Heuristic Alignment and Searching

Mark Voorhies

3/28/2012

Types of alignments

Global Alignment Each letter of each sequence is aligned to a letter or a gap (*e.g.*, Needleman-Wunsch).

Local Alignment An optimal pair of subsequences is taken from the two sequences and globally aligned (*e.g.*, Smith-Waterman).

Types of alignments

Global Alignment Each letter of each sequence is aligned to a letter or a gap (*e.g.*, Needleman-Wunsch).

Local Alignment An optimal pair of subsequences is taken from the two sequences and globally aligned (*e.g.*, Smith-Waterman). *This tends to be more biologically relevant*.

Smith-Waterman

The implementation of local alignment is the same as for global alignment, with a few changes to the rules:

- Initialize edges to 0 *(no penalty for starting in the middle of a sequence)*
- The maximum score is never less than 0, and no pointer is recorded unless the score is greater than 0 *(note that this implies negative scores for gaps and bad matches)*
- The trace-back starts from the highest score in the matrix and ends at a score of 0 *(local, rather than global, alignment)*

Because the naive implementation is essentially the same, the time and space requirements are also the same.

Timing CLUSTALW

Timing CLUSTALW from the command line:

```
for i in 50 100 150 200 250 300 350 400 450; do
    head -n $i -q G217B_iron.fasta Pb01_iron.fasta > temp.fasta;
    time clustalw -infile=temp.fasta -type=DNA -align;
done
```

The output looks like this:

```
Sequences (1:2) Aligned. Score:  0
Guide tree file created:   [temp.dnd]

There are 1 groups
Start of Multiple Alignment

Aligning...
Group 1:                    Delayed
Alignment Score 7238

CLUSTAL-Alignment file created  [temp.aln]


real 0m3.400s
user 0m3.388s
sys 0m0.012s
```

Timing CLUSTALW

You can copy the timing results into Excel.
Here is a Python script that does the same thing:

```
#!/usr/bin/env python
# Time-stamp: <ParseTimes.py 2011-03-29 21:10:59 Mark Voorhies>
"""Parse wall times from a log file on stdin and write them as a CSV
formatted column for Excel/OpenOffice/etc on stdout.  If command line
arguments are given, treat them as a second output column."""

from csv import writer
import re
time_re = re.compile("^real.*(?P<minutes>[\d]+)m(?P<seconds>[\d]+\.[\d]+)s",re.M)

if(__name__ == "__main__"):
    import sys
    args = sys.argv[1:]
    out = writer(sys.stdout)
    i = 0
    for t in time_re.finditer(sys.stdin.read()):
        try:
            y = args[i]
            i += 1
        except IndexError:
            y = ""
        out.writerow(
            (float(t.group("minutes"))*60+float(t.group("seconds")),y))

    del out
```

Timing CLUSTALW

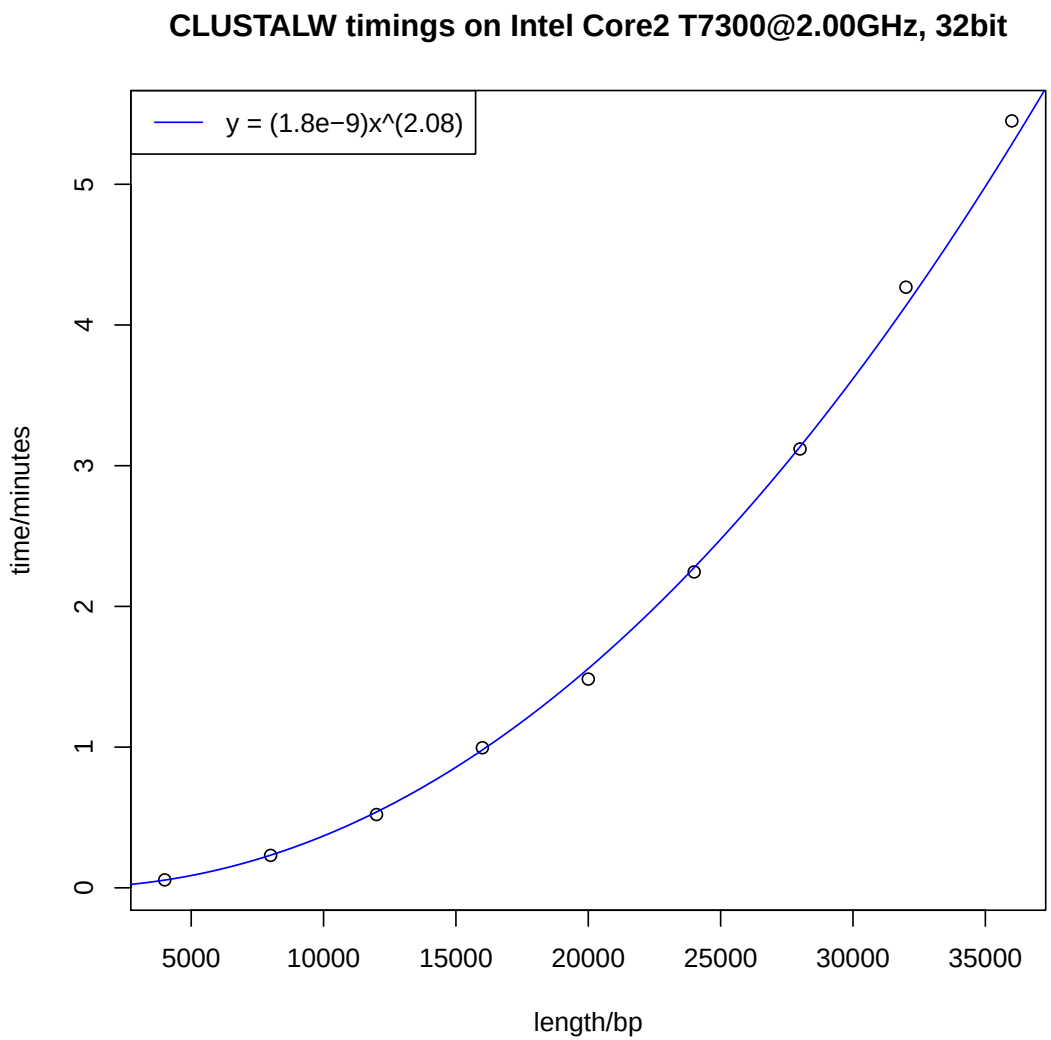
You can fit the timing results to a curve in Excel.

$$
\begin{aligned}
y &= Ax^B && (1)\\
\log y &= \log Ax^B && (2)\\
&= \log A + B \log x && (3)\\
&= A' + B \log x && (4)
\end{aligned}
$$

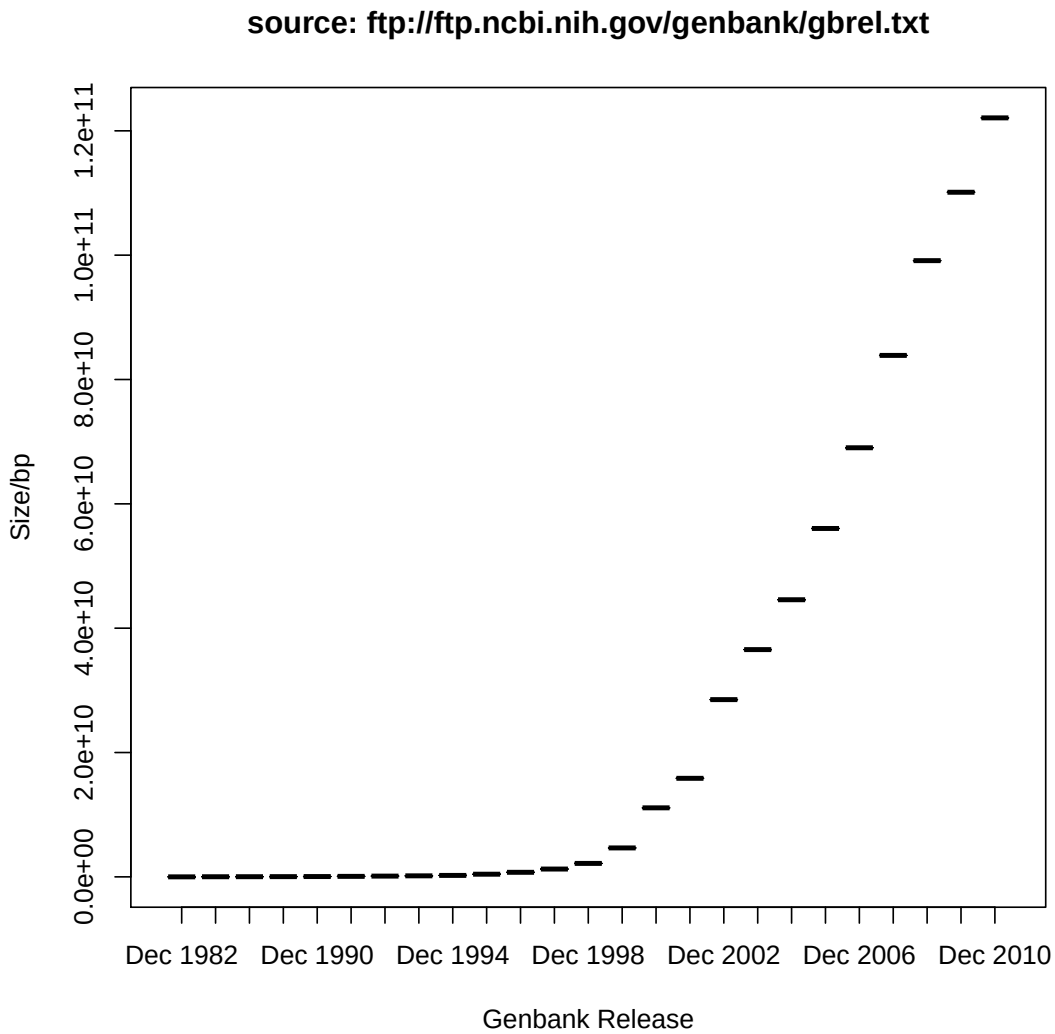
Here is an R script that does the same thing:

```
data <- read.csv("timings.csv", header = FALSE, col.names = c("t","n"))
x <- log(data$n*80)
y <- log(data$t/60)
f <- lm(y ~ x)
x0 <- 0:40000
a <- exp(f$coeff[1])
b <- f$coeff[2]
pdf("ClustalwTimings.pdf")
plot(data$n*80, data$t/60, xlab = "length/bp", ylab = "time/minutes",
     main = "CLUSTALW timings on Intel Core2 T7300@2.00GHz, 32bit")
points(x0, a*x0^b, col = "blue", type = "l")
legend("topleft", c("y = (1.8e-9)x^(2.08)"), col = "blue", lty = 1)
dev.off()
```

CLUSTALW takes $O(MN)$ time



$O(MN)$ time is too slow!



Basic Local Alignment Search Tool

Why BLAST?

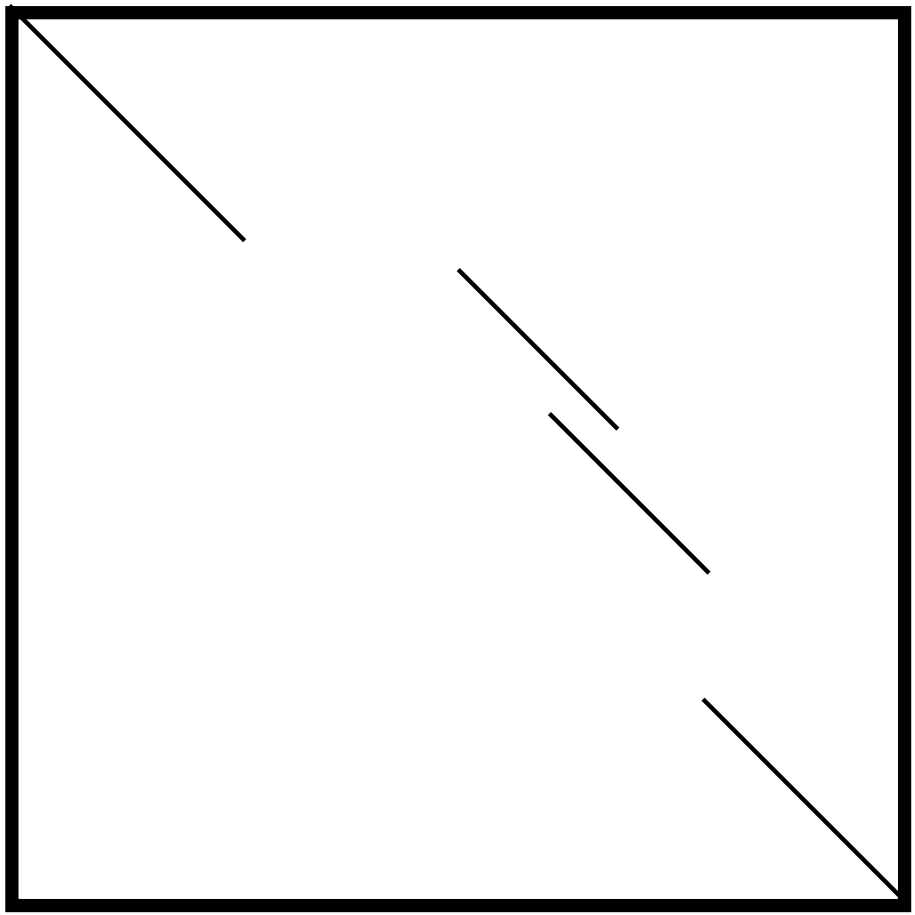
- Fast, heuristic approximation to a full Smith-Waterman local alignment
- Developed with a statistical framework to calculate expected number of false positive hits.
- Heuristics biased towards "biologically relevant" hits.

Seeding searches

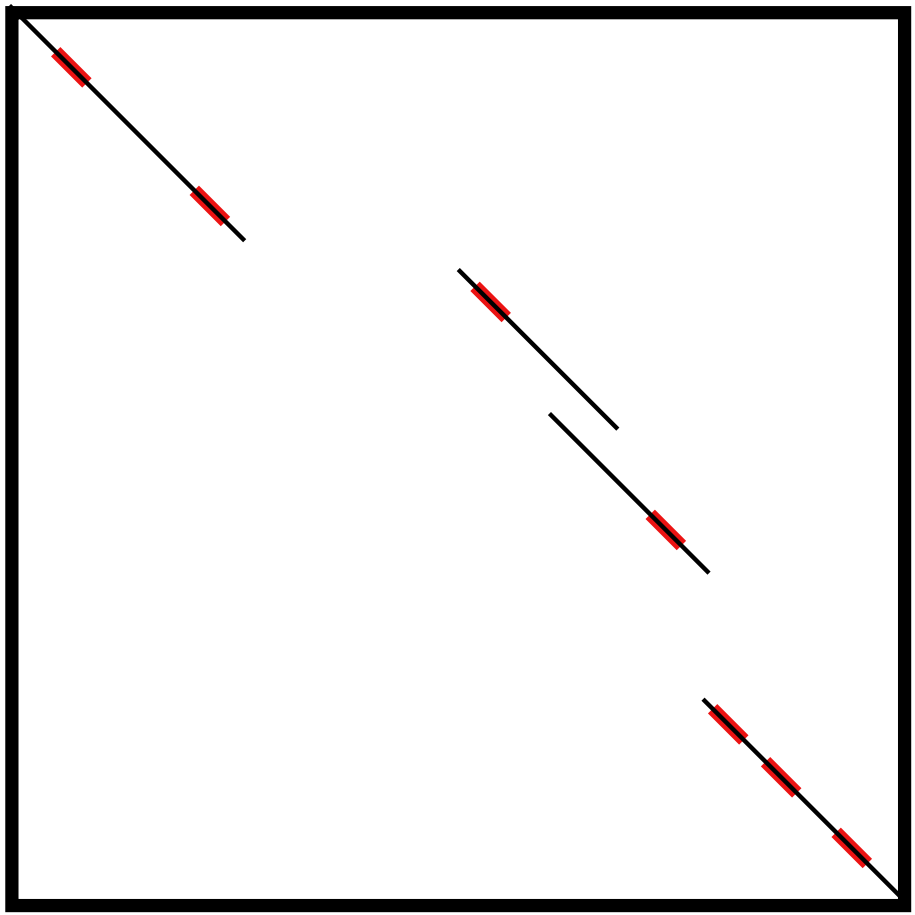
Most of the magic in a sequence-search tool lives in its indexing scheme

Program	Purpose	Indexing
BLAST	Database searching	Target indexing, 3aa or 11nt words
BLAT	mRNA mapping	Query indexing
BOWTIE	RnaSeq	Specialized index for low quality, short reads
e-PCR	Simulated PCR	Annealing-oriented index

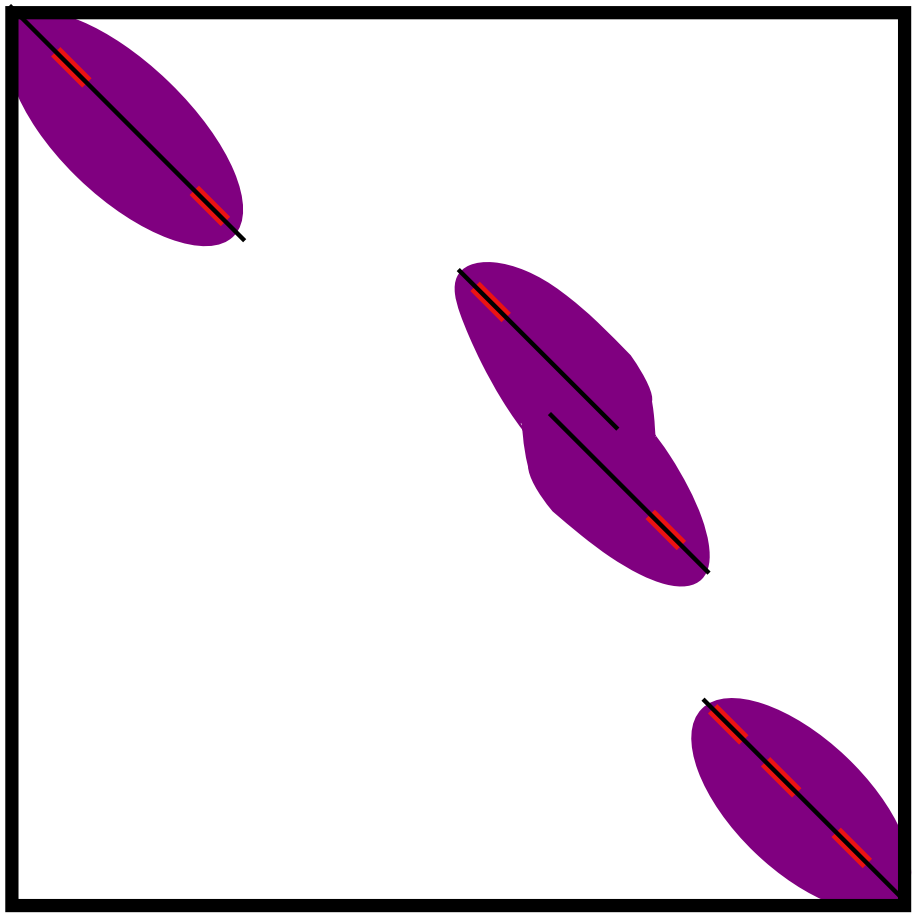
BLAST: A quick overview



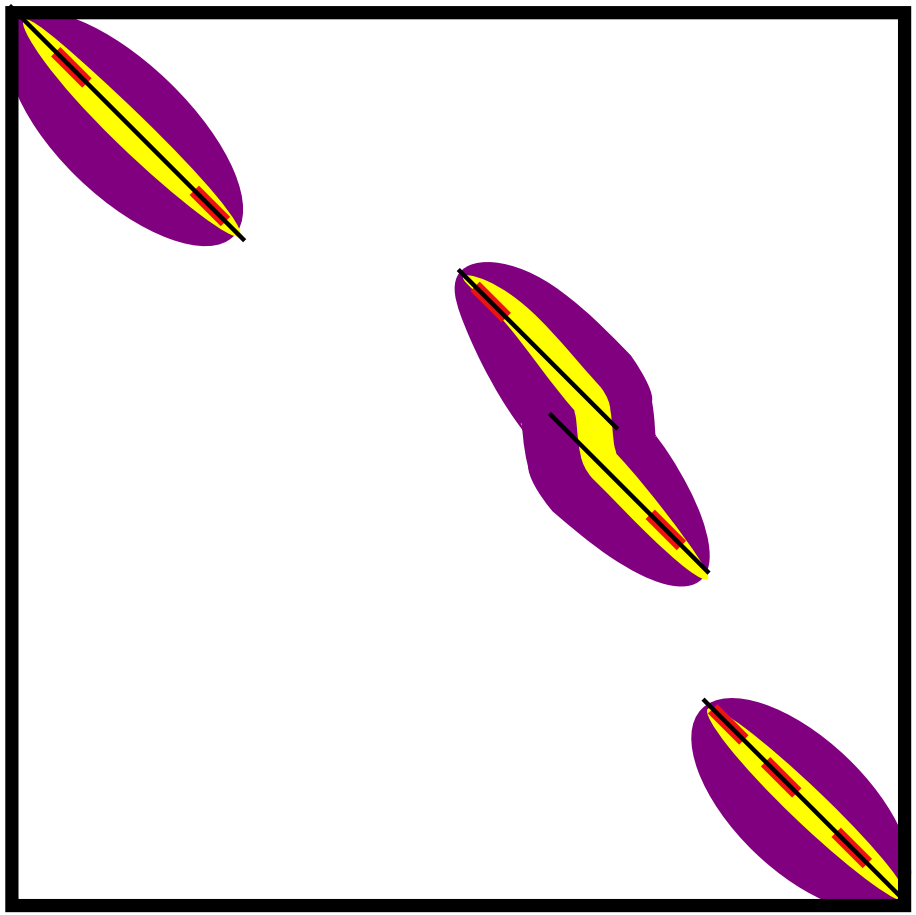
BLAST: Seed from exact word hits



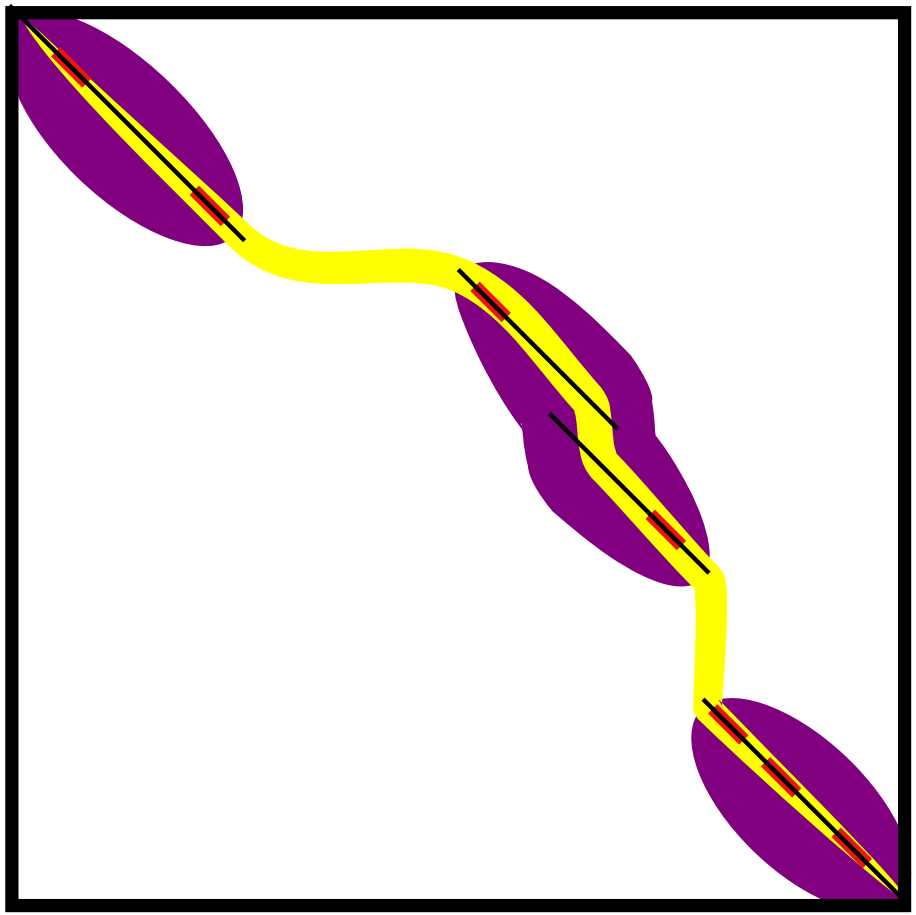
BLAST: Myers and Miller local alignment around seed pairs



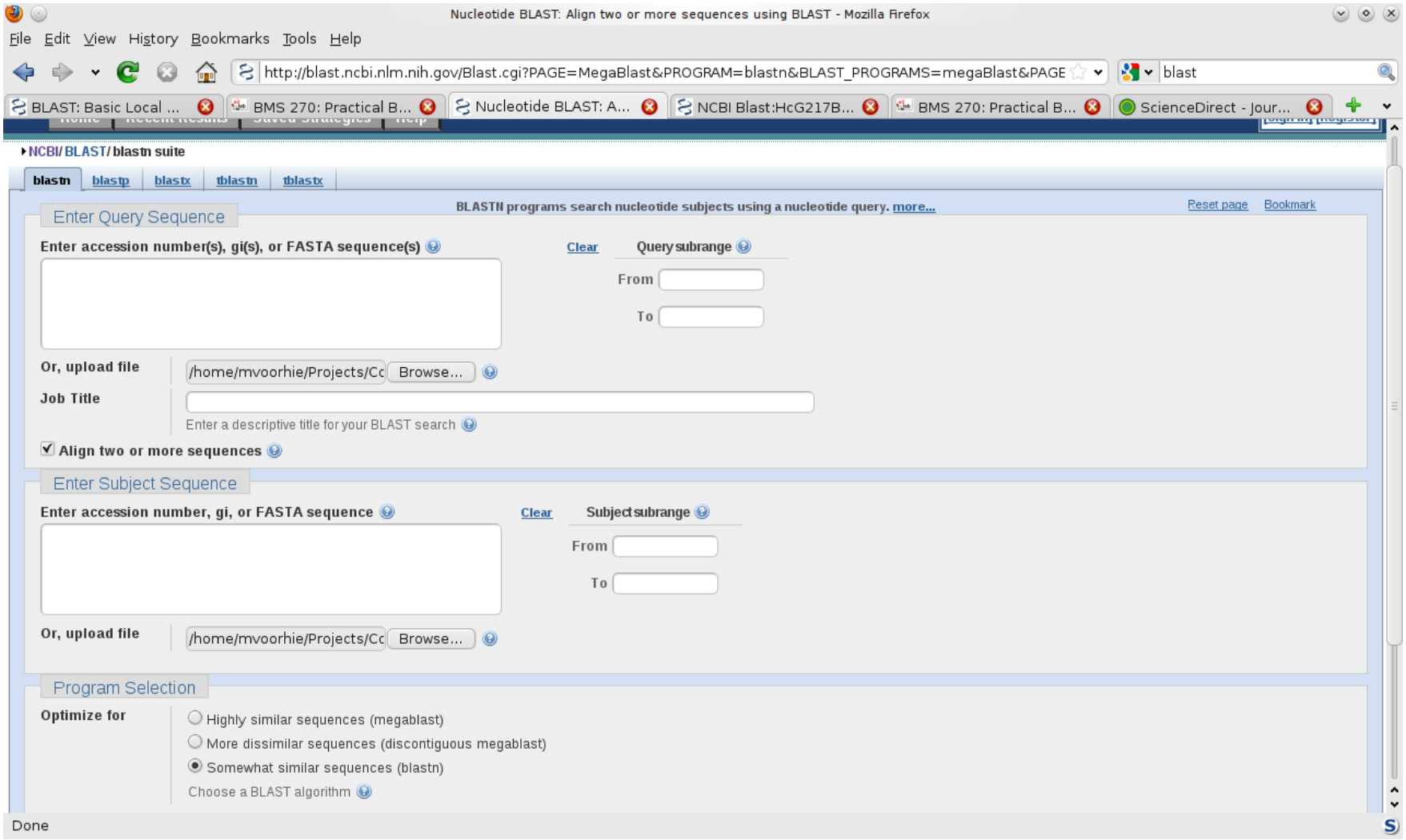
BLAST: High Scoring Pairs (HSPs)



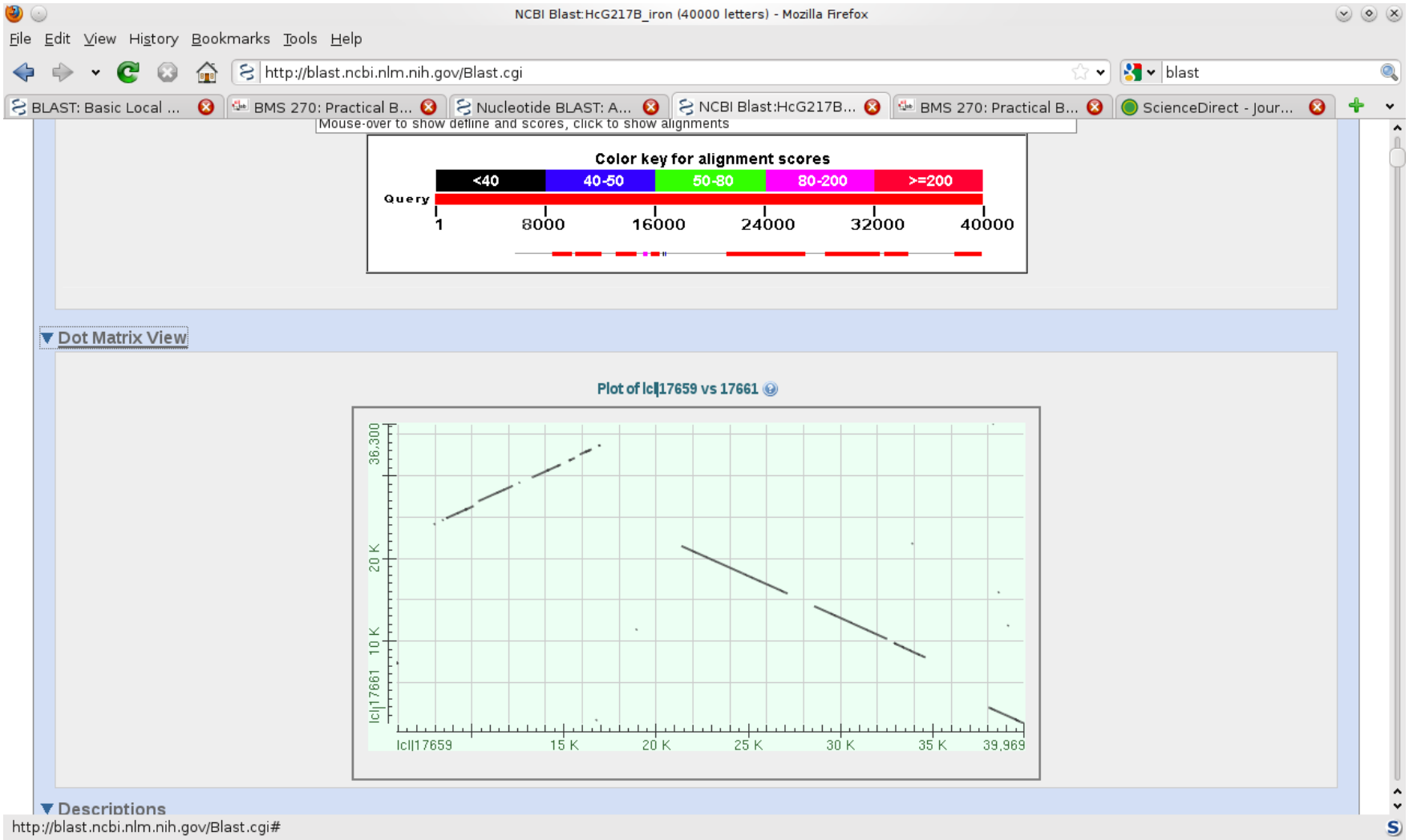
Gapped BLAST: Merge neighboring HSPs



How fast is BLAST?



How fast is BLAST?



How fast is BLAST?

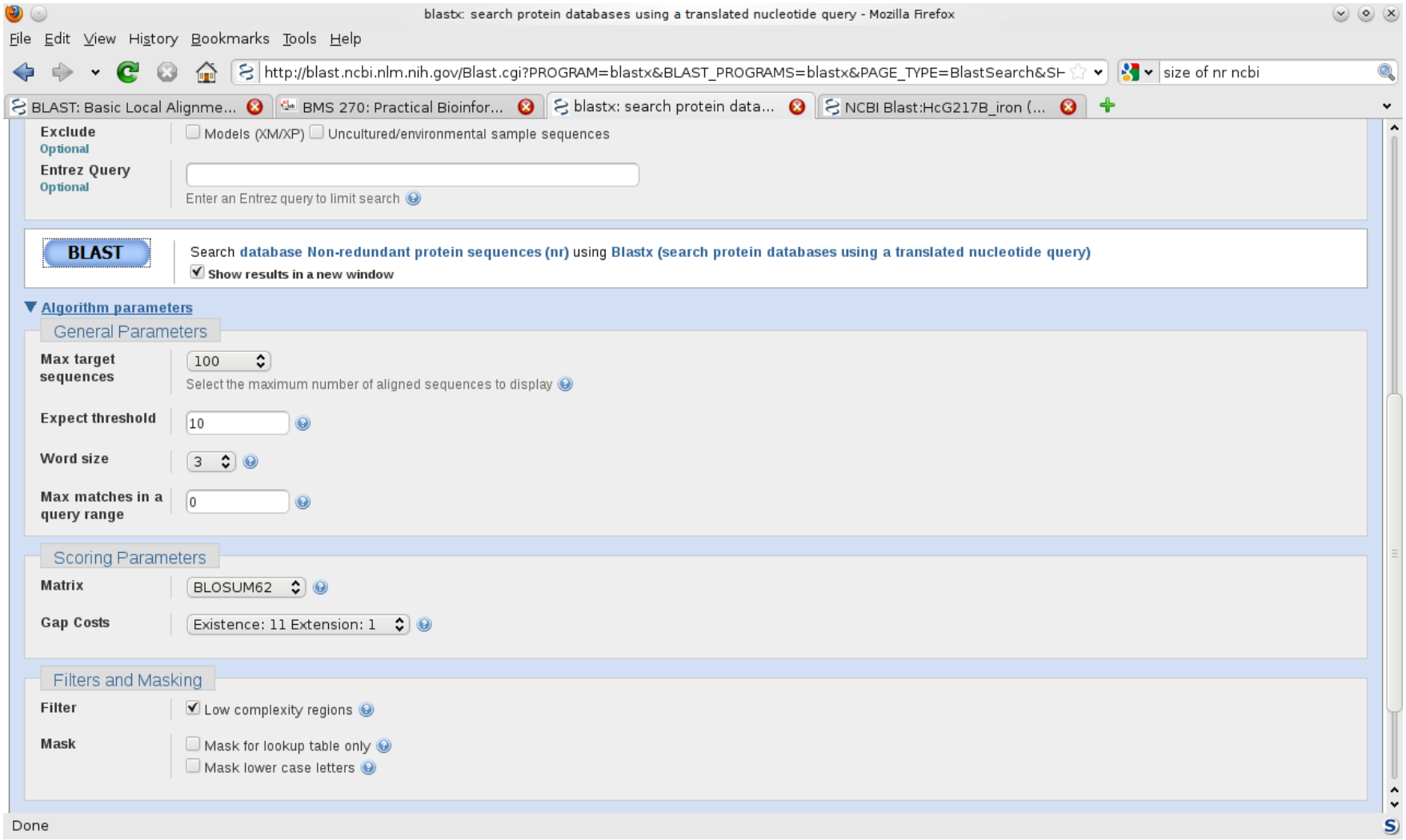
```
time bl2seq -p blastn -i G217B_iron.fasta -j Pb01_iron.fasta -e 1e-6 > temp.blastn

real    0m0.342s
user    0m0.080s
sys     0m0.032s
```

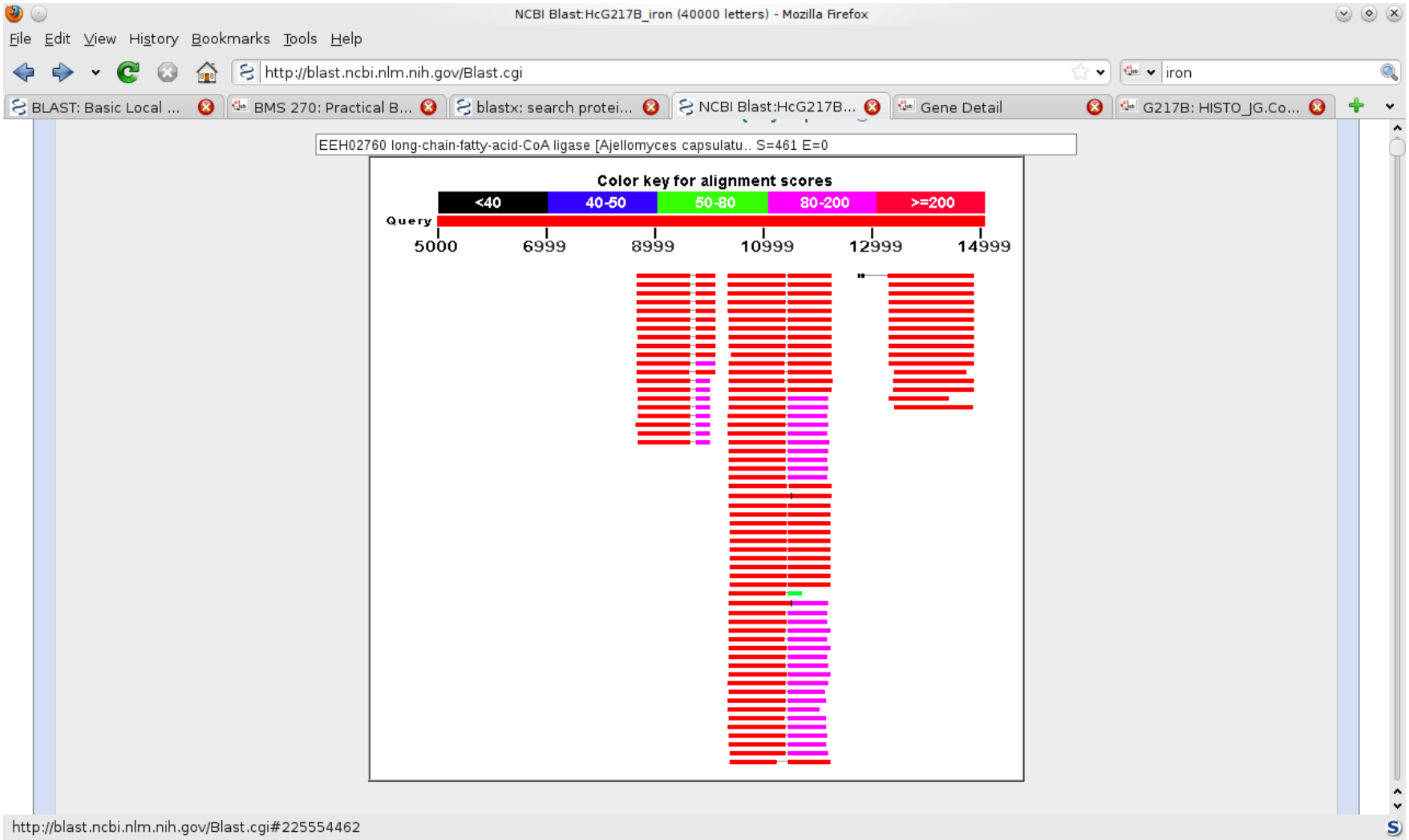
The basic flavors of BLAST

Target Query	Protein	DNA
Protein	BLASTP	TBLASTN
DNA	BLASTX	BLASTN TBLASTX

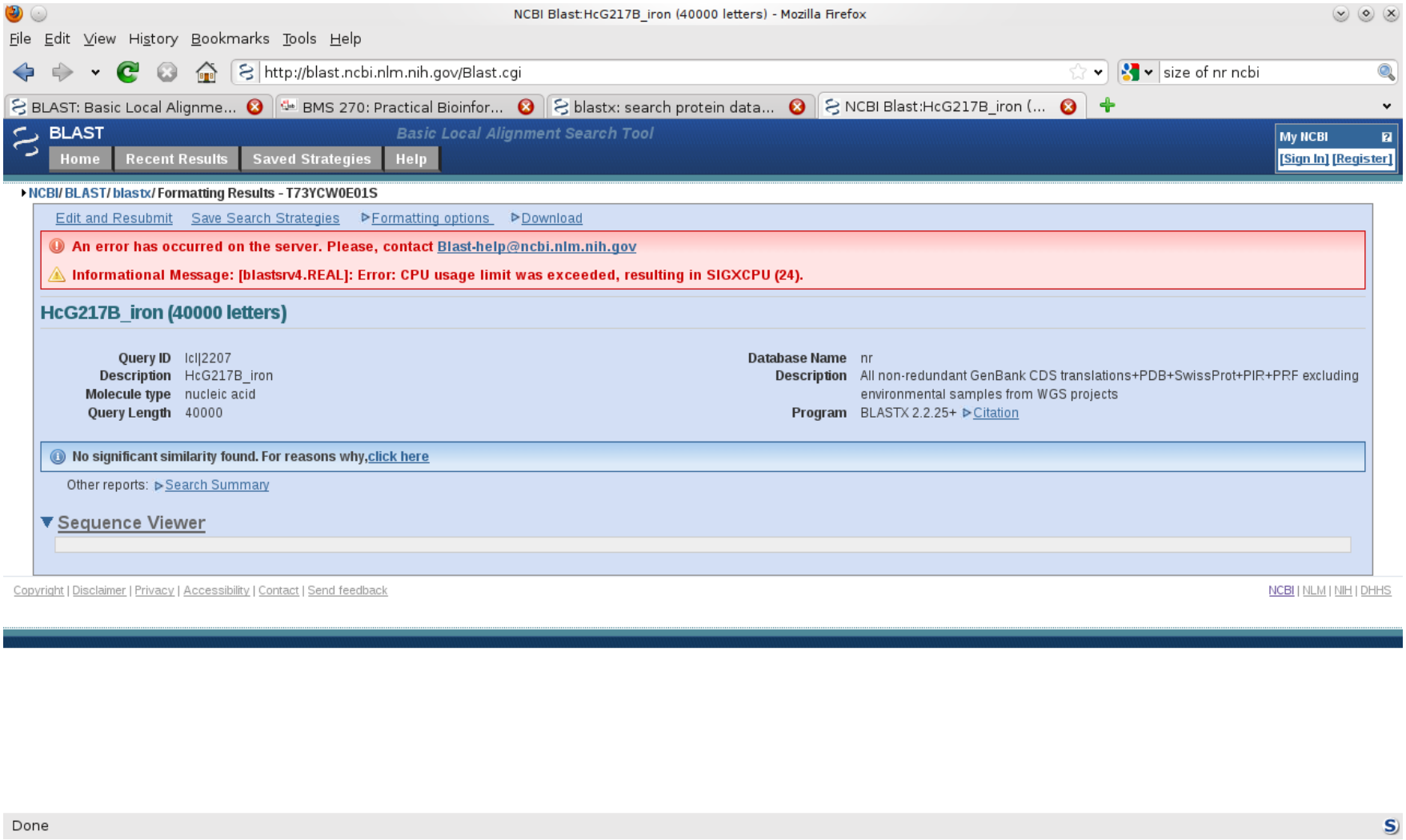
BLASTX: Nucleotide query vs. Protein Database



BLASTX: Nucleotide query vs. Protein Database



Sometimes it's still worth running locally...



Karlin-Altschul Statistics

$$E = kmne^{-\lambda S} \tag{5}$$

- S: HSP score
- E: Expected number of "random" hits in a database of this size scoring *at least* S.
- m: Query length
- n: Database size
- k: Correction for similar, overlapping hits
- λ: normalization factor for scoring matrix

Karlin-Altschul Statistics

$$E = kmne^{-\lambda S} \tag{5}$$

- S: HSP score
- E: Expected number of "random" hits in a database of this size scoring *at least* S.
- m: Query length
- n: Database size
- k: Correction for similar, overlapping hits
- λ: normalization factor for scoring matrix

A variant of this formula is used to generate sum probabilities for combined HSPs.

Karlin-Altschul Statistics

$$E = kmne^{-\lambda S} \tag{5}$$

- S: HSP score
- E: Expected number of "random" hits in a database of this size scoring *at least* S.
- m: Query length
- n: Database size
- k: Correction for similar, overlapping hits
- λ: normalization factor for scoring matrix

A variant of this formula is used to generate sum probabilities for combined HSPs.

$$p = 1 - e^{-E} \tag{6}$$

Karlin-Altschul Statistics

$$E = kmne^{-\lambda S} \tag{5}$$

- S: HSP score
- E: Expected number of "random" hits in a database of this size scoring *at least* S.
- m: Query length
- n: Database size
- k: Correction for similar, overlapping hits
- λ: normalization factor for scoring matrix

A variant of this formula is used to generate sum probabilities for combined HSPs.

$$p = 1 - e^{-E} \tag{6}$$

(If you care about the difference between E and p, you're already in trouble)

Karlin-Altschul Statistics

Important points:

- Extreme value distribution
- Assumption of infinite sequence length
- No rigorous framework for gap statistics (hmmer3 tries to fill this gap)

Summary

- BLAST is very fast, at the expense of not guaranteeing globally optimal results

Summary

- BLAST is very fast, at the expense of not guaranteeing globally optimal results
- But the trade-offs that it makes are biased towards "biologically relevant" results

Summary

- BLAST is very fast, at the expense of not guaranteeing globally optimal results
- But the trade-offs that it makes are biased towards "biologically relevant" results
- And it provides a statistical framework for evaluating its results.

Summary

- BLAST is very fast, at the expense of not guaranteeing globally optimal results
- But the trade-offs that it makes are biased towards "biologically relevant" results
- And it provides a statistical framework for evaluating its results.
- We can, and should, treat our computer work as we would an experiment:
  - Document protocols and observations
  - Run positive and negative controls
  - Keep results organized and dated

Homework

- Search your favorite proteins and collate interesting hits in one FASTA file per query – play with adding informative names and annotations (we will use these FASTA files tomorrow).
- Play with the BLAST book protocols (chapter 9) on the NCBI website
- Play with positive and negative controls (including permuted sequences)