Database Similarity Searching

Irit Orr

Shifra Ben-Dor

Bioinformatics Lecture 6 2022

PAIRWISE ALIGNMENT

$\downarrow$

DATABASE SEARCHING

$\downarrow$

MULTIPLE ALIGNMENT

$$\begin{tikzcd}
 & \text{MULTIPLE ALIGNMENT} \arrow[dl] \arrow[d] \arrow[dr] & \\
\text{Homology Modeling} & \text{Advanced Database Searches, Patterns, Motifs, Promoters} & \text{Phylogenetic Analysis}
\end{tikzcd}$$

Why search databases?

- To find out if a new DNA sequence shares similarities with sequences already deposited in the databanks.
- To find proteins homologous to a putative coding ORF.
- To find similar non-coding DNA stretches in the database, (for example: repeat elements, regulatory sequences).
- To locate false priming sites for a set of PCR oligonucleotides.

What databases are available?

- DNA (nucleotide sequences):

  The big databases: Genbank, Embl, DDBJ and their weekly updates. These databases exchange information routinely.

- Genomic databases, for example: Human, Mouse, Yeast...

- Special databases:

  SRA (Short read archive), TSA (Transcript sequence assembly), EPD (eukaryotic promoter database), ESTs (expressed sequence tags), REPBASE (repetitive sequence database) and many others.

What databases are available?

- Protein (amino acid sequences):

  The major database is:

  Uniprot/Swiss-Prot (high level of annotation)

  Translated databases like:

  Uniprot/TREMBL (translated EMBL)

  GenPept (translation of coding regions in GenBank)

- Special databases like:

  PDB (sequences which have 3D structures)

What is a homologous sequence?

- A homologous sequence, in molecular biology, means that the sequence is similar to another sequence. The similarity is derived from common ancestry.
- Homologous proteins generally are are similar in their folding or their structure.

DNA vs. Protein searches

- DNA is composed of 4 characters: A,C,G,T

  For any position, there is 25% chance that the bases of two aligned sequences would be identical, even if they are unrelated.
- Protein sequence is composed of 20 characters (aa). The sensitivity of the comparison is improved. It is accepted that convergence of proteins is rare, meaning that high similarity between two proteins almost always means homology.

DNA vs. Protein searches

- What should we use to search for similarity, nucleotide or protein sequences?
- If we have a nucleotide sequence, should we search only DNA databases or should we translate it to protein and search protein databases?
- Note that by translating DNA into protein, we'll presumably lose information, since the genetic code is degenerate, meaning that two or more codons translate to the same amino acid.

Codon Wobble

Peptide	(1)	MET	LYS	PRO	HIS
DNA	(1)	ATG	AAA	CCT	CAT
	(2)	ATG	AAG	CCT	CAT
	(3)	ATG	AAA	CCC	CAT
	(4)	ATG	AAG	CCC	CAT
	(5)	ATG	AAA	CCA	CAT
	(6)	ATG	AAG	CCA	CAT
	(7)	ATG	AAA	CCG	CAT
	(8)	ATG	AAG	CCG	CAT
	(9)	ATG	AAA	CCT	CAC
	(10)	ATG	AAG	CCT	CAC
	(11)	ATG	AAA	CCC	CAC
	(12)	ATG	AAG	CCC	CAC
	(13)	ATG	AAA	CCA	CAC
	(14)	ATG	AAG	CCA	CAC
	(15)	ATG	AAA	CCG	CAC
	(16)	ATG	AAG	CCG	CAC

DNA vs. Protein searches

- What about very different DNA sequences that code for similar protein sequences?

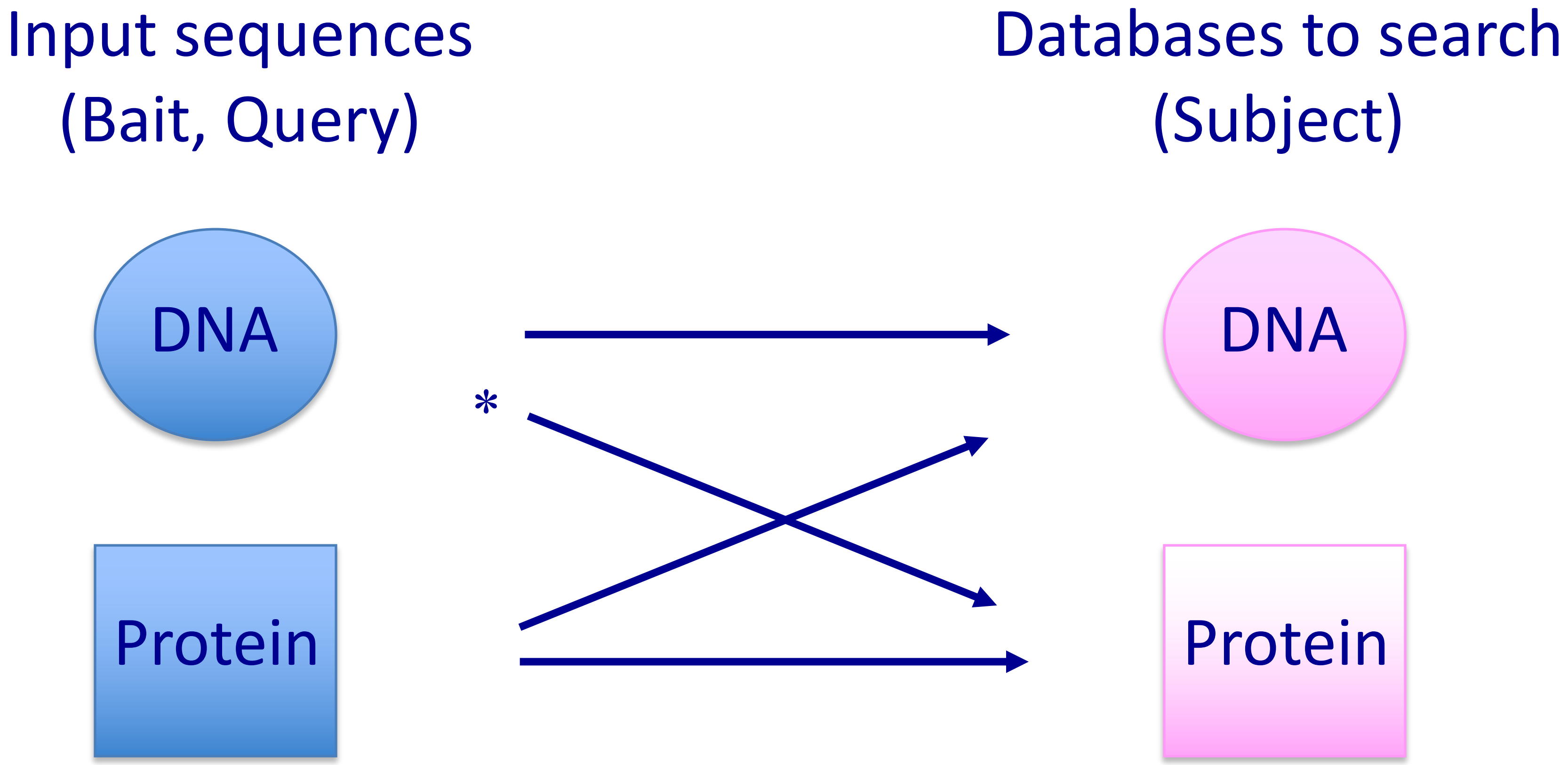
  We certainly do not want to miss those.

- What about the size of the databases?

  DNA databases are huge in size compared to Protein databases.

  - Huge databases = many sequences = many random hits.

DATABASES AND INPUT SEQUENCES



*Does not apply when searching for non translated sequences

DNA vs. Protein searches

- When comparing DNA sequences, we get significantly more random matches than we get with proteins (due to differences in alphabet).
- The DNA databases are much larger, and grow faster than Protein databases. Bigger databases mean more random hits!
- For DNA we usually use identity matrices, for protein more sensitive matrices like PAM and BLOSUM, which allow for better search results.
- Evolutionarily, protein sequences tend to diverge less than the DNA encoding them.

DNA vs. Protein searches

To sum it up:

We should use proteins for database similarity searches when possible (depending, of course, on the biological question, and especially for more distant sequences).

Basic principles of db searching

- When searching a database, we take a query sequence and use an algorithm (program) for the search.
- Every pair compared yields a score.
- Larger scores usually indicate a higher degree of similarity.
- A typical db search will yield a huge number of scores to be analyzed.

Pairwise Alignment

Alignments between 2 sequences:

- Can start and end anywhere in each sequence
- Don’t have to match in all bases
- May include gaps in either of the sequences

```
ATTGTCAAAGACTTGAGCTGATGCAT
        |||| ||| ||||
     GGCAGACATGA.CTGACAAGGGTATCG
```

Aligning 2 Sequences

```
ATTGCAGTGATCG
ATTGCGTCGATCG
```

Solution 1:

```
ATTGCAGTGATCG
|||||   |||||
ATTGCGTCGATCG
```

Solution 2:

```
ATTGCAGT-GATCG
||||| ||  |||||
ATTGC-GTCGATCG
```

Aligning 2 Sequences

What is a better solution?

Solution 1:

```
ATTGCAGTGATCG
|||||   |||||
ATTGCGTCGATCG
```

Solution 2:

```
ATTGCAGT-GATCG
||||| ||  |||||
ATTGC-GTCGATCG
```

?

3 mismatches + 10 matches

2 gaps + 12 matches

Scoring Pairwise Alignments

Scores in an alignment are given to:

- eq_score - the score for a match
- dif_score - the score for a mismatch
- gap_open - the penalty score for opening a gap
- gap_extend - the penalty score for extension of each gap by 1 character (including the first one!)

Optimal alignment using scores

What is a “Better” Alignment?

```
ATTGCAGTGATCG
|||||   |||||
ATTGCGTCGATCG
```

3 mismatches

+ 10 matches =

3(-4) + (10 x 3) = 18

```
ATTGCAGT-GATCG
||||| ||  |||||
ATTGC-GTCGATCG
```

2 gaps + 12 matches =

2(-12 - 4) + (12 x 3) = 4

Pairwise Comparison of Proteins

```
POTYYILT
|  || |:|
PRTYDIIT
```

Matches and conservative substitutions receive good (positive) scores

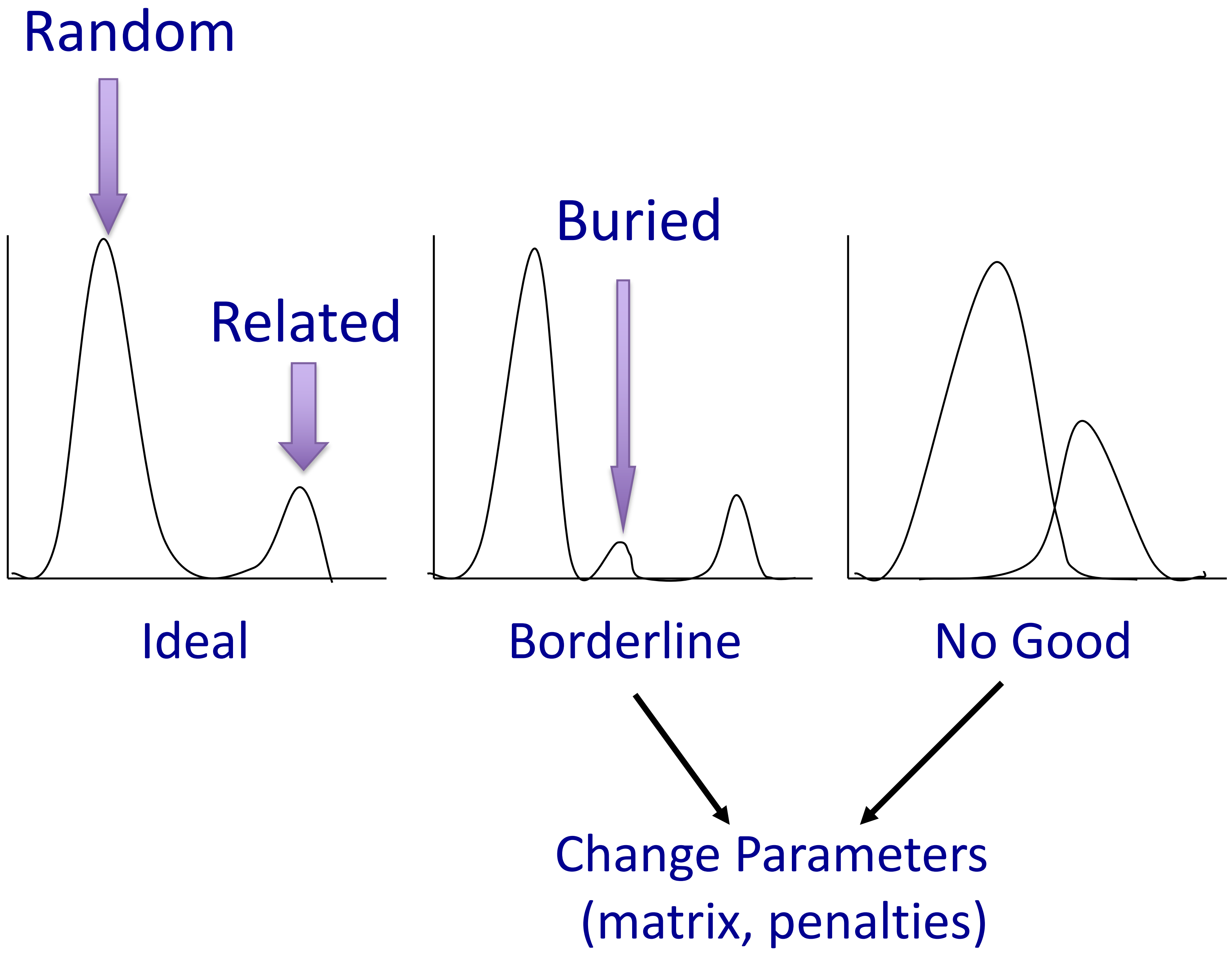
Non-conservative substitutions and gaps receive bad (negative) scores

% identity of the alignment = matches

% similarity is defined by the amount of positive scores out of the alignment (matches + substitutions).

Basic principles of db searching

- Using *pairwise* comparison, each database search normally yields 2 groups of scores: genuinely related and genuinely unrelated sequences, with some overlap between them.
- A good search method should completely separate between the 2 score groups.
- In practice no search method succeeds in total separation.


Random
Related
Buried
Ideal
Borderline
No Good
Change Parameters
(matrix, penalties)

Specificity and sensitivity

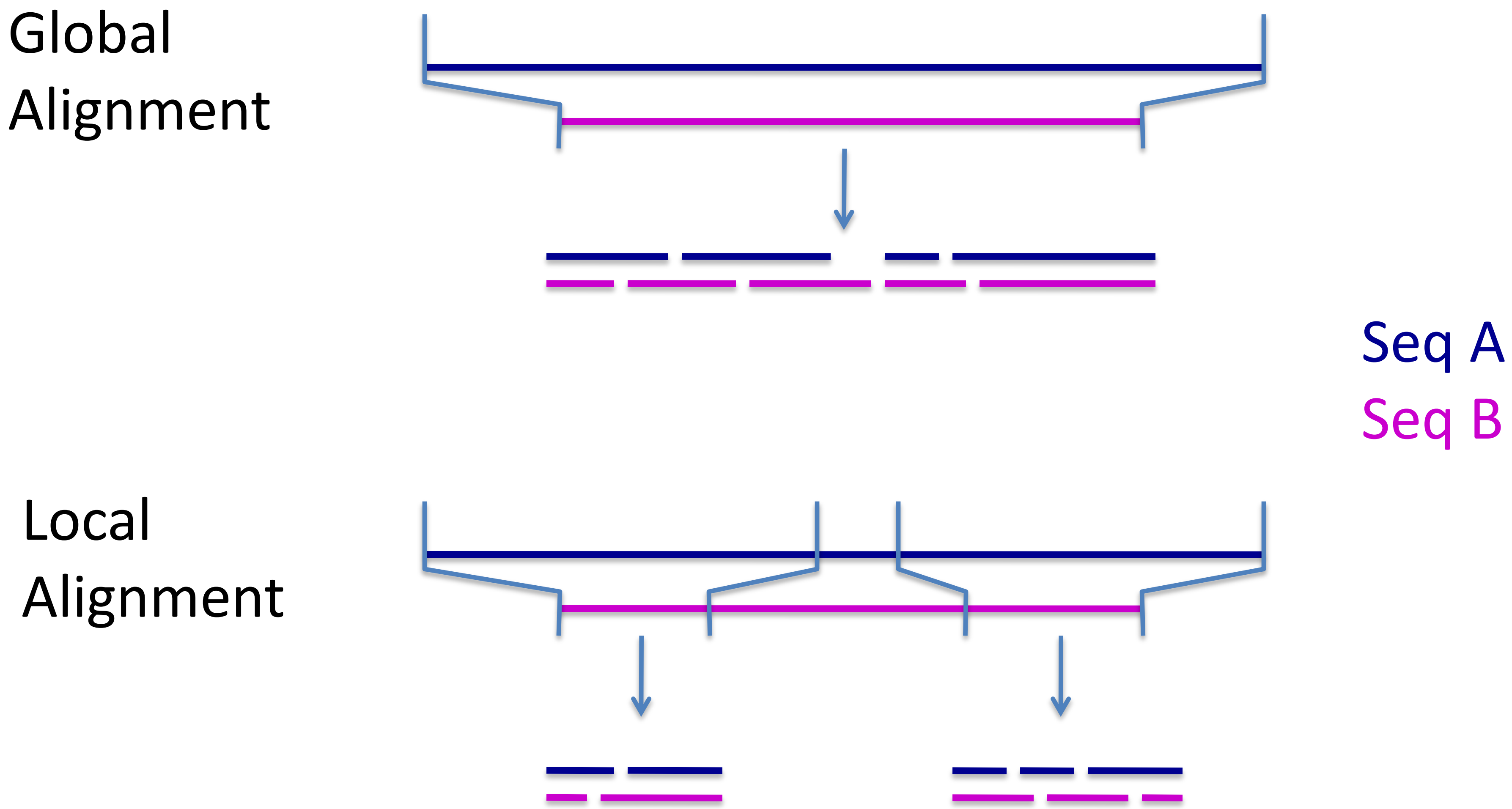
Definitions

- Sensitivity: the ability to detect "true positive" matches. The most sensitive search finds all true matches, but might have lots of false positives
- Specificity: the ability to reject "false positive" matches. The most specific search will return only true matches, but might have lots of false negatives.

Two types of alignment:

- Global alignment

- Local alignment


Global
Alignment
Seq A
Seq B
Local
Alignment

Main algorithms for database searching

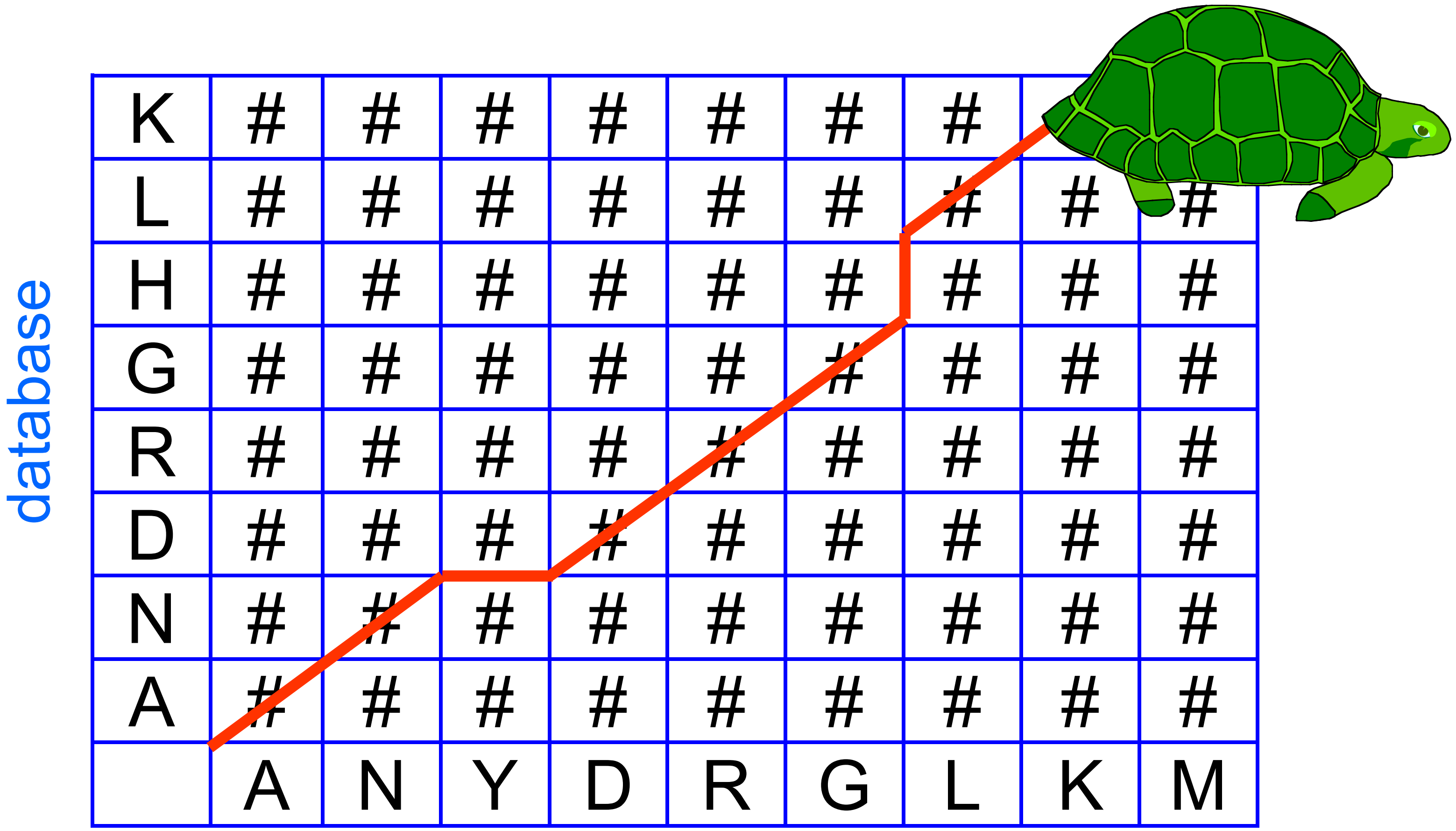
- FastA : Is theoretically better for nucleotides than BLAST (statistics are more rigorous)
- BLAST: Better for proteins than for nucleotides
- Blat: Optimized for fast searches on the genome level
- Smith-Waterman: More sensitive than FastA or BLAST. Good for distantly related sequences.

The Smith-Waterman Tools

- Smith-Waterman searching method is a dynamic programming method.
- Dynamic programming techniques build a solution to a problem by solving sub-problems.
- The Smith-Waterman method uses full pairwise comparison of the query sequence to each sequence in database.
- Then, some statistics are calculated on the search results.

The Smith-Waterman matrix

Score calculation **requires** value in each cell



database

K	#	#	#	#	#	#	#	[illegible]	[illegible]
L	#	#	#	#	#	#	#	#	#
H	#	#	#	#	#	#	#	#	#
G	#	#	#	#	#	#	#	#	#
R	#	#	#	#	#	#	#	#	#
D	#	#	#	#	#	#	#	#	#
N	#	#	#	#	#	#	#	#	#
A	#	#	#	#	#	#	#	#	#
	A	N	Y	D	R	G	L	K	M

query

“All fast alignment programs that I am aware of break the alignment problem into two parts. Initially in a “search stage,” the program detects regions of the two sequences which are likely to be homologous. The program then in an “alignment stage” examines these regions in more detail and produces alignments for the regions which are indeed homologous according to some criteria. The goal of the search stage is to detect the vast majority of homologous regions while reducing the amount of sequence that is passed to the alignment stage.”

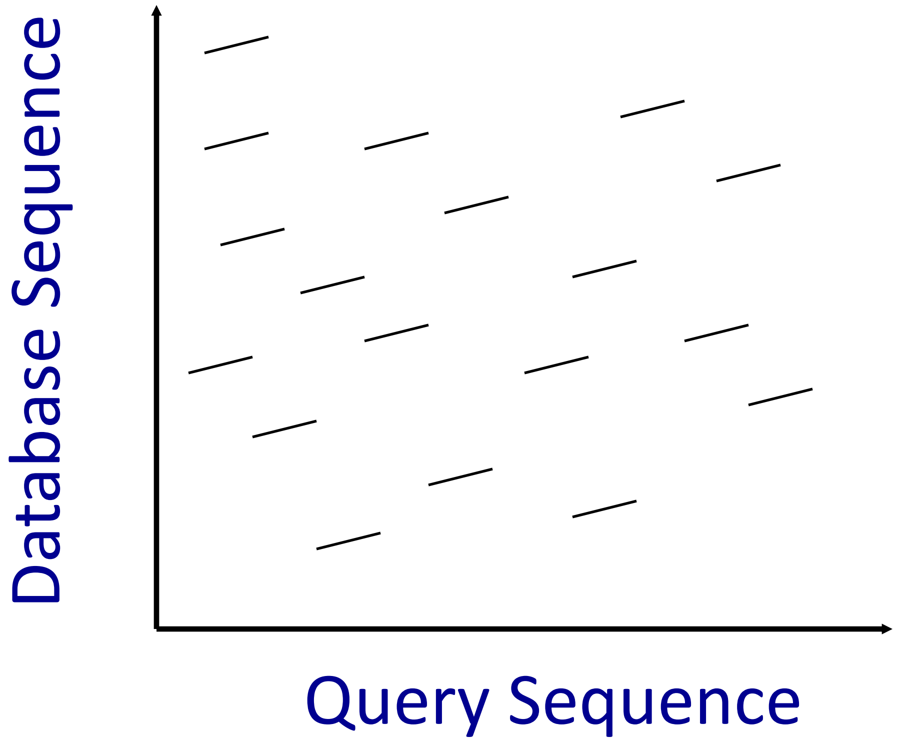
Kent, W.J. 2002. Genome Research 4: 656-664.

The FastA software package

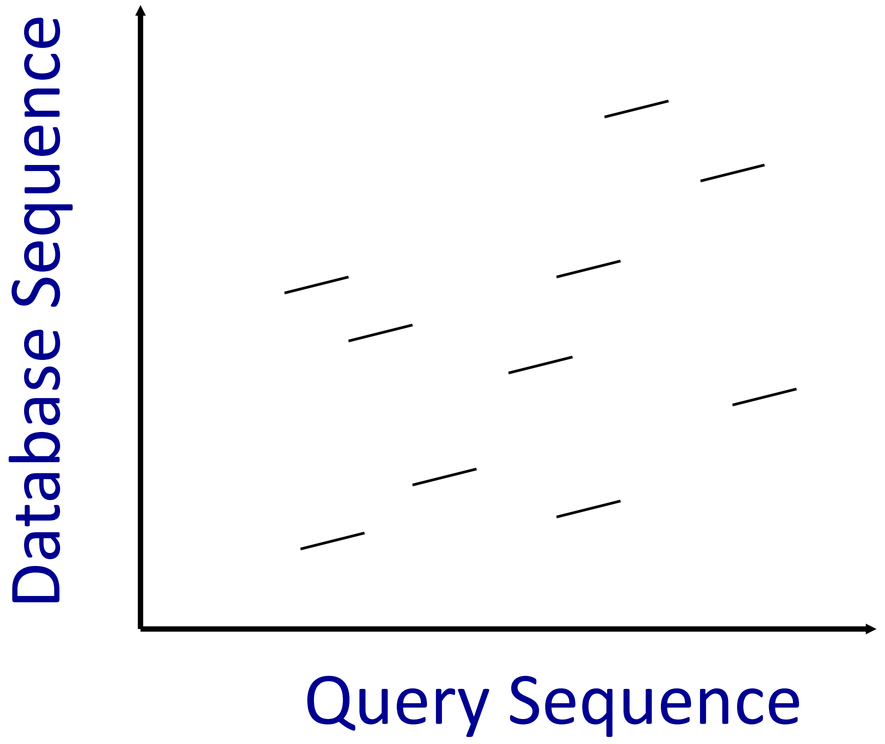
- FastA uses the method of Pearson and Lipman (PNAS 85: 2444-2448, 1988).
- FastA is a family of programs, which include:
  - FastA, TFastA, Ssearch, etc…
- FastA is a GLOBAL alignment algorithm

How does the FastA program work?

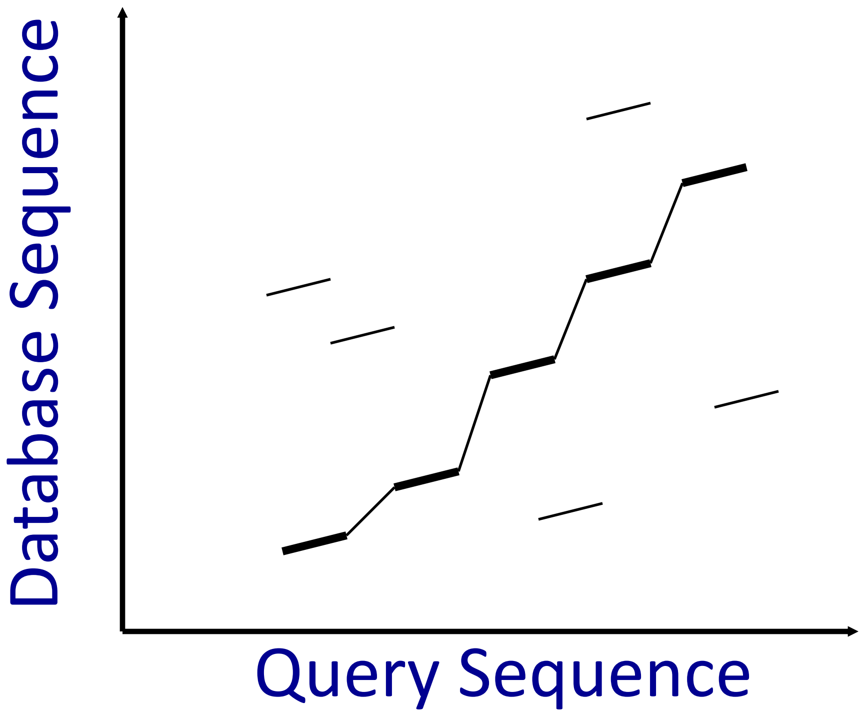
FastA locates regions of the query sequence and the search set sequence that have high densities of exact word matches.



The ten highest-scoring regions are saved and rescored using a scoring matrix. The score of the highest scoring initial region is saved as the **<u>init1 score</u>**.



Next: FastA determines if any of the initial regions from different diagonals may be joined together to form an approximate alignment with gaps. Only non-overlapping regions may be joined. The score for the joined regions is the sum of the scores of the initial regions minus a joining penalty for each gap. The score of the highest scoring region, at the end of this step, is saved as the **initn score**.



After computing the initial scores, FastA uses dynamic programming (Smith-Waterman algorithm) over a narrow region of high scoring diagonals between the query sequence and the search set sequence, to produce an alignment with a new score.

The alignment score is the **opt score**.

- Last: FastA uses a simple linear regression against the natural log of the search set sequence length to calculate a normalized **z-score** for the sequence pair.

- Using the distribution of the z-score, the program can estimate the number of sequences that would be expected to produce, purely by chance, a z-score greater than or equal to the z-score obtained in the search. This is reported as the **E() score**.

! Note:

- The program will calculate the extreme value distribution, μ and lambda, for expected scores distribution.
- The calculations vary with sequence length in the dataset, their composition, and the scoring matrix used.
- Therefore, small datasets will effect the accuracy of the statistical calculations in FASTA.

Parameters in the FastA package

- Search speed and selectivity are controlled with the “ktup” (wordsize) parameter.
  - Tips for ktup: For proteins, the default, ktup=2, ktup=1 is more sensitive but slower.
  - For DNA, ktup=6, the default, ktup=3 or ktup=4 give more sensitivity, ktup=1 for oligonucleotides (length <20).

Output of FastA

```
The best scores are:                   init1 initn   opt    z-sc E(699079)..

EM_HUM1:HSACHRA    Begin: 1  End:  1667
! Y00762 Human mRNA for muscle acetyl... 8335  8335  8335  9159.3       0
EM_HUM2:S77094    Begin: 1  End:  1667
! S77094 nicotinic acetylcholine rece... 8299  8299  8299  9119.6       0
EM_OM:BTACHRA1    Begin: 10  End:  1422
! X02509 B.Taurus mRNA for acetylchol... 6018  6244  6048  6636.8       0
EM_RO:MMACHRAM    Begin: 4  End:  1634
! X03986 Mouse mRNA for muscle nicoti... 5570  5630  5881  6457.6       0
EM_RO:MMACHRAB    Begin: 59  End:  1731
! M17640 Mus musculus acetylcholine r... 5552  5607  5873  6448.4       0
EM_RO:RNACRA1    Begin: 27  End:  1678
! X74832 R.norvegicus mRNA for acetyl... 5550  5713  5807  6375.8       0
EM_OV:XLACHRA    Begin: 32  End:  1416
! X07067 Xenopus mRNA for muscle aety... 3309  3309  3558  3901.8       0
EM_OV:FSACHRA    Begin: 243  End:  1572
! J00963 Ray (T.californica) acetylch... 3345  3345  3527  3865.4       0
EM_OV:TMACHR    Begin: 120  End:  1449
! M25893 T.marmorata acetylcholine re... 3318  3318  3500  3836.4       0
EM_OV:DRU70438    Begin: 180  End:  1536
! U70438 Danio rerio muscle nicotinic... 3129  3129  3426  3753.7       0
EM_OV:XLACHRA1    Begin: 16  End:  1397
```

Output of FastA

```
y00762
EM_HUM1:HSACHRA

ID   HSACHRA    standard; RNA; HUM; 1667 BP.
AC   Y00762;
NI   g28308
DT   02-APR-1988 (Rel. 15, Created)
DT   23-MAR-1995 (Rel. 43, Last updated, Version 6)
DE   Human mRNA for muscle acetylcholine receptor alpha-subunit . . .


SCORES Init1:8335  Initn:8335  Opt: 8335 z-score: 9159.3 E(): 0
 100.0% identity in 1667 bp overlap

                     10        20        30        40        50
y00762       AAGCACAGGCCACCACTCTGCCCTGGTCCACACAAGCTCCGGTAGCCCATGGA
             |||||||||||||||||||||||||||||||||||||||||||||||||||||
HSACHRA      AAGCACAGGCCACCACTCTGCCCTGGTCCACACAAGCTCCGGTAGCCCATGGA
                     10        20        30        40        50
```

BLAST - Basic Local Alignment Search Tool

- Blast programs use a heuristic search algorithm. The programs use the statistical methods of Karlin and Altschul (1990,1993).
- Blast programs were designed for fast database searching, with minimal sacrifice of sensitivity to distant related sequences.
- Blast is a LOCAL alignment algorithm

BLAST - Basic Local Alignment Search Tool

- BLAST programs search databases in a special compressed format.
- To use your own private database with blast, you need to format it in blast format.

BLAST Programs

- BLAST is actually a family of programs
  - BLASTN - Nucleotide query searching a nucleotide database.
  - BLASTP - Protein query searching a protein database.
  - BLASTX - Translated nucleotide query sequence (6 frames) searching a protein database.
  - TBLASTN - Protein query searching a translated nucleotide (6 frames) database.
  - TBLASTX - Translated nucleotide query (6 frames) searching a translated nucleotide (6 frames) database.

Blast method

- Blast uses a heuristic method to find the highest scoring local alignment between the query sequence and the search set sequence.
- The original blast algorithm did not allow gaps, and relied on the statistics of ungapped alignments.
- The current version of Blast allows short gaps and has better statistics.

BLAST - Basic Local Alignment Search Tool

BLAST uses “common words” for the initial database search.

(This is done to increase the search speed).

Steps used by Blast algorithm

- The database

  is indexed and a dictionary of words with different length is built.

- The query sequence
- Is first scanned for low-complexity regions or repeats.
- A list of words, (of a given length), is created, starting from position 1, then 2 until the end of the sequence is reached.

Steps used by Blast algorithm

Step 1:

The program scans each db sequence for an exact match to a given length word.

If a match is found, it is used to seed a possible ungapped alignment between the 2 sequences.

Steps used by Blast algorithm

- The query sequence:
- Using a scoring matrix (for example, BLOSUM62), the short words are evaluated for exact match with the words of the databases sequences.
  - The object of the evaluation is to find the scores for aligning the query-word with any other word (of given length) from the database.
  - Score calculations are done and saved.

Steps used by Blast algorithm

- The query sequence:
- A cutoff threshold score, T, is selected, in order to select the most significant matches.
  - For example: if T = 13, only words that score above 13 are kept.
- This evaluation procedure is repeated for each word of given length in the sequence dictionary.

Steps used by Blast algorithm

- The query sequence:

- The remaining high-scoring words from the whole procedure are organized into an efficient search tree, in order to compare them rapidly to all database sequences.

Steps used by Blast algorithm

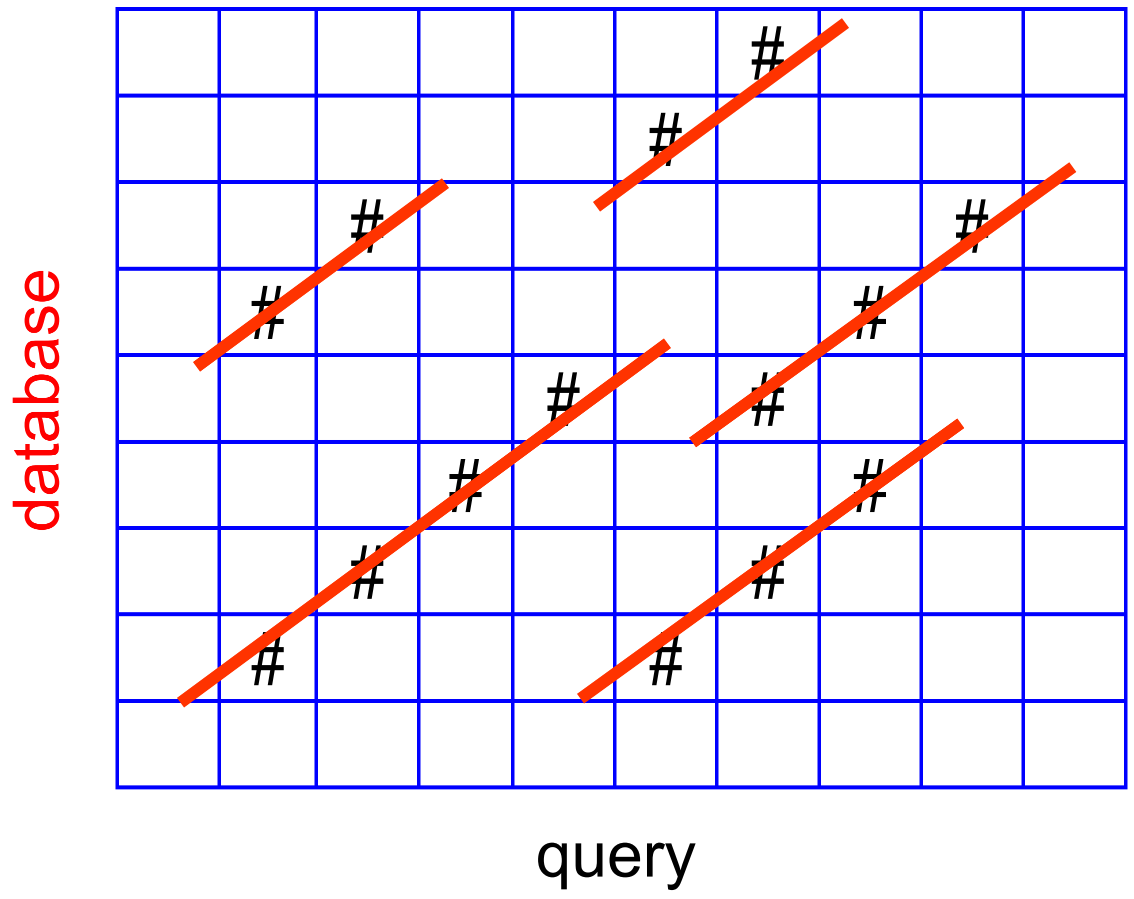
Step 2:

(a) In the original method, the alignment was extended at both sides of the hit, as long as the score of the alignment increased. The extension stopped when the score began to fall.

This sequence segment of matches (hits) was called HSP (High Scoring segment-Pair).

BLAST - Local alignment

Extending the best diagonals



Steps used by Blast algorithm

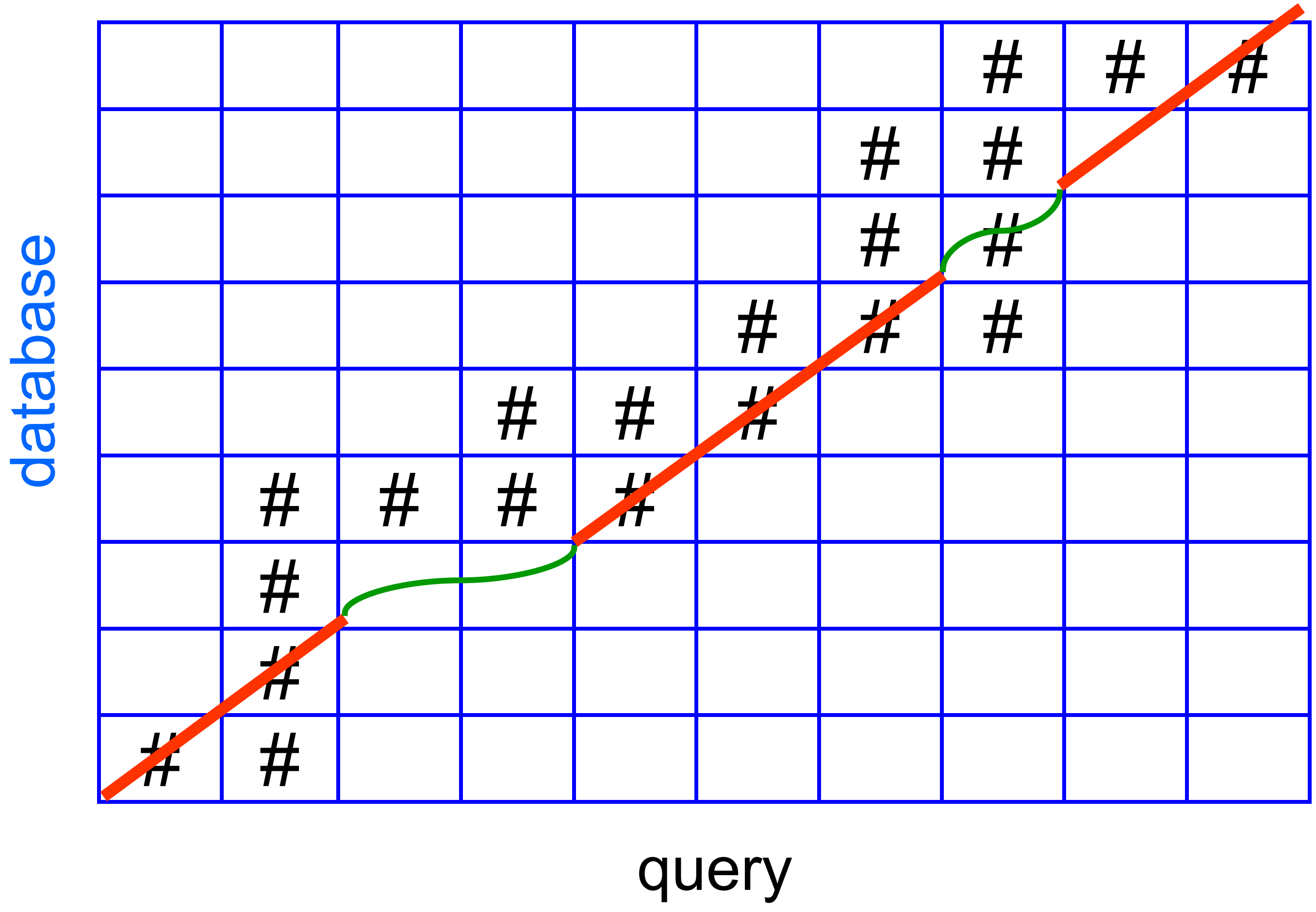
Step 2:

(b) In the later version of BLAST the method changed. The change is in a lower T value, resulting in a longer words list.

The program looks for the word match first, then tries to locate several short words on the same diagonal, and then extends the alignment on both sides of the hits on the same diagonal, allowing short gaps.

The newly found diagonals are scored (sum of the scores of the individual matches).

Extension of HSPs allowing short gaps



Steps used by Blast algorithm

Step 3:

The HSPs of the entire database are compared to a cutoff score S, and when greater then S, are listed.

Step 4:

Statistical significance calculations are done for each HSP score.

E-Score

- The E-score is the Expect value: How many times do I expect to see an alignment with a bit score of X in a database of size Y.
- It is not a measure of similarity, it only gives an approximation of how relevant the results are
- It is dependent on the database SIZE, so different databases will give different E-scores for the same alignment
- Statistically significant E-scores: under 0.05

Steps used by Blast algorithm

Step 5:

Alignment of the segments are done, the alignment score is obtained, and the E() value for this score is calculated.

Last step:

If the calculated E() for the database sequence meets the user given E() for the program, this score is reported.

Blast output

- The list of hits
- Database accession codes, name, description, general information about the hit
- Score in bits, the alignment score expressed in units of information. Usually 30 bits are required for significance.
- Expectation value E()

It is important to keep in mind that the **E() value does not represent a measure of similarity between the two sequences!**

Blast output

- The information for each hit
- A header including hit name, description, length
- The same for all additional entries removed due to redundancy
- Composite expectation value
- Each hit may contain several HSPs
- score and expectation value
- – how many identical residues
- – how many residues contributing positively to the score
- The local alignment itself

Statistical evaluation of results

- When the program finds a similarity between your query sequence and a database sequence it is not always clear how significant this similarity really is.

- To evaluate if this similarity is statistically significance, you should run a pairwise comparison, preferably with the randomization option

Other issues

- Low Complexity Sequence
- Compositional Bias
- Repeat regions
- Database size

Low Complexity Regions

- Regions of low complexity (in both DNA and protein) will give hits that are significant but not necessarily biologically interesting (a long stretch of A for example)

- To deal with that we can filter the query sequence, either for the initial word match, or for the full alignment process

Compositional Bias

- Some sequences have compositional bias - either because the underlying genome is G/C or A/T rich (causes a tendency towards certain amino acids over others), or have regions that are particularly rich in one amino acid (Proline, Histidine, Glutamine)

- The traditional scoring matrices are built on the assumption that the amino acid distribution is normal, and won’t score these matches properly

Compositional Bias

- A new scoring method has been added for BlastP, and is now the default: Conditional Compositional Score Matrix Adjustment

- This allows the program to “fix” the matrix based on parameters in the pair of sequences being compared.

- This will give more true hits, and less hits that are significant, but not biologically interesting

Repeat Sequences

- In genomic DNA, there are repetitive elements that will skew any search results (for example Alu repeats in human)
- A filter can be used to ‘clean’ up the query sequence, and mask any known repeats before the search is performed.

Database size

- The e-score is dependent on database size.
- If we change the database, we change the score
- This is true for whole databases (nr vs. swissprot) as well as database segments

Database segments

- We can subdivide the database to help us find more relevant results
- We can limit it by a specific species, by groups of species (plants, vertebrates, mammals, viruses, bacteria…)
- We can also limit (at NCBI) by Entrez terms, including NOT

Blat – Blast Like Alignment Tool

- Blat is a shortcut on Blast, optimized for fast searching of sequences against a genome database

- Kent, W.J. 2002. BLAT -- The BLAST-Like Alignment Tool. Genome Research 4: 656-664.

Blat – Blast Like Alignment Tool

- Blat looks for two perfect matches of a given length (default K of 11 for DNA), constrained to the same diagonal and are near each other.
- For protein, it uses three perfect 4-mers (on-line) or one perfect 5-mer (stand alone)
- It uses overlapping K-mers of the query, and non-overlapping K-mers of the database (minus repeats)
- It finds initial hits, and groups them together as long as they are close to each other

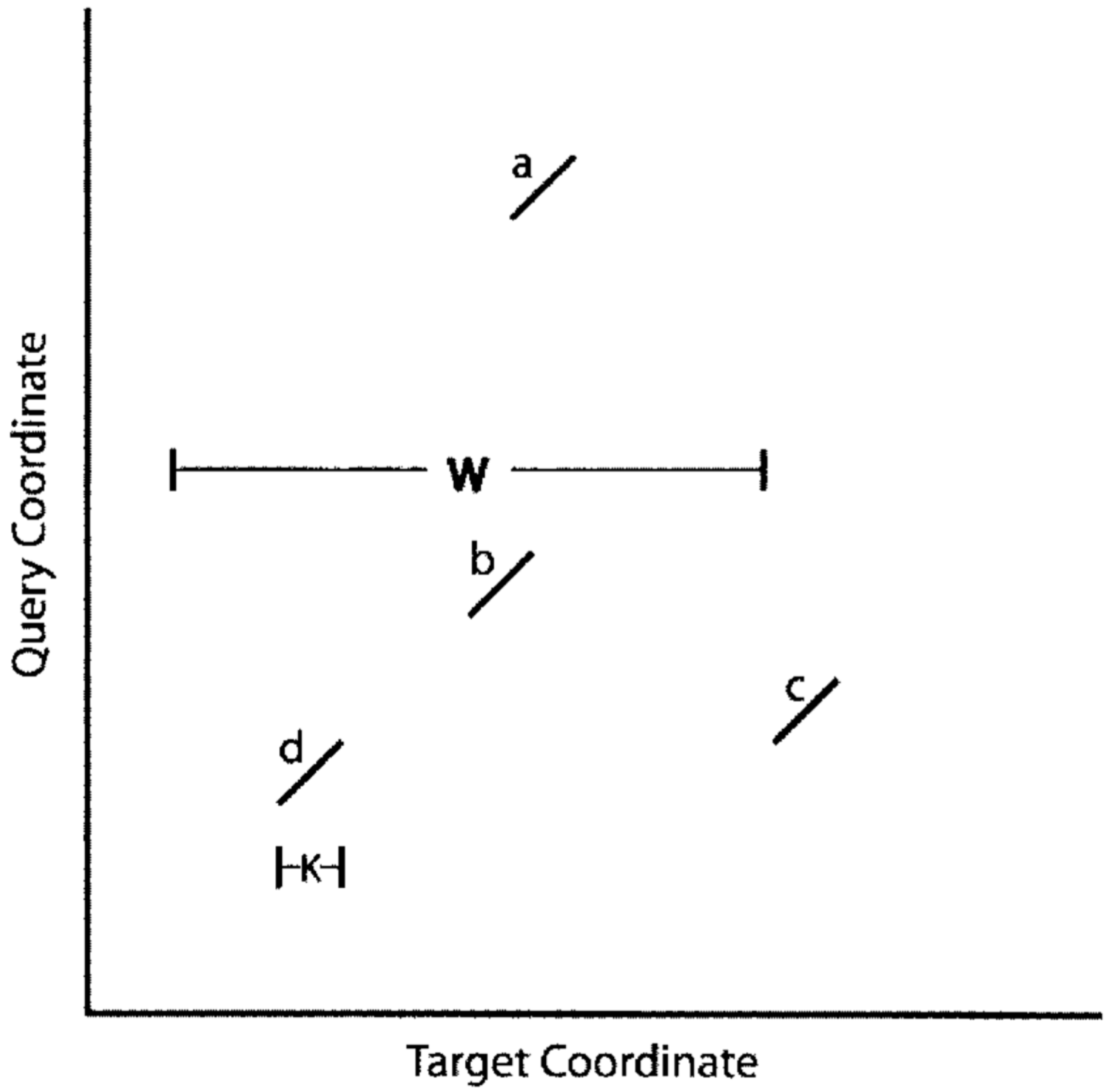


Figure 1 A pair of hits and two other hits. The hits a, b, c, and d are all K letters long. Hits d and b have the same diagonal coordinate and are within W letters of each other. Therefore they would match the "two perfect K-mer" search criteria.

Blat – Blast Like Alignment Tool

- It then goes on to do the alignment, refining the original hits
- As it was optimized for mRNA to genome alignments, it knows to allow large gaps, and looks for cannonical splice junctions (GT/AG) in deciding where to place the gap
- It works on genomic alignments as well, but only when the sequences are closely related (more than 90% identical)

Conclusions

Local alignments:

- Can start and end anywhere within the sequences
- Therefore the location of the highest score (S) can be at any cell within the matrix

Multiple Solutions:

- Sometimes more than one optimal alignment exists
- In such cases, there are several optimal alignment paths on the matrix

Heuristic Vs. Rigorous

Heuristic algorithms are less sensitive

- Both FastA and BLAST are approximations of Smith-Waterman algorithm. Hence, both perform heuristic filtering of the database.
- Distantly related sequences (e.g. - containing many substitutions and gaps) are less likely to be discovered with heuristic applications

Rigorous algorithms are computationally intensive

- Smith-Waterman finds similarities in a rigorous way (e.g. - calculates the accumulating score for every cell in the matrix). Hence, it runs 100-1000 slower than FastA and BLAST.
- Hardware accelerators make rigorous algorithms practical.

Comparison of programs

- SW, BLAST and Blat: local alignments
- FASTA: global alignments

BLAST can report more than one segment of HSP per database entry

FASTA reports only one segment (match) for each database entry.

- Speed (not accelerated):
- Blat >> BLAST > FASTA >> SW
- Sensitivity: SW > FASTA > BLAST > Blat

Tips for DB searches

- Use latest database version
- Run Blast first, then depending on your results run a finer tool (fasta, SW, etc…)
- Where possible use translated sequence. It's better to use proteins then DNA.
- E() < 0.05 is statistically significant, usually biologically interesting. Check also 0.05 < E() <10 because you might find interesting hits.

Tips for DB searches

- Pay attention to abnormal composition of the query sequence, it usually causes biased scoring.
- If the query sequence has repeated segments, or low complexity segments, remove them or mask them before running your search.

Tips for DB searches

- Where should I do my search? Local or Network?

  Generally, we use network, except for special cases:

  If you want to change parameters that the web version doesn't allow

  If you have patent issues, and don't want to run on a public server

- Second, local databases are (almost) always available, where network databases may not be (communication problems, blocking IPs, system overload)