ~249Mbp with a query sequence of 1000 bases using DP, and comparing it with Chr 1 of Mouse (~195Mbp)? What is the space requirement in the two cases?

Use computation time as 10M matrix cells per second (or operation time of your machine)

Linear Space Alignments

Fortunately, situation is better with memory than speed:

- there are techniques that give the optimal alignment in limited memory, of order $n + m$ rather than nm, but comes with a cost of doubling of time.

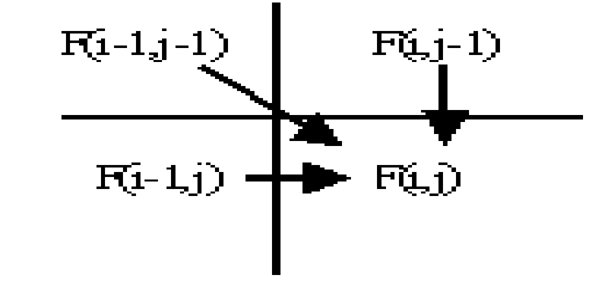
These methods are commonly referred to as linear space alignment methods.

Linear Space Alignments

If only the maximal score is needed, the problem is very simple:

Recurrence relation for *F*(*i*, *j*) depends only on entries one row back - one can throw away rows that are further than one back from the current point.

For local alignment, the maximum score in the whole matrix is required - easy to keep track of the maximum value as the matrix is being built.



F(i-1,j-1)

F(i,j-1)

F(i-1,j)

F(i,j)

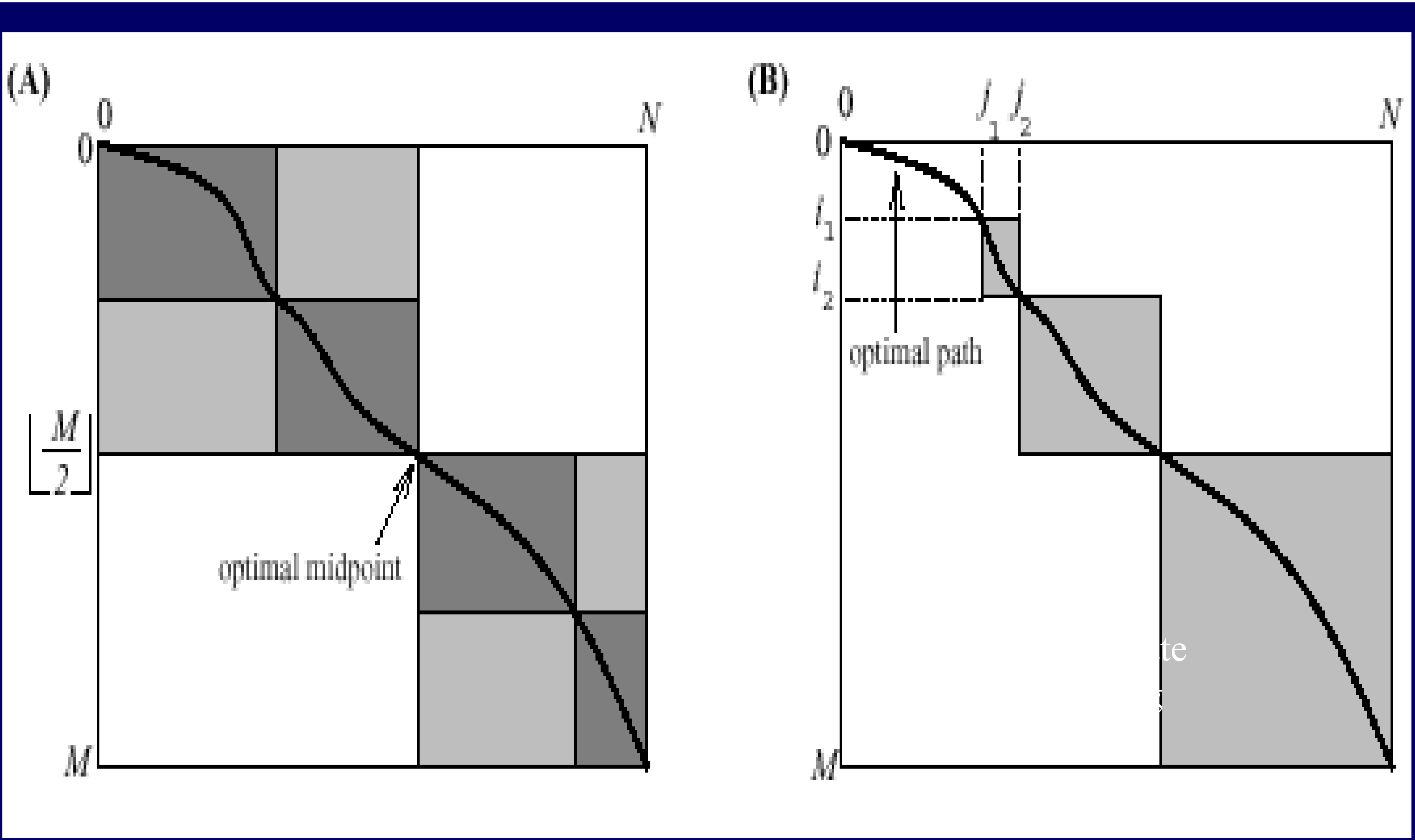
Linear Space Alignments

While this gives us the score, it will not find the Alignment!

If we throw away rows to avoid *O*(*nm*) storage, then we also loose the traceback pointers.

An approach used to obtain the alignment uses the principle of divide and conquer.

Snapshot of Execution of Hirschberg’s algorithm



Linear Space Alignments

Steps:

- **Let $u = m/2$ (integer part)**
- **Identify a v such that the cell (u, v) is on the optimal alignment, i.e., v is the column where the alignment crosses the $u = m/2$ row of the matrix**

- this splits the DP problem into two subproblems: from $(0, 0)$ to (u, v), & from (u, v) to (m, n)

- full alignment will be concatenation of the optimal alignments for these two separate submatrices

Linear Space Alignments

- This is done **recursively**, by successively halving each region, until sequences of zero length are being aligned, or alternatively, sequences are short enough & standard $O(nm)$ alignment and traceback method can be used.

But how do we find *v*?

- By combining the results of "forward" and "backward" DP passes at row *u*, for **each point** along the middle row, i.e.,

- optimal score from (0, 0) to each point in row *u* and optimal score from that point to (*m*, *n*).

Linear Space Alignments

- Adding these numbers gives the optimal score over all paths from (0, 0) to (m, n) that pass through all the points in row u.
- A sweep along the middle row, checking these sums, determines a point ($m/2$, v) where an optimal path crosses the middle row.
- This is then done recursively.

Linear Space Alignments

EMBOSS Programs:

matcher: linear-space version of local alignment algorithm by M. S. Waterman & M. Eggert

stretcher: global alignment algorithm using linear space

Database Search - Need for Heuristics

- Dynamic prog'g algorithms are computer intensive & time consuming ~ O(nm)
- Searching large databases using these algorithms is not feasible

e.g., if the time to compare two 1kb sequences is ~ milliseconds, comparison of 1kb sequence with the human genome will take 3 hours!

Database Search - Need for Heuristics

- Solution 1: implement the algorithm in hardware
  – expensive

  SW algorithm has been implemented on Hybrid Core machines

- Solution 2: distribute the job to several processors to do in parallel mode

  – gets expensive as no. of processors ~ 1000

- Solution 3: use heuristics – gives approx. solutions

Heuristic Algorithms

Basic idea: first locate high-scoring short stretches and then extend them

Two well known programs:

~ 50 times faster than DP

- FASTA: Fast Alignment Tool
- BLAST: Basic Local Alignment Search Tool

Both find high scoring local alignments between a query sequence and a target database

Heuristic algorithms aim at speeding up the database search at the price of possibly missing the best scoring alignment

FASTA

- Proposed by Pearson & Lipman, 1988
- Compares sequences pairwise
- Heuristics: A good alignment will have exact matching subsequences
- Gain on speed but loss of sensitivity
- Best suited for global alignment
- Works better with DNA sequences

http://www.ebi.ac.uk/Tools/sss/fasta

FASTA: The algorithm

- Based on the logic of the dot matrix method
- View sequences as sequences of short words (*k*-tuple)

Motivation:

- Good alignments should contain many exact matches
- Hashing can find exact matches in O(n) time
- Diagonals can be formed from exact matches quickly and sorted
- Apply more precise alignment to small search space at the end

FASTA: The algorithm

- Look for all *k*-tuple matches between query & database sequences
  - For DNA *k* = 2-6 (default 6)
  - For proteins, *k* =1,2 (default 2)
- This is done by a lookup table, or hash
- For each *k*-tuple the database is pre-processed to identify its positions
- Query is scanned – for each *k*-tuple in the query, look-up the table/hash to identify matches in the database

Lookup method for finding an alignment

Position 1 2 3 4 5 6 7 8 9 10 11

Seq A n c s p t a . . . . .

Seq B . . . . . . a c s p r k

Amino acid	Position in		Offset pos A – pos B
	Protein A	Protein B	
a	6	6	0
c	2	7	-5
k	-	11	
n	1	-	
p	4	9	-5
r	-	10	
s	3	8	-5
t	5	-	

Dot matrix method

	n	c	s	p	t	a				
a						●				
c		●								
s			●							
p				●						
r										
k										

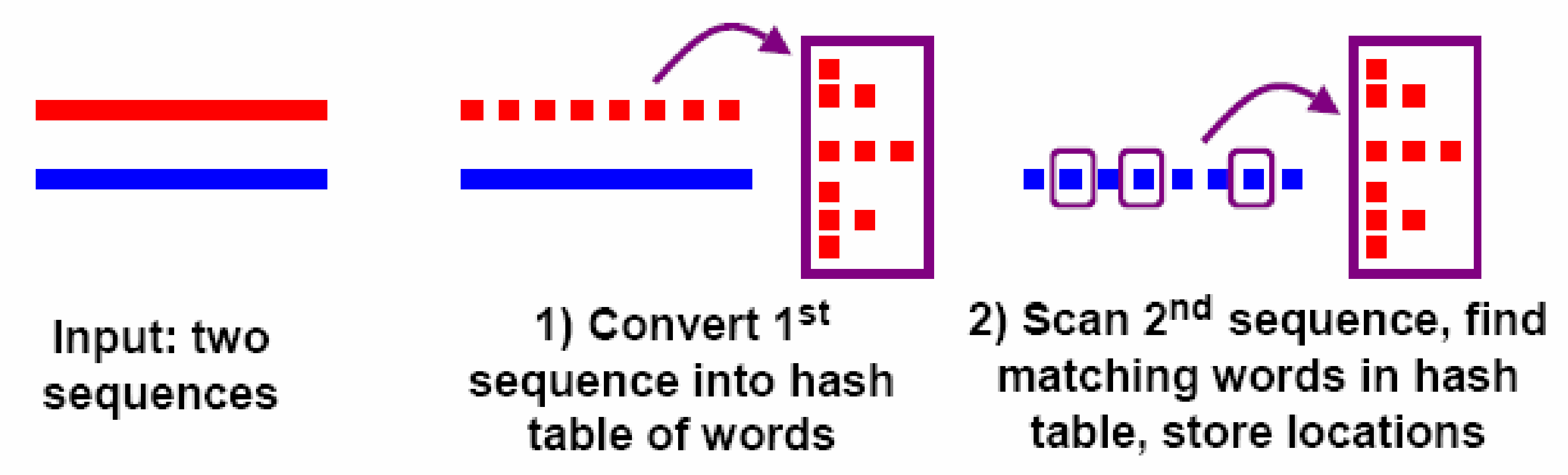
$\sim O(n)$

Possible alignment:

```
n c s p t a
  | | |
a c s p r k
```

FASTA: The algorithm

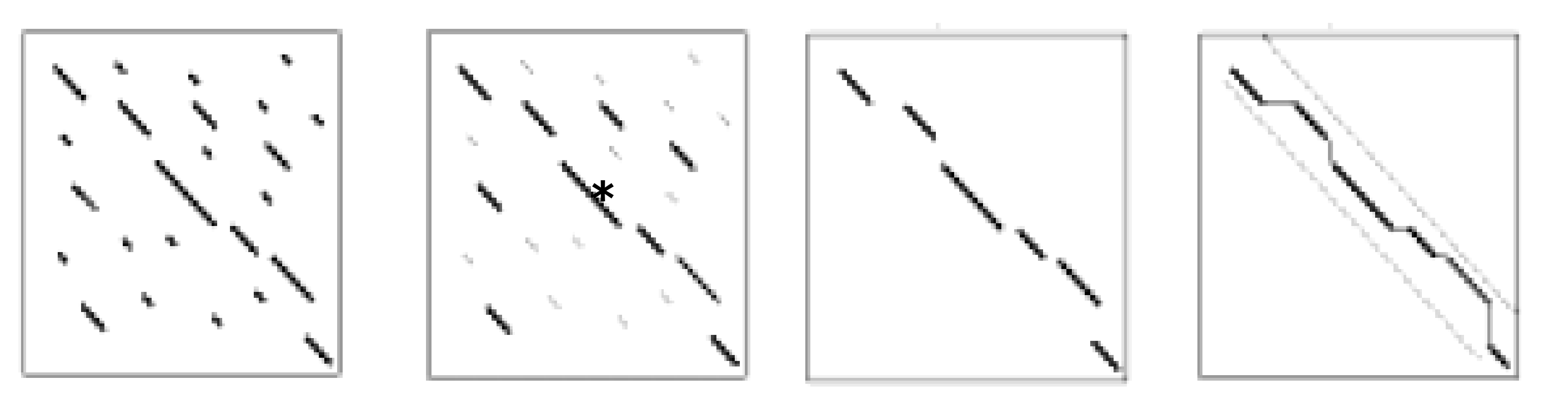
- Look for adjacent hot spots (substrings of exact matches) on the same diagonal
  - join them to form larger segments
  - space between hot spots gets –ve score
- Find 10 highest scoring diagonals
- Find the best diagonal run and filter out low scoring runs – by recomputing the score for each diagonal using scoring matrix and threshold score
- Combine close diagonal runs, including indels
- Compute alternative local alignments, using a band around the best diagonal run
- Finally use DP to align the query against best ranking resulting sequences



Input: two sequences

1) Convert 1st sequence into hash table of words

2) Scan 2nd sequence, find matching words in hash table, store locations



- Identify regions of identity (hot spots),
- Scan regions using a scoring matrix & save the best initial regions, regions with score < threshold are discarded, * marks best region
- Optimally join initial regions with score > threshold,
- Recalculate an optimised alignment around the highest scoring region

BLAST

- Basic Local Alignment Search Tool

  - proposed by Altshul *et al*, 1990, developed by NCBI

- The algorithm has been designed to balance speed and sensitivity in aligning sequences
- Designed to work best for local ungapped alignments – now incorporates gaps
- Website: http://www.ncbi.nlm.nih.gov/blast/

BLAST: Fastest alignment tool

Motivation:

- Good alignments should contain many close matches
- Statistics can determine which matches are significant
  - Much more sensitive than % identity
- Extending matches in both directions finds alignment
  - Yields high-scoring/maximum segment pairs (HSP/MSP)

BLAST: Fastest alignment tool

- View sequences as sequence of short words (*k*-tuples)
  - DNA: 7, 11, 15 (default 11),
  - protein: 2, 3, 6 (default 6)
- Create hash table of neighborhood (closely-matching) words
- Use statistics to set threshold for "closeness"

BLAST: Fastest alignment tool

- First look for short matching segments
- Choose matching segments above a threshold score, hits
- Overlapping hits form a larger segment
- Large segment extended in either direction if its score is above a threshold
- Extension of alignment continues until the score falls below the drop-off threshold from maximum value

PQG - 18
PEG - 15
PRG - 14
PSG - 13
PQA - 12

BLAST: Example

- Sequence: ASTNC
- Word ASTN has score of 9 for exact match using PAM250 scoring matrix
- If threshold score is 17, ASTN will not be used for querying the database
- STNC has score 19 for exact match using PAM250, and will be used

Note: not all exact matches are used if the score is below the threshold.

BLAST: The Algorithm

- Look for high scoring segments pairs (HSPs)
  - looks for similar instead of identical pairs
  - uses scoring matrix to score aligned pairs
  - only those pairs which score above a threshold are considered for extension
  - the extension is without gaps

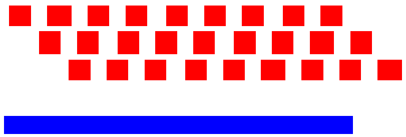
```
L P   P Q G   L L   query
M P   P E G   L L   Db seq
      <word>
←             →
Ext. to Left  Ext. to Rt
```

HSP score
= 9 + 15 + 8
= 32

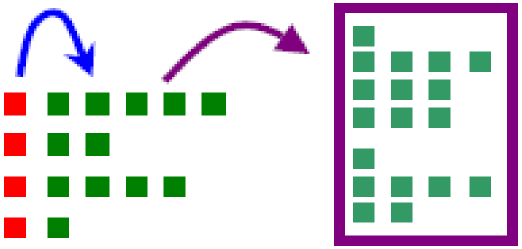
BLAST: The Algorithm

- Ungapped extension of HSPs with scores > *T* (threshold) - identifies maximal segment pairs (MSPs)
- Extension continues until the score drops below a threshold drop-off from the maximum score encountered
- Highest scoring segment pair, MSP (Maximal scoring Segment Pair) identified
- For gapped alignment, BLAST uses the same strategy as FASTA of joining segments on different diagonals.

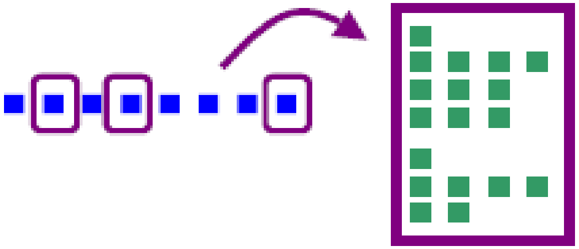
BLAST: Algorithm



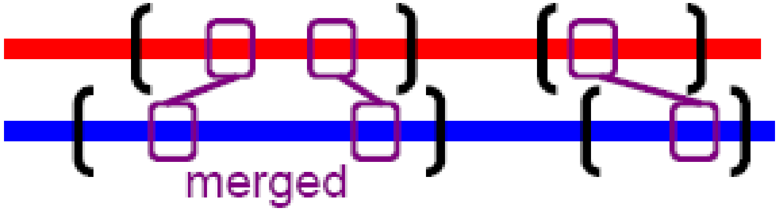
1) Convert 1st sequence into words (using all frames for given word size)



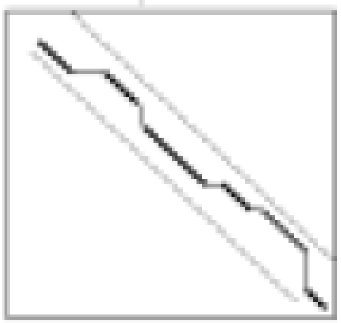
2) Calculate for each word list of "neighborhood" words (scoring threshold T) and enter in dictionary



3) Scan 2nd sequence, find matching words in dictionary, store locations



4) For each match, extend alignment in both directions while score above threshold S, merge segments

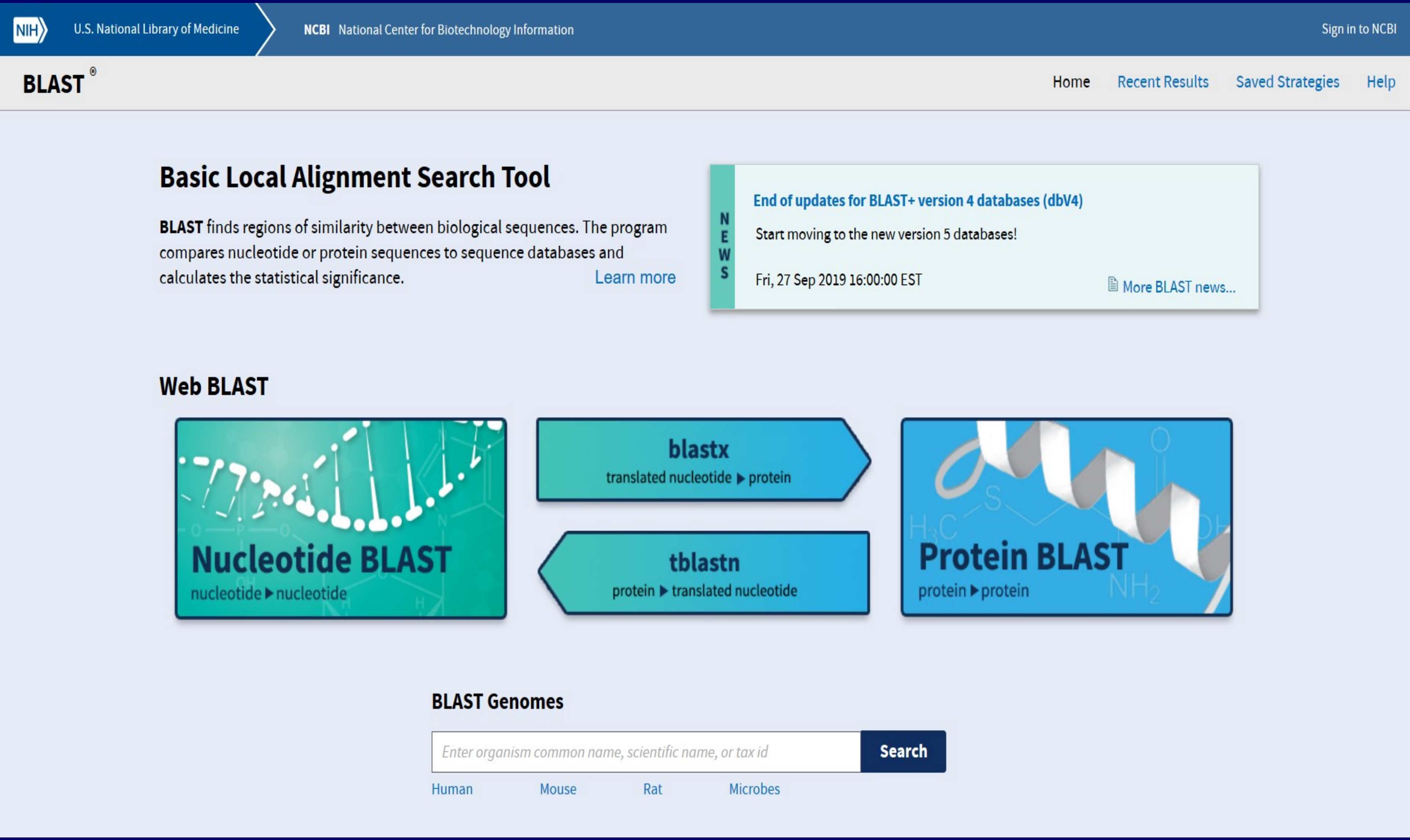


5) Align best segments using dynamic programming, report statistically significant matches

Selecting a BLAST program

BLAST is suite of programs, the most popular being

- blastn: compares nucleotide sequence with nucleotide database
- blastp: compares protein sequence with a protein sequence database



NIH U.S. National Library of Medicine | NCBI National Center for Biotechnology Information | Sign in to NCBI

BLAST®

Home | Recent Results | Saved Strategies | Help

Basic Local Alignment Search Tool

BLAST finds regions of similarity between biological sequences. The program compares nucleotide or protein sequences to sequence databases and calculates the statistical significance. Learn more

NEWS

End of updates for BLAST+ version 4 databases (dbV4)

Start moving to the new version 5 databases!

Fri, 27 Sep 2019 16:00:00 EST

More BLAST news...

Web BLAST

Nucleotide BLAST
nucleotide ▶ nucleotide

blastx
translated nucleotide ▶ protein

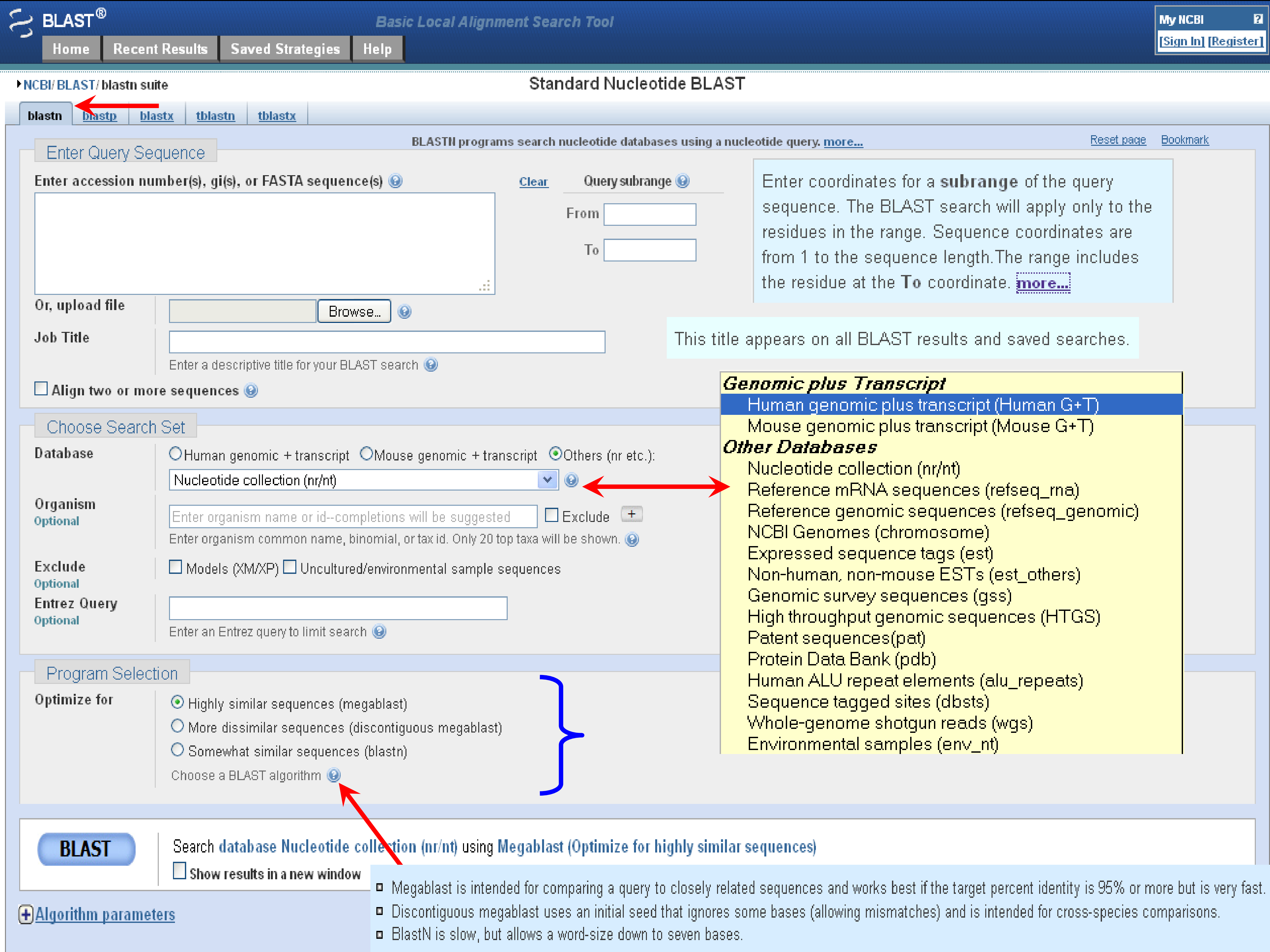
tblastn
protein ▶ translated nucleotide

Protein BLAST
protein ▶ protein

BLAST Genomes

Enter organism common name, scientific name, or tax id | Search

Human | Mouse | Rat | Microbes



Standard Nucleotide BLAST

BLASTN programs search nucleotide databases using a nucleotide query. more...

Enter accession number(s), gi(s), or FASTA sequence(s)

Query subrange
From
To

Enter coordinates for a **subrange** of the query sequence. The BLAST search will apply only to the residues in the range. Sequence coordinates are from 1 to the sequence length.The range includes the residue at the **To** coordinate. more...

Or, upload file

Job Title

Enter a descriptive title for your BLAST search

This title appears on all BLAST results and saved searches.

□ Align two or more sequences

Database: ○ Human genomic + transcript ○ Mouse genomic + transcript ● Others (nr etc.):

Nucleotide collection (nr/nt)

Genomic plus Transcript
- Human genomic plus transcript (Human G+T)
- Mouse genomic plus transcript (Mouse G+T)

Other Databases
- Nucleotide collection (nr/nt)
- Reference mRNA sequences (refseq_rna)
- Reference genomic sequences (refseq_genomic)
- NCBI Genomes (chromosome)
- Expressed sequence tags (est)
- Non-human, non-mouse ESTs (est_others)
- Genomic survey sequences (gss)
- High throughput genomic sequences (HTGS)
- Patent sequences(pat)
- Protein Data Bank (pdb)
- Human ALU repeat elements (alu_repeats)
- Sequence tagged sites (dbsts)
- Whole-genome shotgun reads (wgs)
- Environmental samples (env_nt)

Organism
Optional
Enter organism name or id--completions will be suggested □ Exclude +
Enter organism common name, binomial, or tax id. Only 20 top taxa will be shown.

Exclude
Optional
□ Models (XM/XP) □ Uncultured/environmental sample sequences

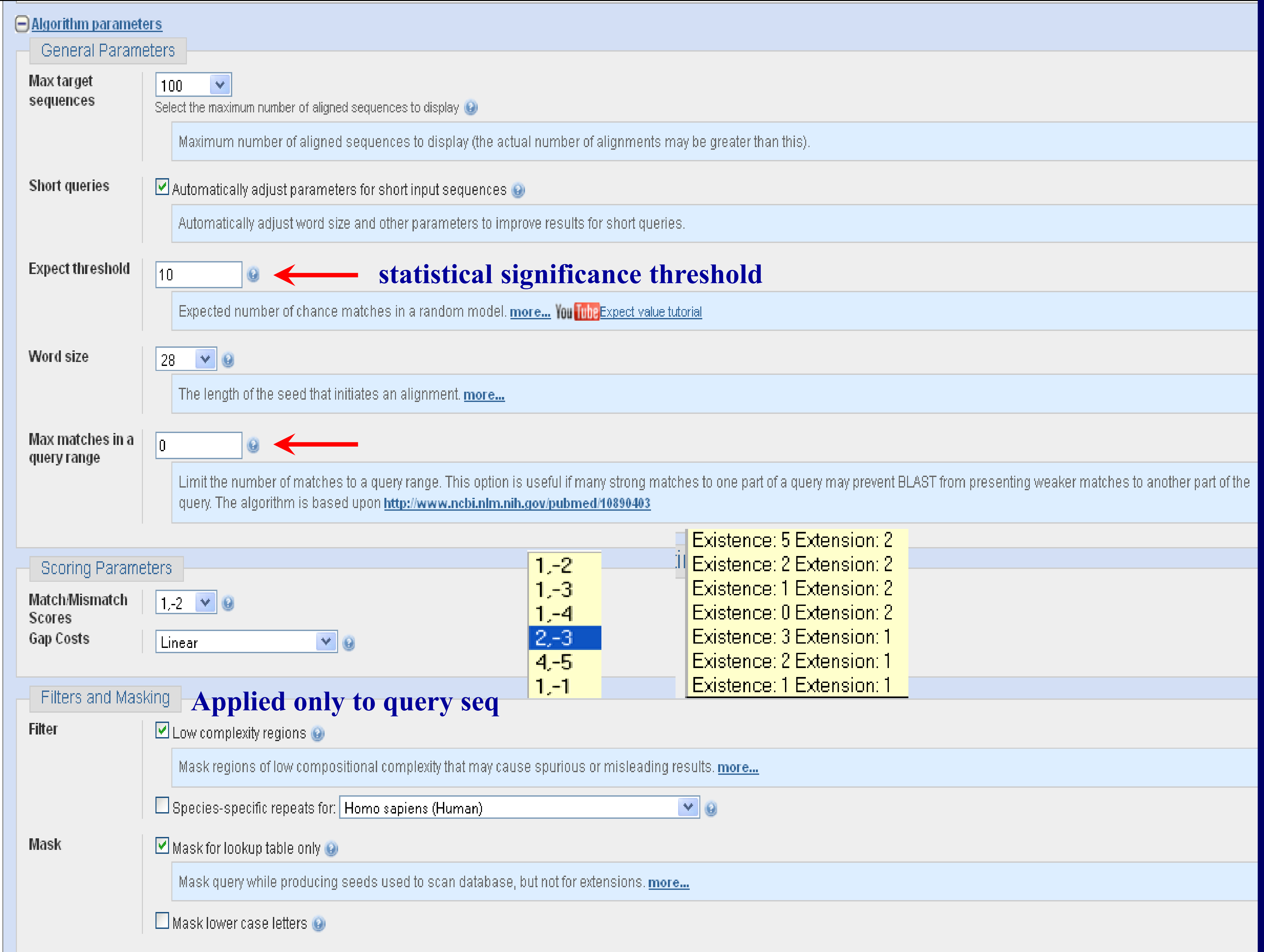
Entrez Query
Optional
Enter an Entrez query to limit search

Optimize for
● Highly similar sequences (megablast)
○ More dissimilar sequences (discontiguous megablast)
○ Somewhat similar sequences (blastn)
Choose a BLAST algorithm

Search database Nucleotide collection (nr/nt) using Megablast (Optimize for highly similar sequences)

□ Show results in a new window

- Megablast is intended for comparing a query to closely related sequences and works best if the target percent identity is 95% or more but is very fast.
- Discontiguous megablast uses an initial seed that ignores some bases (allowing mismatches) and is intended for cross-species comparisons.
- BlastN is slow, but allows a word-size down to seven bases.



⊟ Algorithm parameters

General Parameters

Max target sequences: 100

Select the maximum number of aligned sequences to display

Maximum number of aligned sequences to display (the actual number of alignments may be greater than this).

Short queries: ☑ Automatically adjust parameters for short input sequences

Automatically adjust word size and other parameters to improve results for short queries.

Expect threshold: 10 ← **statistical significance threshold**

Expected number of chance matches in a random model. more... YouTube Expect value tutorial

Word size: 28

The length of the seed that initiates an alignment. more...

Max matches in a query range: 0 ←

Limit the number of matches to a query range. This option is useful if many strong matches to one part of a query may prevent BLAST from presenting weaker matches to another part of the query. The algorithm is based upon http://www.ncbi.nlm.nih.gov/pubmed/10890403

Scoring Parameters

Match/Mismatch Scores: 1,-2

- 1,-2
- 1,-3
- 1,-4
- 2,-3
- 4,-5
- 1,-1

Gap Costs: Linear

- Existence: 5 Extension: 2
- Existence: 2 Extension: 2
- Existence: 1 Extension: 2
- Existence: 0 Extension: 2
- Existence: 3 Extension: 1
- Existence: 2 Extension: 1
- Existence: 1 Extension: 1

Filters and Masking **Applied only to query seq**

Filter: ☑ Low complexity regions

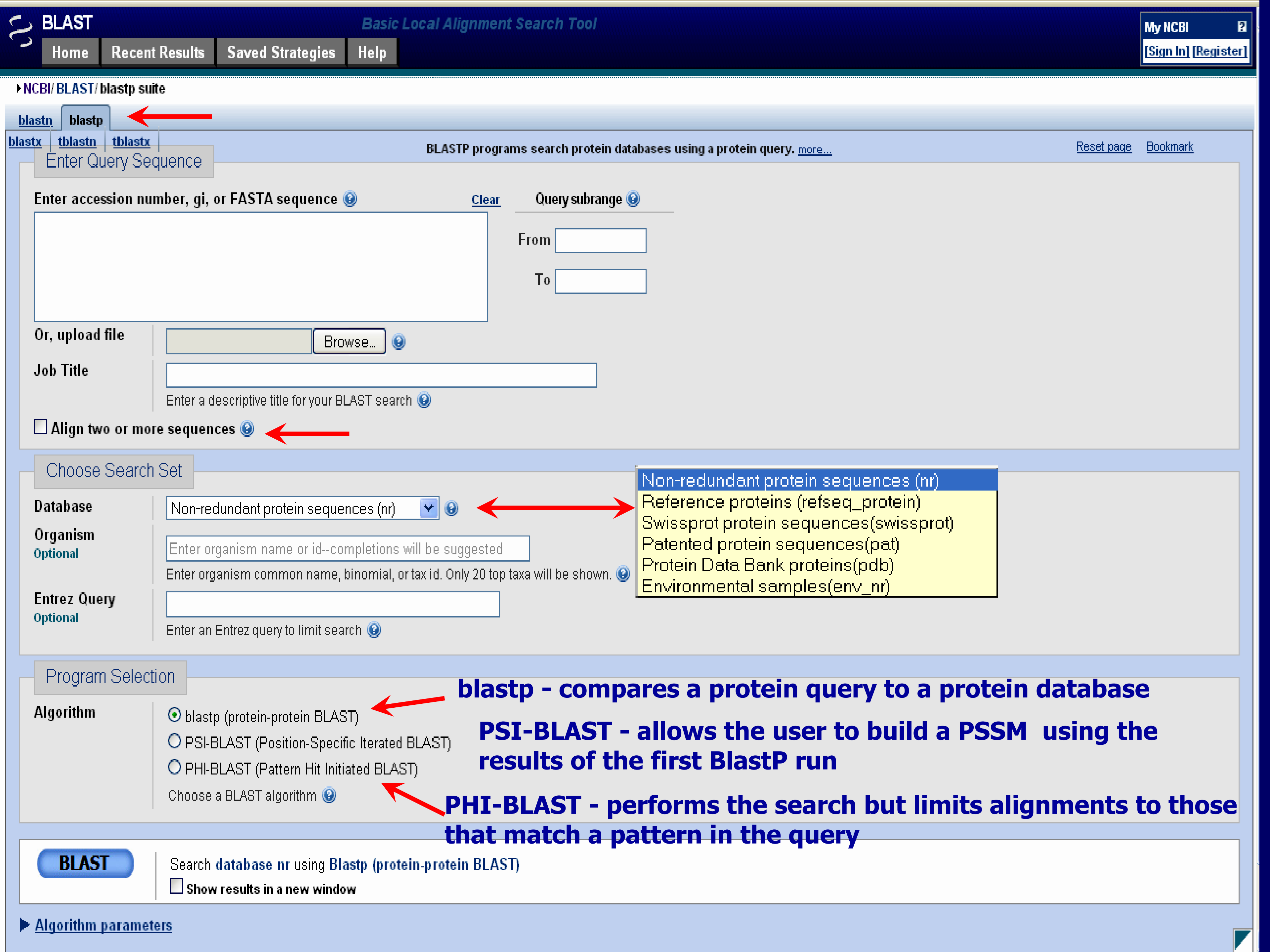
Mask regions of low compositional complexity that may cause spurious or misleading results. more...

☐ Species-specific repeats for: Homo sapiens (Human)

Mask: ☑ Mask for lookup table only

Mask query while producing seeds used to scan database, but not for extensions. more...

☐ Mask lower case letters



BLASTP programs search protein databases using a protein query. more...

Enter Query Sequence

Enter accession number, gi, or FASTA sequence Clear

Query subrange

From

To

Or, upload file Browse...

Job Title

Enter a descriptive title for your BLAST search

□ Align two or more sequences

Choose Search Set

Database: Non-redundant protein sequences (nr)

- Non-redundant protein sequences (nr)
- Reference proteins (refseq_protein)
- Swissprot protein sequences(swissprot)
- Patented protein sequences(pat)
- Protein Data Bank proteins(pdb)
- Environmental samples(env_nr)

Organism Optional: Enter organism name or id--completions will be suggested

Enter organism common name, binomial, or tax id. Only 20 top taxa will be shown.

Entrez Query Optional

Enter an Entrez query to limit search

Program Selection

Algorithm

● blastp (protein-protein BLAST)

○ PSI-BLAST (Position-Specific Iterated BLAST)

○ PHI-BLAST (Pattern Hit Initiated BLAST)

Choose a BLAST algorithm

blastp - compares a protein query to a protein database

PSI-BLAST - allows the user to build a PSSM using the results of the first BlastP run

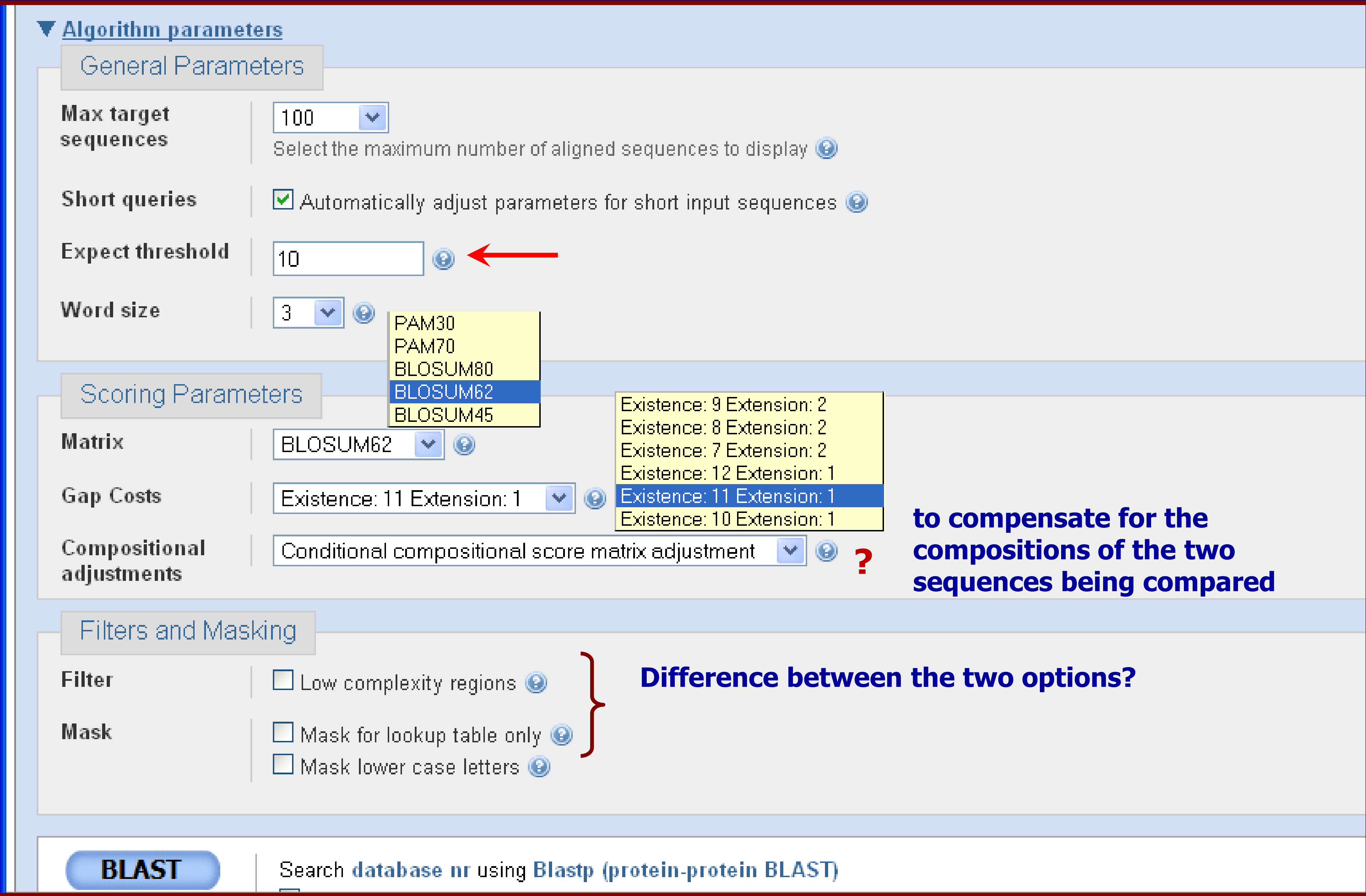
PHI-BLAST - performs the search but limits alignments to those that match a pattern in the query

BLAST

Search database nr using Blastp (protein-protein BLAST)

□ Show results in a new window

Algorithm parameters



▼ Algorithm parameters

General Parameters

Max target sequences: 100
Select the maximum number of aligned sequences to display

Short queries: ☑ Automatically adjust parameters for short input sequences

Expect threshold: 10

Word size: 3

Scoring Parameters

Matrix: BLOSUM62

- PAM30
- PAM70
- BLOSUM80
- BLOSUM62
- BLOSUM45

Gap Costs: Existence: 11 Extension: 1

- Existence: 9 Extension: 2
- Existence: 8 Extension: 2
- Existence: 7 Extension: 2
- Existence: 12 Extension: 1
- Existence: 11 Extension: 1
- Existence: 10 Extension: 1

Compositional adjustments: Conditional compositional score matrix adjustment

?

to compensate for the compositions of the two sequences being compared

Filters and Masking

Filter: □ Low complexity regions

Mask: □ Mask for lookup table only
□ Mask lower case letters

Difference between the two options?

BLAST

Search database nr using Blastp (protein-protein BLAST)

Specialised BLAST: PSI-BLAST

- Position Specific Iterated (PSI) BLAST
- Designed to find remote homologues (15 – 25% identity levels)
- Construct scoring matrices by multiple alignment of hits obtained
- Search the database with the new scoring matrix for every iteration
- Iterate until convergence is reached

Specialised BLAST: PSI-BLAST

- The idea of constructing a scoring matrix from the hits is that the new scoring matrix is tailor-made to find sequences similar to the query.

- allows detection of homologues in the range of 15%-25% sequence identity levels.

Specialised BLAST: PHI-BLAST

- Pattern Hit Iterated (PHI) BLAST
- Designed to find motifs
- Given a protein sequence and a motif/pattern within it, the program finds all proteins which carry that pattern and have similar surrounding residues
- Combines regular expression with local alignment around the matching region

BLAST: Statistics

Since BLAST uses a heuristic approach to accelerate aligning of two sequences, it becomes essential to compute the statistical significance of the alignment.

- Compute the probability that the alignment is obtained by "chance".

Strategy adopted by BLAST - compute distribution of alignment scores for random sequences.

- very little known about the random distribution of optimal global alignment scores

Statistics for ungapped local alignment score is quite well understood

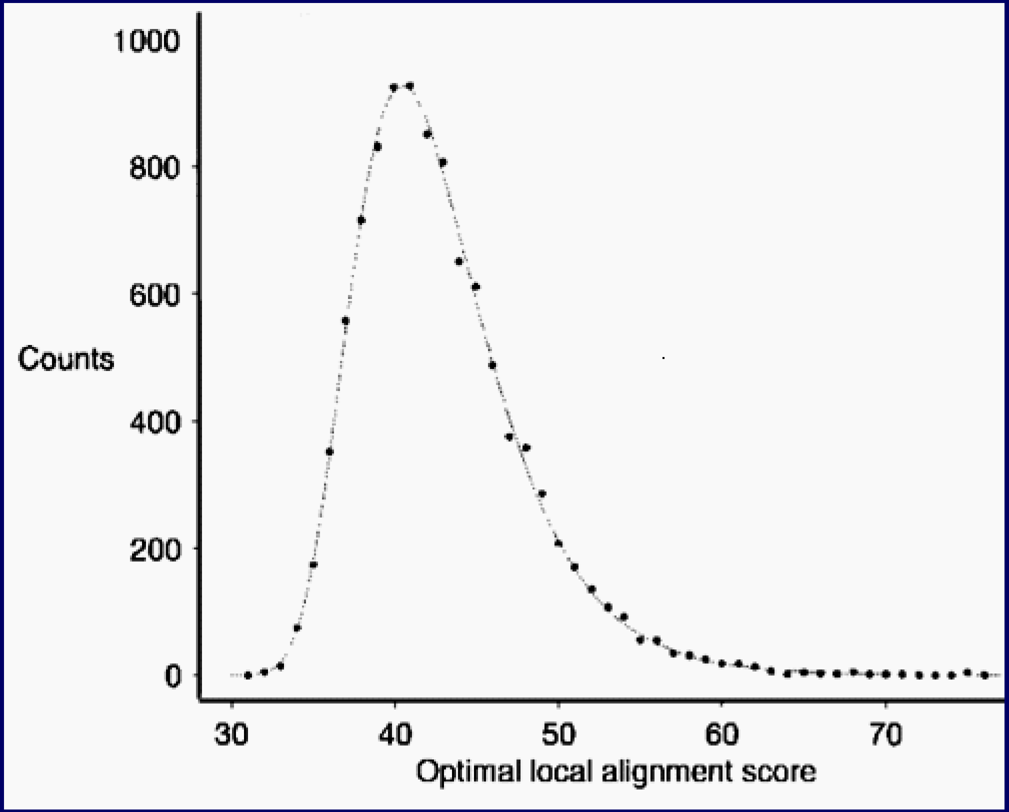
BLAST: Statistics

- Model random sequences generated: for proteins choose amino acids randomly with specific probabilities

- Align these random sequences using the Smith-Waterman algorithm and compute the score for the optimal alignment

- It is known that the distribution of the maximum of independent identically distributed (i.i.d.) random variables is an extreme value distribution

- the scores of optimal alignment of random sequences falls in this category



For 10000 pairs of 1000-length protein sequences, 10000 optimal local alignments were computed using Smith-Waterman. From these scores the values for *K* and λ were derived: 0.035 and 0.252

K **and** ***λ*** **are parameters related to the position of maximum and width of the distribution**

BLAST: Statistics

- In the limit of large sequence lengths *m* and *n*, expected number of HSPs with score at least *S* is:

$$E = K(mn)e^{-\lambda S}$$

Dependent on database size

- *K* and λ are empirical parameters derived from the distribution

- E-value depends on the size of query sequence as well as that of the database.

⇒for same aligned pair of sequences, E-value may be different depending on the database searched

Rule of thumb – search a larger database whenever possible

Significance of a hit: example

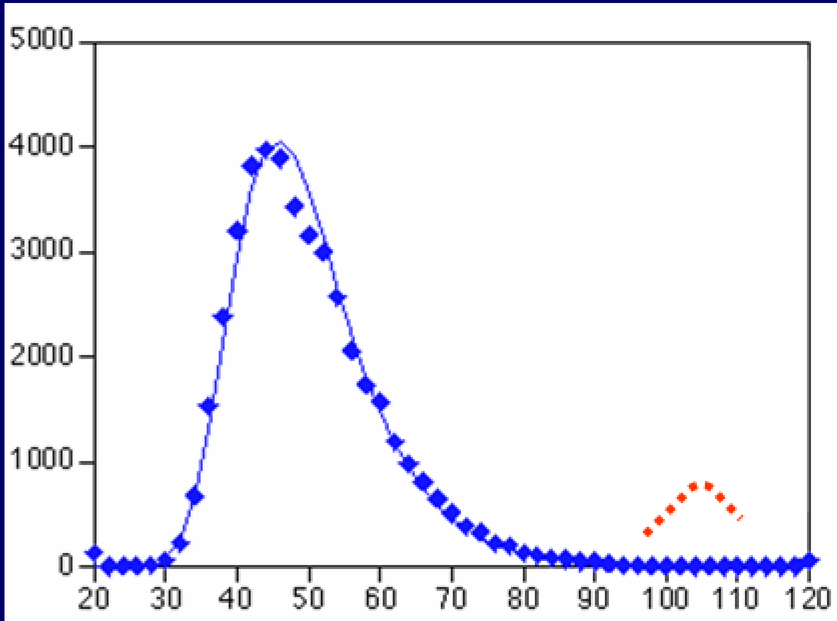
Search against a database of 10,000 sequences.

An extreme-value distribution (blue) is fitted to the distribution of all scores.

It is found that 99.9% of the blue distribution has a score below 112.

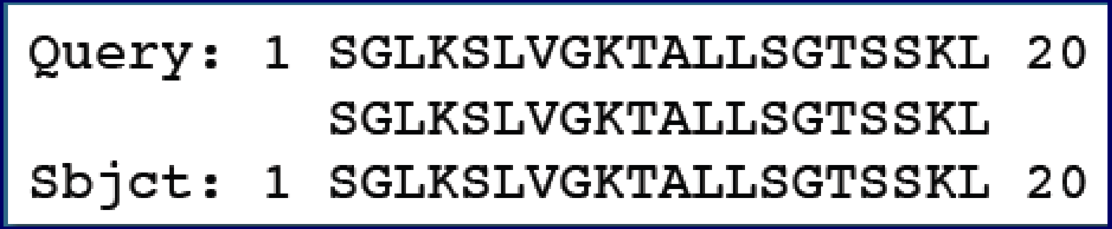
This means that when searching a database of 10,000 sequences you'd expect to get 0.1% * 10,000 = 10 hits with a score of 112 or better for random reasons

10 is the E-value of a hit with score 112. You want E-values well below 1!

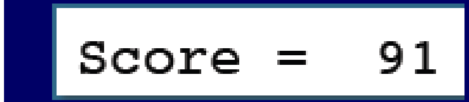


BLAST: Bit Scores

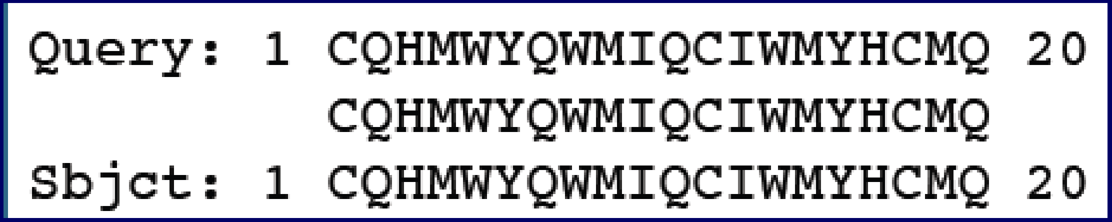
- Raw scores have little meaning without detailed knowledge of the scoring system used, or more simply its statistical parameters *K* and λ



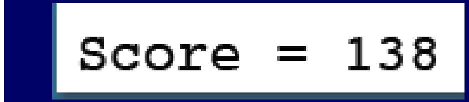
```
Query: 1 SGLKSLVGKTALLSGTSSKL 20
         SGLKSLVGKTALLSGTSSKL
Sbjct: 1 SGLKSLVGKTALLSGTSSKL 20
```



```
Score =  91
```



```
Query: 1 CQHMWYQWMIQCIWMYHCMQ 20
         CQHMWYQWMIQCIWMYHCMQ
Sbjct: 1 CQHMWYQWMIQCIWMYHCMQ 20
```



```
Score = 138
```

Based on scores ⇒ alignment 2 is better than alignment 1, but both the alignments are of the same length and have 100% identity

BLAST: Bit Scores

- Raw score, S, - sum of the alignment's pair-wise scores using a specific scoring matrix.
- By normalizing a raw score using the formula

$$S' = \frac{\lambda S - \ln K}{\ln 2}$$

E-value corresponding to a given bit score is

$$E = mn\,2^{-S'}$$

Note: λ and K are parameters dependent upon the scoring system employed.

BLAST: Bit Scores

- Bit score is calculated from the raw score by normalizing with the statistical variables that define a given scoring system.

- effect of normalization is to change the score distribution into standard normal distribution

- So, bit scores from different alignments, even those employing different scoring matrices can be compared.

- Higher the score, better the alignment, but the significance of an alignment can not be deduced from the score alone.

Using BLAST: Inferring Homology

How do you know when a certain level of similarity implies homology?

Does doing multiple searches help?

Using BLAST: Inferring Homology

Rules of thumb expressed in terms of percent identity in the optimal alignment:

- ~ 45% identity - the proteins will have very similar structures and are very likely to have a common or at least a similar function.
- ~ 25% identity - they are likely to have a similar general folding pattern.
- 18–25% identity - defined as 'twilight zone' - lower degree of sequence similarity cannot rule out homology.

Using BLAST: Inferring Homology

- In general, two sequences significantly similar over the entire length are likely to be homologous
- 50% similarity over a short sequence often occurs by chance
- Low complexity regions can be highly similar without being homologous
- Homologous sequences are not always highly similar
- Suggested BLAST cutoffs
  - For nucleotide-based searches, look for hits with e-value ≤ 10^{-6} and sequence identity ≥ 70%
  - For protein-based searches, look for hits with e-value ≤ 10^{-3} and sequence identity ≥ 25%

Substitution Matrices

Importance of scoring matrices

- Scoring matrices appear in all analysis involving sequence comparison.
- Choice of matrix can strongly influence the outcome of the analysis.
- Scoring matrices implicitly represent a particular theory of evolution.
- Understanding theories underlying a given scoring matrix can aid in making a proper choice

Substitution Matrices for Nucleotides

BLAST Matrix: In general, different substitution matrices are tailored to detecting similarities among sequences that are diverged by differing degrees, e.g.

	A	T	C	G
A	2	-3	-3	-3
T	-3	2	-3	-3
C	-3	-3	2	-3
G	-3	-3	-3	2

default – blastn

	A	T	C	G
A	1	-3	-3	-3
T	-3	1	-3	-3
C	-3	-3	1	-3
G	-3	-3	-3	1

default - megablast

- Default scoring scheme for blastn target sequences that are 90% identical, while the default scoring matrix for megablast is appropriate for sequences that are 99% identical,

Substitution Matrices for Nucleotides

Distance matrix:

Another measure, called edit distance, or cost is also used, e.g., Transition/Transversion Matrix:

	A	T	C	G
A	0	5	5	1
T	5	0	1	5
C	5	1	0	5
G	1	5	5	0

Using a Transition/Transversion matrix reduces noise in comparisons of distantly related sequences

Substitution Matrices For Nucleotides

ATGCCGTGATAGTCGAT
ACGGCTCGATCTACTAC

Identity Matrix: Score: 8 1's = **8**

Blast Matrix: Score: 8 x 1 – 3 x 9 = 8 – 27 = **- 19**

Transition/Transversion Matrix: (distance/cost)

Score: 8 x 0 + 3 x 1 + 6 x 5 = **33**

Which is better?

It is clear that raw alignment scores are meaningless without specific knowledge of the scoring matrix used.

PAM Matrices

PAM units & PAM matrices - developed by Margaret Dayhoff and coworkers.

- examined 1572 accepted mutations between 71 families of closely related sequences of proteins and noticed that the substitutions were not random.

⇒ evolutionarily related proteins need not have same AA at every position: can have a comparable one.

PAM stands for "Point Accepted Mutations" or "Percent of Accepted Mutations" ("accepted" refers to mutations that have become fixed in the population)

PAM matrices refer to various degrees of sensitivity depending on the evolutionary distance between sequence pairs

PAM Matrices

PAM units measure the amount of evolutionary distance between two protein sequences.

One PAM of evolution means that the total number of substitutions is 1% of the sequence length

After 100 PAMs of evolution, not every position would have changed, because some positions will have mutated several times, perhaps returning to their original state

In fact, even after 250 PAMs, proteins are still sufficiently similar that sequence homology can frequently be detected.

PAM Matrices

- If changes were purely random
  - Frequency of each possible substitution would be proportional to background frequencies
- In related proteins:
  - Observed substitution frequencies called the target (replacement) frequencies are biased toward those that do not disrupt the protein's function
  - These point mutations are "accepted" during evolution
- Log-odds approach:
  - Scores proportional to the natural log of the ratio of target frequencies to background frequencies

PAM Matrices

Score matrix entry for time *t* given by:

$$s(a, b|t) = \log \frac{P(a|b, t)}{q_a q_b}$$

Conditional probability that *b* is substituted by *a* in *t*

Frequency of AAs *a & b*

$$P(a|b) = P(a \cap b)/P(b)$$

PAM Matrices Construction

- Based on the hypothesis that proteins diverge by accumulating uncorrelated mutations

- Align closely related sequences (> 85% identity)

- considering very similar sequences allow the correct alignments to be determined with high certainty

- Observe the probability of AA changes & compute the log-odds ratio

- Normalize the matrix (relative frequencies of various mutations multiplied by a carefully chosen constant) to give an average change of 1% of all positions to obtain PAM-1 matrix

PAM Matrices Construction

How to derive scoring matrices for distantly related sequences from data about closely related sequences?

- **Scoring matrices to any PAM distance can be determined by extrapolating from PAM 1 – by successive iteration of a reference mutation matrix:**

$$\boxed{M_n = (M_1)^n}$$

e.g., for PAM 250, multiply PAM-1 250 times with itself.

Assumption in this evolutionary model is that AA substitutions observed over short periods of evolutionary history can be extrapolated to longer distances.

PAM Matrices Construction

$$\boxed{M_n = (M_1)^n}$$

$\boxed{M_1}$ - matrix reflecting 99% sequence conservation and one accepted point mutation (PAM 1) per 100 residues

$\boxed{M_n}$ - substitution probabilities after *n* PAMs

PAM250 – most frequently used matrix that is geared to very distant, but still detectable homologies

PAM matrices are derived from global alignments of closely related sequences

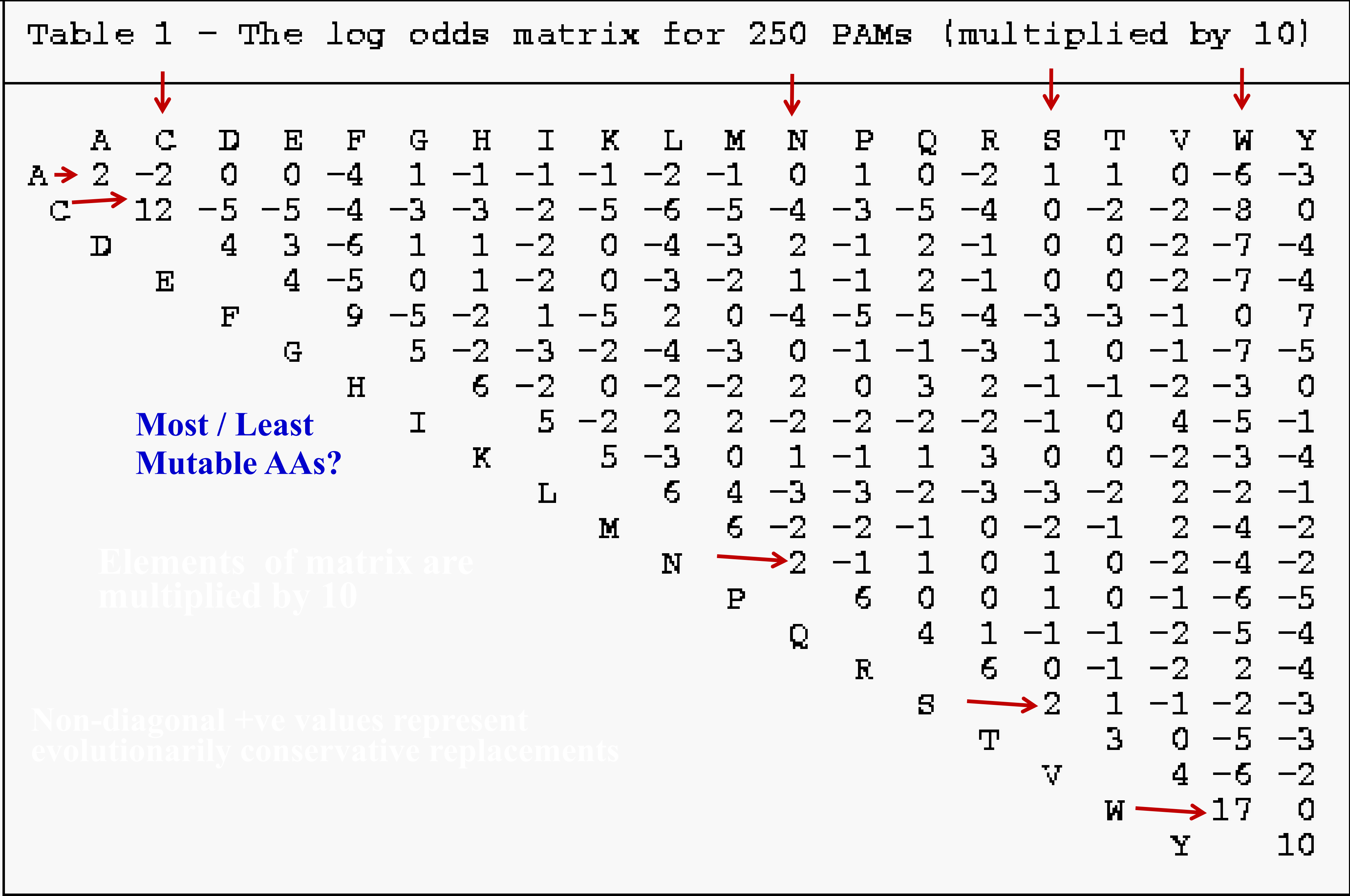


Table 1 – The log odds matrix for 250 PAMs (multiplied by 10)

	A	C	D	E	F	G	H	I	K	L	M	N	P	Q	R	S	T	V	W	Y
A	2	-2	0	0	-4	1	-1	-1	-1	-2	-1	0	1	0	-2	1	1	0	-6	-3
C		12	-5	-5	-4	-3	-3	-2	-5	-6	-5	-4	-3	-5	-4	0	-2	-2	-8	0
D			4	3	-6	1	1	-2	0	-4	-3	2	-1	2	-1	0	0	-2	-7	-4
E				4	-5	0	1	-2	0	-3	-2	1	-1	2	-1	0	0	-2	-7	-4
F					9	-5	-2	1	-5	2	0	-4	-5	-5	-4	-3	-3	-1	0	7
G						5	-2	-3	-2	-4	-3	0	-1	-1	-3	1	0	-1	-7	-5
H							6	-2	0	-2	-2	2	0	3	2	-1	-1	-2	-3	0
I								5	-2	2	2	-2	-2	-2	-2	-1	0	4	-5	-1
K									5	-3	0	1	-1	1	3	0	0	-2	-3	-4
L										6	4	-3	-3	-2	-3	-3	-2	2	-2	-1
M											6	-2	-2	-1	0	-2	-1	2	-4	-2
N												2	-1	1	0	1	0	-2	-4	-2
P													6	0	0	1	0	-1	-6	-5
Q														4	1	-1	-1	-2	-5	-4
R															6	0	-1	-2	2	-4
S																2	1	-1	-2	-3
T																	3	0	-5	-3
V																		4	-6	-2
W																			17	0
Y																				10

Most / Least Mutable AAs?

Elements of matrix are multiplied by 10

Non-diagonal +ve values represent evolutionarily conservative replacements

PAM Matrices

Salient Points:

- Matrices for greater evolutionary distances are extrapolated from those for lesser ones
- Number with the matrix (PAM40, PAM100, PAM250) refers to the evolutionary distance; larger numbers represent greater distances
- No clear correspondence between PAM distance and evolutionary time, since different protein families evolve at different rates
- Does not take into account different evolutionary rates between conserved & non-conserved regions.

BLOSUM – BLocks SUbstitution Matrix

BLOSUM matrix – given by Henikoff and Henikoff (1992)

- **Uses an alternative approach to determine a family of scoring matrices**

- **Uses structurally conserved protein domains from the BLOCKS database, which contains ungapped multiple alignments, called blocks, of core regions from hundreds of proteins**

- **Directly tabulates frequencies *p*(*x*,*y*) for distantly related proteins, instead of having a need to extrapolate from observation**

BLOSUM Matrices

Each matrix is tailored to a particular evolutionary distance, *viz*., for BLOSUM62 matrix, the default matric for BLAST:

In each block, sequences sharing at least 62% AA identity are clustered together

Then frequencies of aligned pairs $p(x, y)$ counted

Sequences more identical than 62% are represented by a single sequence in the alignment so as to avoid over-weighting closely related family members.

BLOSUM50 Estimation

- For each alignment in BLOCKS db, the sequences are grouped into clusters with at least 50% identical residues
- All pairs of sequences are compared, observed pair frequencies noted (e.g., A aligned with A makes up 1.5% of all pairs, A aligned with C makes up 0.01% of all pairs, etc.)
- Expected pair frequencies are computed from single AA frequencies, e.g,

  $f_{A,C}=f_A$ x $f_C=7\%$ x $3\% = 0.21\%$
- For each AA pair substitution scores are computed as:

$$\log \frac{\text{Pair\_freq (obs)}}{\text{Pair\_freq (expected)}}$$

- $S_{i,j}$ are scaled to give an integer value

```
ID   FIBRONECTIN_2; BLOCK
COG9_CANFA  GNSAGEPCVFPFIFLGKQYSTCTREGRGDGHLWCATT
COG9_RABIT  GNADGAPCHFPFTFEGRSYTACTTDGRSDGMAWCSTT
FA12_HUMAN  LTVTGEPCHFPFQYHRQLYHKCTHKGRPGPQPWCATT
HGFA_HUMAN  LTEDGRPCRFPFRYGGRMLHACTSEGSAHRKWCATTH
MANR_HUMAN  GNANGATCAFPFKFENKWYADCTSAGRSDGWLWCGTT
MPRI_MOUSE  ETDDGEPCVFPFIYKGKSYDECVLEGRAKLWCSKTAN
PB1_PIG     AITSDDKCVFPFIYKGNLYFDCTLHDSTYYWCSVTTY
SFP1_BOVIN  ELPEDEECVFPFVYRNRKHFDCTVHGSLFPWCSLDAD
SFP3_BOVIN  AETKDNKCVFPFIYGNKKYFDCTLHGSLFLWCSLDAD
SFP4_BOVIN  AVFEGPACAFPFTYKGKKYYMCTRKNSVLLWCSLDTE
SP1_HORSE   AATDYAKCAFPFVYRGQTYDRCTTDGSLFRISWCSVT
COG2_CHICK  GNSEGAPCVFPFIFLGNKYDSCTSAGRNDGKLWCAST
COG2_HUMAN  GNSEGAPCVFPFTFLGNKYESCTSAGRSDGKMWCATT
COG2_MOUSE  GNSEGAPCVFPFTFLGNKYESCTSAGRNDGKVWCATT
COG2_RABIT  GNSEGAPCVFPFTFLGNKYESCTSAGRSDGKMWCATS
COG2_RAT    GNSEGAPCVFPFTFLGNKYESCTSAGRNDGKVWCATT
COG9_BOVIN  GNADGKPCVFPFTFQGRTYSACTSDGRSDGYRWCATT
COG9_HUMAN  GNADGKPCQFPFIFQGQSYSACTTDGRSDGYRWCATT
COG9_MOUSE  GNGEGKPCVFPFIFEGRSYSACTTKGRSDGYRWCATT
COG9_RAT    GNGDGKPCVFPFIFEGHSYSACTTKGRSDGYRWCATT
FINC_BOVIN  GNSNGALCHFPFLYNNHNYTDCTSEGRRDNMKWCGTT
FINC_HUMAN  GNSNGALCHFPFLYNNHNYTDCTSEGRRDNMKWCGTT
FINC_RAT    GNSNGALCHFPFLYSNRNYSDCTSEGRRDNMKWCGTT
MPRI_BOVIN  ETEDGEPCVFPFVFNGKSYEECVVESRARLWCATTAN
MPRI_HUMAN  ETDDGVPCVFPFIFNGKSYEECIIESRAKLWCSTTAD
PA2R_BOVIN  GNAHGTPCMFPFQYNQQWHHECTREGREDNLLWCATT
PA2R_RABIT  GNAHGTPCMFPFQYNHQWHHECTREGRQDDSLWCATT
```

$$S_{A,C} = \log \frac{0.01}{0.21} = -1.3$$

BLOSUM50 matrix:

- **Positive scores on diagonal (identities)**
- **Similar residues get positive scores (marked in red)**
- **Dissimilar residues get smaller (negative) scores**

A	5																			
R	-2	7																		
N	-1	-1	7																	
D	-2	-2	2	8																
C	-1	-4	-2	-4	13															
Q	-1	1	0	0	-3	7														
E	-1	0	0	2	-3	2	6													
G	0	-3	0	-1	-3	-2	-3	8												
H	-2	0	1	-1	-3	1	0	-2	10											
I	-1	-4	-3	-4	-2	-3	-4	-4	-4	5										
L	-2	-3	-4	-4	-2	-2	-3	-4	-3	2	5									
K	-1	3	0	-1	-3	2	1	-2	0	-3	-3	6								
M	-1	-2	-2	-4	-2	0	-2	-3	-1	2	3	-2	7							
F	-3	-3	-4	-5	-2	-4	-3	-4	-1	0	1	-4	0	8						
P	-1	-3	-2	-1	-4	-1	-1	-2	-2	-3	-4	-1	-3	-4	10					
S	1	-1	1	0	-1	0	-1	0	-1	-3	-3	0	-2	-3	-1	5				
T	0	-1	0	-1	-1	-1	-1	-2	-2	-1	-1	-1	-1	-2	-1	2	5			
W	-3	-3	-4	-5	-5	-1	-3	-3	-3	-3	-2	-3	-1	1	-4	-4	-3	15		
Y	-2	-1	-2	-3	-3	-1	-2	-3	2	-1	-1	-2	0	4	-3	-2	-2	2	8	
V	0	-3	-3	-4	-1	-3	-3	-4	-4	4	1	-3	1	-1	-3	-2	0	-3	-1	5
	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V

BLOSUM 62 Matrix

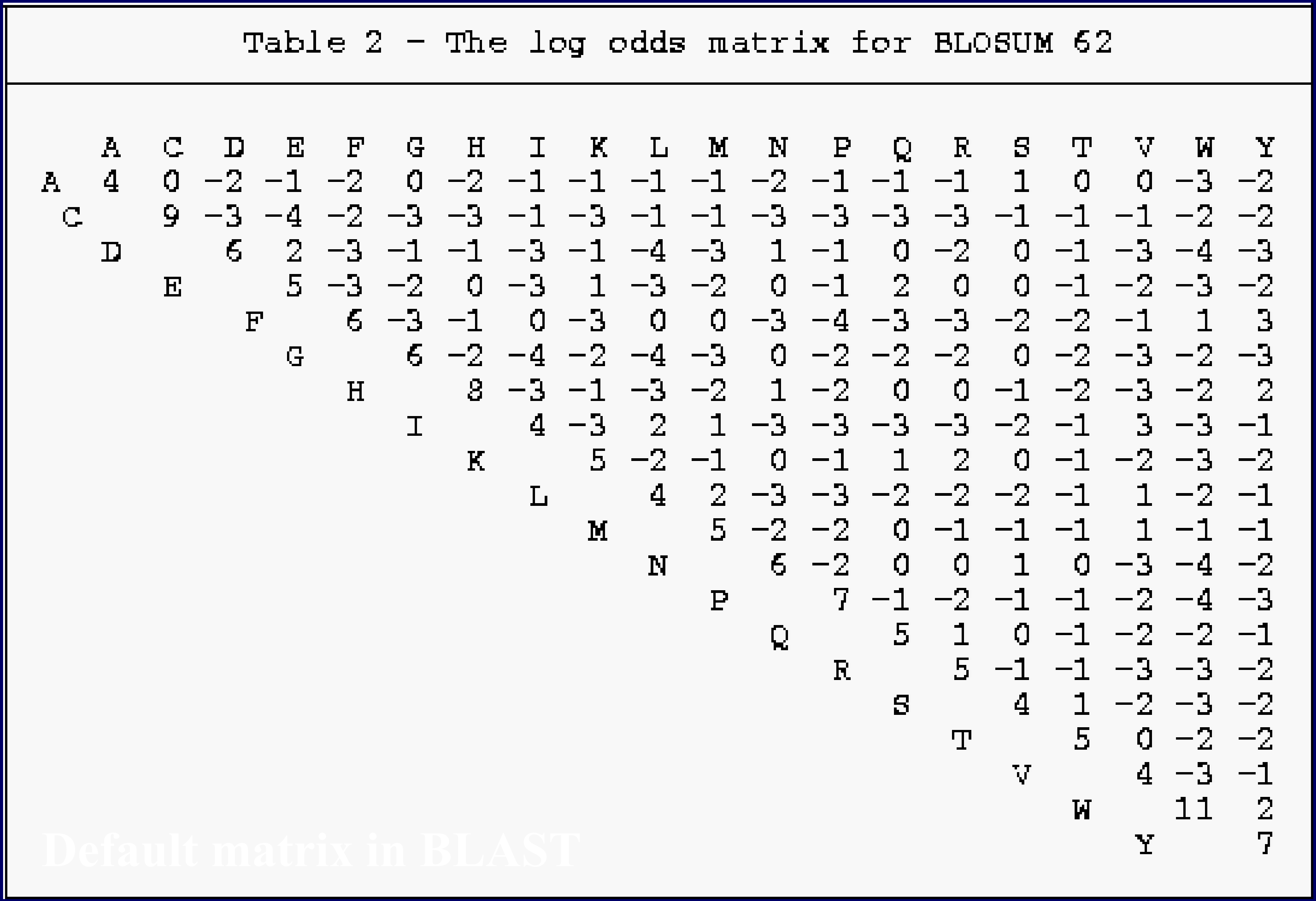


Table 2 - The log odds matrix for BLOSUM 62

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

Default matrix in BLAST

BLOSUM Matrices

Salient Points:

- Derived from local, ungapped alignments of distantly related sequences
- Uses blocks of protein sequence fragments from different families (the BLOCKS database)
- Blocks represent structurally conserved regions
- Amino acid pair frequencies calculated by summing over all possible pairs in block
- Different evolutionary distances are incorporated into this scheme with a clustering procedure (identity over particular threshold = same cluster)

BLOSUM Matrices

- All matrices are directly calculated; no extrapolations are used – no explicit model
- Number after the matrix (BLOSUM62) refers to the minimum percent identity of the blocks used to construct the matrix; greater numbers represent lesser distances.
- BLOSUM 62 is the default matrix in BLAST
- For database searches, Blosum matrices have often been the better performers

- the reason being these matrices are based on the replacement patterns found in more highly conserved regions of the sequences.

BLAST: Nucleotide Scoring Matrix

- Scoring Matrices for match (M)/mismatch (N):
  - 1, -2
  - 1, -3
  - 1, -4
  - 2, -3
  - 4, -5 ⟵ default
  - 1, -1

Relative magnitudes of M & N determines the No. of nucleic acid PAMs (point accepted mutations per 100 residues) for which they are most sensitive at finding homologs.

BLAST: Nucleotide Scoring Matrix

A ratio of 0.33 (1/-3) is appropriate for sequences that are about 99% conserved

A ratio of 0.5 (1/-2) is best for sequences that are 95% conserved

A ratio of one (1/-1) is best for sequences that are 75% conserved

- the (absolute) reward/penalty ratio should be increased as one looks at more divergent sequences