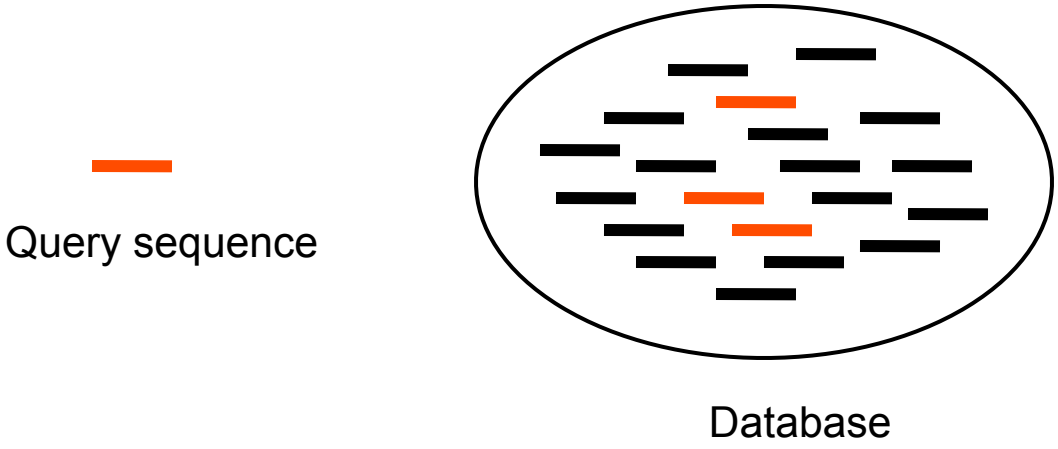
BLAST

Anders Gorm Pedersen
&
Rasmus Wernersson

Database searching

Using pairwise alignments to search databases for similar sequences



Database searching

Most common use of pairwise sequence alignments is to search databases for related sequences. For instance: find probable function of newly isolated protein by identifying similar proteins with known function.

Most often, ***local*** alignment (“Smith-Waterman”) is used for database searching: you are interested in finding out if ANY domain in your protein looks like something that is known.

Often, full Smith-Waterman is too time-consuming for searching large databases, so heuristic methods are used (fasta, BLAST).

Database searching: heuristic search algorithms

FASTA (Pearson 1995)

Uses heuristics to avoid calculating the full dynamic programming matrix

Speed up searches by an order of magnitude compared to full Smith-Waterman

The statistical side of FASTA is still stronger than BLAST

BLAST (Altschul 1990, 1997)

Uses rapid word lookup methods to completely skip most of the database entries

Extremely fast

- One order of magnitude faster than FASTA
- Two orders of magnitude faster than Smith-Waterman

Almost as sensitive as FASTA

BLAST flavors

BLASTN
- Nucleotide query sequence
- Nucleotide database

BLASTP
- Protein query sequence
- Protein database

BLASTX
- Nucleotide query sequence
- Protein database
- Compares all six reading frames with the database

TBLASTN
- Protein query sequence
- Nucleotide database
- ”On the fly” six frame translation of database

TBLASTX
- Nucleotide query sequence
- Nucleotide database
- Compares all reading frames of query with all reading frames of the database

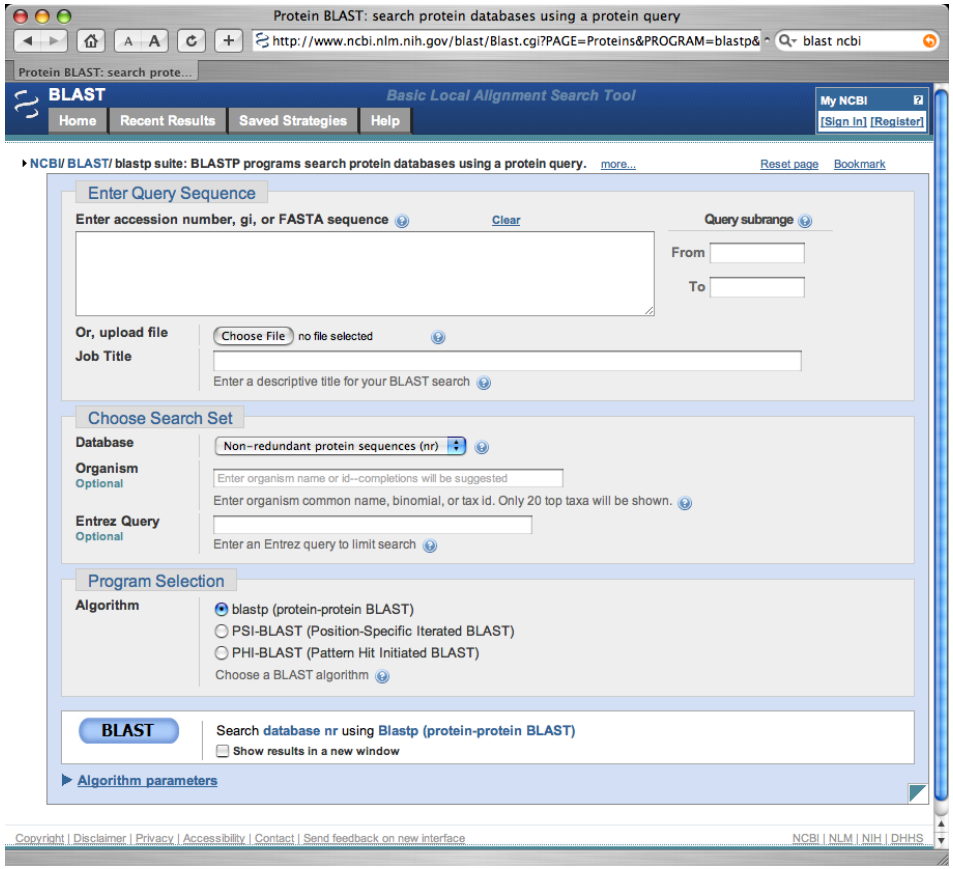
Searching on the web: BLAST at NCBI

Very fast computer dedicated to running BLAST searches

Many databases that are always up to date (e.g. NR and Human Genome

Nice simple web interface

But you still need knowledge about BLAST to use it properly



When is a database hit significant?

- **Problem**:
  - Even unrelated sequences can be aligned (yielding a low score)
  - How do we know if a database hit is meaningful?
  - When is an alignment score sufficiently high?

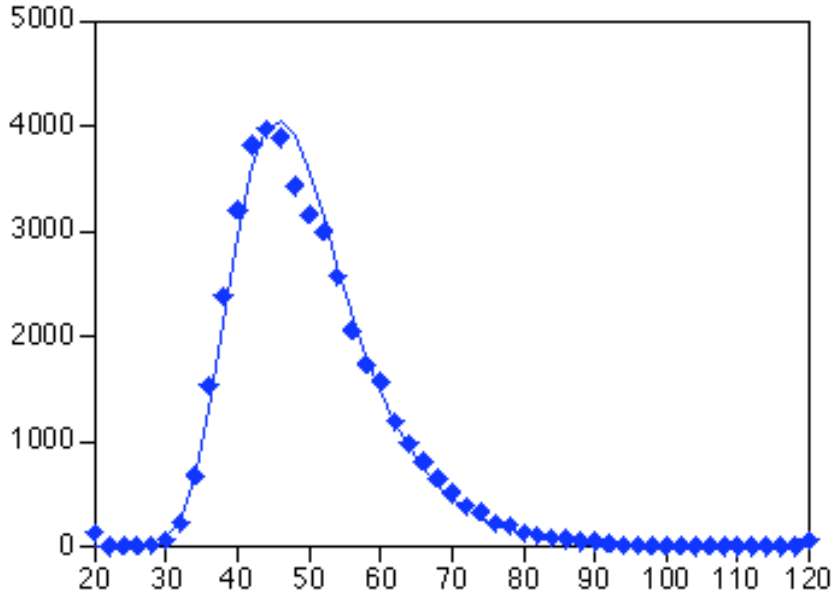
- **Solution**:
  - Determine the range of alignment scores you would expect to get for random reasons (i.e., when aligning unrelated sequences).
  - Compare actual scores to the distribution of random scores.
  - Is the real score much higher than you'd expect by chance?

Distribution of random alignment scores

- **Software simulation**

Random alignment scores follow extreme value distributions

Searching a database of unrelated sequences result in scores following an extreme value distribution



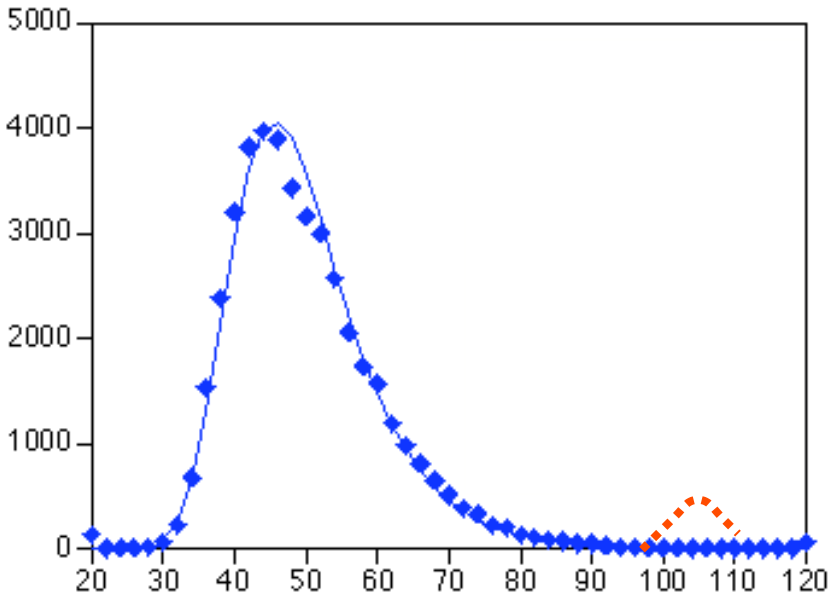
The exact shape and location of the distribution depends on the exact nature of the database and the query sequence

Significance of a hit: one possible solution

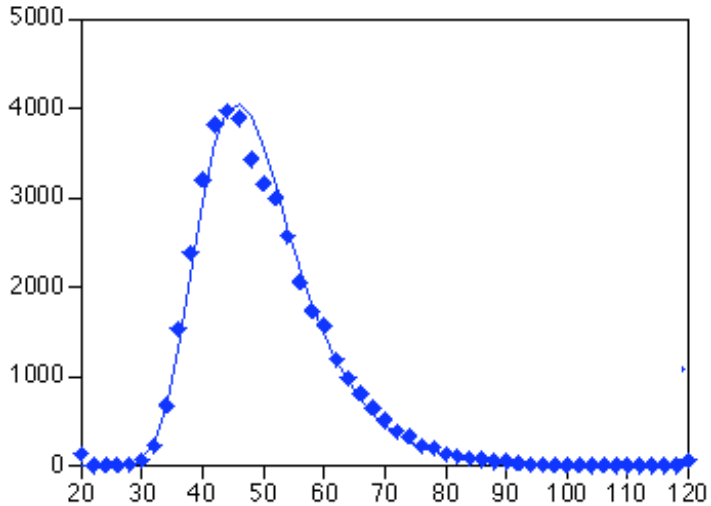
(1) Align query sequence to all sequences in database, note scores

(2) Determine shape of background distribution (which is an extreme value distribution) from distribution of all scores

(3) Use fitted extreme-value distribution to predict how many random hits to expect for any given score (the "**E-value**")



Database searching: E-values in BLAST



BLAST uses precomputed extreme value distributions to calculate E-values from alignment scores

For this reason BLAST only allows certain combinations of substitution matrices and gap penalties

This also means that the fit is based on a different data set than the one you are working on

A word of caution: BLAST tends to overestimate the significance of its matches

E-values from BLAST are fine for identifying sure hits
One should be careful using BLAST's E-values to judge if a marginal hit can be trusted (e.g., **you may want to use E-values of 10^{-4} to 10^{-5}**).

BLAST heuristics

- Best possible search:
  - Do full pairwise alignment (Smith-Watermann) between the query sequence and **all** sequences in the database.
  - (“ssearch” does this).

- BLAST speeds up the search by at least two orders of magnitude, by pre-screening the database sequences and only performing the full Dynamic Programming on “*promising*” sequences.

- This is done by indexing all databases sequences in a so-called **suffix-tree** which makes it very fast to search for perfect matching sub-strings.
  - A suffix tree is the quickest possible way (so far) to search for the *longest matching sub-string* between two strings.

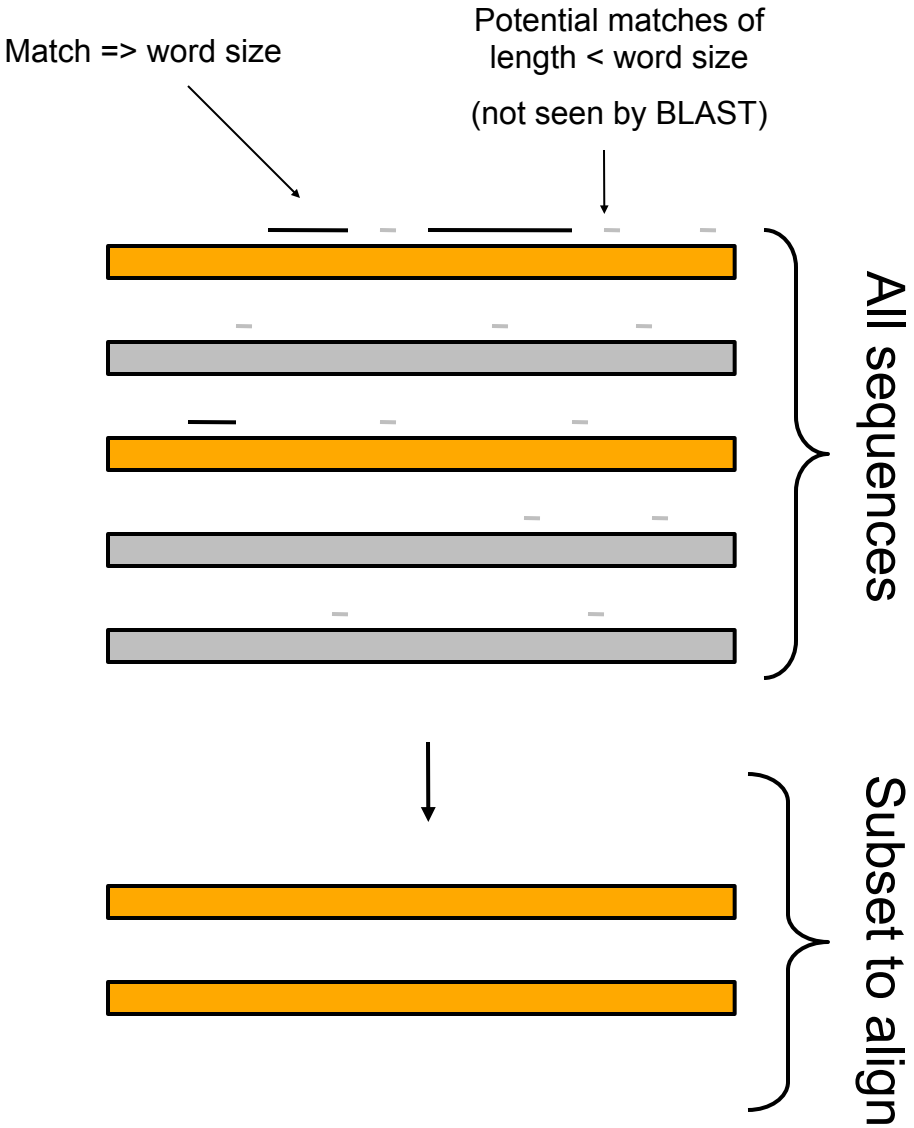
- When a BLAST search is run, candidate sequences from the database is picked based on perfect matches to small sub-sequences in the query sequence. (**BLASTN** and **BLASTP** does this differently - more about this in a moment).
  - Full Smith-Waterman is then performed on these sequences.

BLASTN

- Alignment matrix:
  - Perfect match: **1**
  - Mismatch: **-3**

- Notice: All mismatches are equally penalized:
  - E.g. A:G == A:C == A:T
  - More advanced models for DNA evolution does exist.

- Heuristics:
  - Perfect match “word” of the size: 7, 11 (default) or 15.



BLASTP

- Alignment matrix:
  - PAM and BLOSUM-series (default: BLOSUM 62)
- Notice: These alignment matrices incorporate knowledge about protein evolution.
- Heuristics:
  - 2 x “Near match” within a window.
  - Default word length: 3 aa
  - Default window length: 40 aa